UNIVERSIDADE FEDERAL DE SANTA CATARINA CENTRO TECNOLÓGICO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

Fabíola Maria Kretzer

Desenvolvimento de uma Unidade Instrucional para Formação de Professores da Educação Básica para o Ensino de Computação

Florianópolis
2019.2

Fabíola Maria Kretzer

Desenvolvimento de uma Unidade Instrucional para Formação de Professores da Educação Básica para o Ensino de Computação

Trabalho de Conclusão de Curso submetido ao Departamento de Informática e Estatística em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientadora: Prof.ª Christiane Gresse von Wangenheim, PMP, Dr.ª rer. nat.

Florianópolis
2019.2

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Kretzer, Fabíola Maria
Desenvolvimento de Curso para Formação de Professores da Educação Básica para o Ensino de Computação / Fabíola Maria Kretzer ; orientador, Christiane Gresse von Wangenheim, 2019.
180 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2019.

Inclui referências.

1. Ciências da Computação. 2. programação. 3. educação básica. 4. aprendizagem. 5. professores. I. Gresse von Wangenheim, Christiane. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

Fabíola Maria Kretzer

Desenvolvimento de uma Unidade Instrucional para Formação de Professores da Educação Básica para o Ensino de Computação

Trabalho de conclusão de curso submetido ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharelado em Ciências da Computação.

Florianópolis, 20 de novembro de 2019

Banca Examinadora:

_____

Prof.ª Dr.ª rer. nat. Christiane Gresse von Wangenheim, PMP
Orientadora
Universidade Federal de Santa Catarina

_____

Prof. Dr. Jean Carlo R. Hauck
Avaliador
Universidade Federal de Santa Catarina

_____

Ms. Giselle Araújo e Silva de Medeiros
Avaliadora

Florianópolis
2019.2

Este trabalho é dedicado aos meus queridos pais que, com muito carinho e apoio, acreditaram nos meus sonhos e não mediram esforços para que eu chegasse até esta etapa da minha vida.

AGRADECIMENTOS

Agradeço a Deus por ter me concedido a dádiva da vida, a força e a coragem necessária para a condução deste trabalho.

Agradeço aos meus pais pela paciência, incentivo e confiança ao longo de toda a graduação, em especial durante a execução deste trabalho.

Agradeço a minha orientadora Profª. Dr. Christiane Gresse von Wangenheim pela sua disponibilidade e incentivo que foram fundamentais para realizar e prosseguir este estudo.

Agradeço a minha família pelo apoio e contribuição no meu desenvolvimento pessoal e profissional.

Agradeço aos colegas do Grupo de Qualidade de Software – GQS/UFSC e a todos os colegas do Instituto Nacional para Convergência Digital – INCoD/UFSC pelo apoio oferecido no decorrer deste trabalho.

Agradeço as minhas amigas e amigos pelas conversas e coragem que me foram transmitidos.

Agradeço a todos os professores do curso e das escolas que estudei, pela importância na minha vida escolar e acadêmica.

Agradeço a Universidade Federal de Santa Catarina – UFSC por todas as experiências profissionais e pessoais que tive até o presente momento.

RESUMO

Atualmente, a computação tornou-se cada vez mais influente no dia-a-dia dos cidadãos. Os dispositivos eletrônicos se tornaram onipresentes, ao ponto de as pessoas não saberem mais viver sem tecnologia. Devido a essa importância, a computação precisa ser popularizada na Educação Básica, dando aos alunos a oportunidade de aprender competências básicas em computação. Ao proporcionar oportunidades de aprendizado de computação na escola, se estimula os alunos a criar, inovar e a prosperar em um ambiente de mudanças rápidas e contínuas. Essa integração de ensino de computação eficaz na escola exige professores motivados, dedicados e com competências computacionais, pedagógicas e tecnológicas, a fim de ensinar computação de uma forma que realmente envolva os alunos. No entanto, a quantidade de professores preparados não é suficiente e juntamente com a sobrecarga no currículo escolar na Educação Básica, dificulta oferecer a aprendizagem desse conteúdo aos alunos. Por isso, é necessário proporcionar formação continuada na área de computação aos professores de outras áreas que estejam atuando em sala de aula. E, mesmo existindo algumas iniciativas que visam integrar a computação de forma interdisciplinar, muitos desses programas ainda não ensinam aspectos pedagógicos e tecnológicos relacionados ao ensino de computação. Neste contexto, o presente projeto visa o desenvolvimento sistemático de uma unidade instrucional para ensinar conteúdos de computação (como algoritmos e programação) alinhada aos currículos de referência do *ACM/CSTA K-12 Computer Science Framework* e da SBC voltada aos Anos Finais do Ensino Fundamental. Também busca proporcionar aos professores competências pedagógicas e tecnológicas. A unidade instrucional tem o intuito de ensinar competências e habilidades de computação, por meio de atividades fechadas de programação utilizando o Code.org, desenvolvimento de jogos para *desktops*, utilizando o ambiente de programação visual Scratch e aplicativos para celulares *Android*, utilizando o ambiente de programação visual App Inventor. A unidade instrucional também pretende ensinar conhecimento pedagógico e tecnológico para os professores entenderem como aplicar os conteúdos de computação. Os primeiros resultados de uma aplicação indicam que a unidade instrucional pode contribuir com formação dos professores de forma agradável e divertida. Além disso, vários professores mostraram interesse em aprender mais sobre os assuntos envolvidos e também indicam uma ótima experiência de aprendizado. Espera-se que os resultados obtidos no presente trabalho proporcionem a popularização da computação no contexto da Educação Básica, para que os cidadãos aprendam desde cedo habilidades de computação. A longo prazo espera-se que estimule mais jovens a utilizar computação nas suas vidas profissionais.

Palavras-chave: computação, programação, unidade instrucional, educação básica, ambiente de programação visual, aprendizagem, professores.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM - *Association for Computing Machinery*

BNCC - Base Nacional Comum Curricular

CSTA - *Computer Science Teachers Association*

MOOC - *Massive Open Online Courses*

TI - Tecnologia da Informação

UI - Unidade Instrucional

SUMÁRIO

1. INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A influência da computação é sentida pelas pessoas diariamente e experimentada em um nível pessoal, social e global (CSTA, 2016). A computação impulsiona a criação de empregos e a inovação, ocupando um papel muito importante em toda a economia e sociedade (ANDRADE *et al.*, 2018). O computador possui a capacidade de representar a realidade física como um mundo virtual, na qual transforma e realiza processamento de ideias, imagens e informações em aplicativos, animações ou carros autônomos (CSTA, 2016). Num mundo rodeado de tecnologia, os cidadãos bem preparados, precisam ter uma compreensão clara dos princípios e práticas da computação, o que vai além de simples uso de TI (Tecnologia da Informação) (CSTA, 2016). Consequentemente, é importante que a pessoas aprendam sobre computação desde a infância.

Os avanços tecnológicos e o número crescente de profissões que dependem da computação tornam crucial que todos os alunos tenham oportunidades de se tornar não apenas alfabetizados em TI, mas consigam aprender de forma independente e utilizem a tecnologia a medida que ela evolui, incluindo o uso ativo da computação (CSTA, 2011). O aprendizado da computação faz o aluno desenvolver a criatividade, design, resolução de problemas, análise de uma variedade de possíveis soluções para um problema, colaboração e habilidades (GAL-EZER & STEPHENSON, 2010). Essas competências estimulam os alunos a criar, inovar e prosperar em um ambiente de mudanças rápidas e contínuas (NAUGHTON, 2012). Deste modo, o aprendizado também permite que os alunos não sejam apenas usuários de tecnologia, mas também possam projetar e construir futuras soluções tecnológicas (GAL-EZER & STEPHENSON, 2010).

Uma alternativa encontrada são iniciativas que buscam ensinar computação as crianças e adolescentes na Educação Básica, com foco em programação (GROVER & PEA, 2013) explorando diversas maneiras de ensino, como MOOC (DE KEREKI & MANATAKI, 2016), *workshops* (ROBINSON & PÉREZ-QUINÕNES, 2014), cursos (MISSFELDT FILHO, 2019), entre outros. Em geral, essas iniciativas utilizam

linguagens de programação baseada em blocos, como App Inventor (MITa, 2019) ou Scratch (MITb, 2019), as quais não exigem aprendizagem de sintaxe e semântica complexas, como nas linguagens textuais. Isso possibilita que os alunos se concentrem no aprendizado da lógica e das estruturas envolvidas na programação (GROVER & PEA, 2013).

Porém, a implementação de ensino de computação nas escolas exige professores motivados e dedicados com competências computacionais, pedagógicas e tecnológicas, a fim de ensinar computação de uma forma que realmente envolva os alunos (GAL-EZER & STEPHENSON, 2010). Idealmente, os professores de computação seriam preparados por meio de um rigoroso programa de formação de professores, normalmente por meio de graduação ou pós-graduação (BERGES *et al.*, 2013). Embora tais programas ofereçam aos professores uma compreensão profunda da disciplina, é um caminho lento que requer vários anos para ser concluído (GRANOR, DELYSER & WANG, 2016). A inscrição e o acompanhamento de tais programas é também cada vez mais limitado pelo crescimento contínuo do emprego na indústria, já que o incentivo para seguir carreiras lucrativas na indústria de TI é muito maior do que o incentivo para ensinar (GOODE, 2007). Como resultado, os professores de computação nas escolas são muito escassos (NI & GUZDIAL, 2012) (INEP, 2015) e os alunos da Educação Básica possuem poucas oportunidades de aprender computação e de analisar como a computação influencia sua vida cotidiana (CSTA, 2016).

Outra possibilidade para implementar o ensino de computação em maior escala na Educação Básica, pode ser de forma interdisciplinar (LYE & KOH, 2014; VON WANGENHEIM *et al.*, 2017). Deste modo, os alunos podem aprender computação no contexto da disciplina de formação e atuação do professor. Um exemplo é a unidade instrucional (UI) oferecida a alunos da disciplina de história em escolas brasileiras em nível do Ensino Fundamental, na qual são criados jogos digitais com o tema história/cultura de Santa Catarina e Civilizações Antigas (ALVES, 2016). Nesse cenário, os professores já possuem conhecimento de conteúdo da área de estudo que ensinam (por exemplo, história, artes, ciências, etc.), conhecimento pedagógico geral, mas tipicamente não possuem competências de computação (VON WANGENHEIM *et al.*, 2017). A fim de ensinar computação de maneira eficaz e envolvente, os

professores precisam ter conhecimento de conteúdo em computação (especialmente algoritmos e programação), conhecimento de conteúdo pedagógico (como ajudar os alunos a aprender computação), e também conhecimento tecnológico (para instalar e manter a infraestrutura de TI necessária) (LIU *et al.*, 2011). Dessa forma, é importante recrutar e ensinar computação a professores *in-service* que possuem credenciais em outras áreas de estudo (GOODE, 2008; COOPER *et al.*, 2015). Consequentemente, é importante oferecer a formação continuada para criar condições a curto prazo para a popularização da computação nas escolas (LAMPROU, REPENNING & ESCHERLE, 2017).

Assim, este projeto tem o objetivo de desenvolver uma unidade instrucional para proporcionar aos professores *in-service* (atuantes em outras disciplinas como, artes, história, geografia e outras) a oportunidade de aprender competências básicas de computação, as quais permitam a integração do ensino de computação as disciplinas nessas respectivas áreas.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo geral deste projeto é o desenvolvimento sistemático de uma unidade instrucional na forma de um curso presencial de aproximadamente 40 horas/aula, para ensinar aos professores *in-service* da Educação Básica, conceitos básicos e a prática de programação. Serão incorporadas atividades fechadas de programação no Code.org (CODE.ORG, 2019) e atividades abertas de desenvolvimento de jogos no Scratch (MITb, 2019) e de aplicativos no App Inventor (MITa, 2019). A unidade instrucional é alinhada ao currículo de referência do ensino de computação da ACM/CSTA (2016) e SBC (2018) principalmente voltado aos Anos Finais do Ensino Fundamental. A unidade instrucional é projetada de forma que permita a sua adoção de forma interdisciplinar inserida em conteúdo programado do Ensino Fundamental, demonstrando aspectos pedagógicos referentes ao ensino da computação e também aspectos tecnológicos para possibilitar ao professor preparar a infraestrutura necessária para as aulas.

Este projeto visa todo o design instrucional de todas as atividades relativas à unidade instrucional, incluindo análise de contexto, definição dos objetivos de aprendizagem, definição da estratégia instrucional e a preparação dos recursos didáticos a serem utilizados durante a sua execução. Como resultado, é criado um plano de ensino, todo o material didático para acompanhar a UI, apresentações para exposição de conceitos e exemplos, exercícios propostos e avaliações da aprendizagem.

Visa-se também a aplicação e avaliação da unidade com o objetivo de analisar a qualidade tanto da percepção de ensino dos professores quanto outros aspectos. Com isso, espera-se gerar uma forma eficaz de ensinar professores do Ensino Fundamental sobre importantes conceitos de computação e também fornecer habilidades e conhecimentos a professores de outras disciplinas que permitam a integração de computação nas aulas.

1.2.1 Objetivos específicos

Os objetivos específicos são:

O1. Sintetizar a fundamentação teórica;

O2. Analisar o contexto referente aos professores, ambiente, alunos e currículos de referência e definir os objetivos de aprendizagem;

O3. Levantar o estado da arte em relação a unidades instrucionais semelhantes;

O4. Definir o design instrucional da unidade instrucional (seleção e sequenciamento do conteúdo, estratégias de ensino, etc.);

O5. Desenvolver o material didático para a unidade instrucional;

O6. Aplicar e avaliar a unidade instrucional na prática.

1.3 METODOLOGIA DE PESQUISA E TRABALHO

A fim de alcançar os resultados esperados, a metodologia de pesquisa foi dividida em cinco etapas, conforme com o respectivo objetivo a ser buscado da seguinte forma:

Etapa 1 – Fundamentação teórica: focada em analisar e sintetizar os principais conceitos envolvidos nos temas a serem abordados neste trabalho. Este estudo é realizado por meio de das seguintes atividades:

Atividade 1.1: Sintetizar conceitos do processo de ensino e aprendizagem;

Atividade 1.2: Sintetizar conceitos de ensino de computação na educação Básica;

Atividade 1.3: Sintetizar a teoria sobre formação de professores;

Atividade 1.4: Sintetizar ambientes de programação utilizados para ensinar computação na Educação Básica.

Etapa 2 - Mapeamento do estado da arte: etapa para mapear os trabalhos existentes relacionados a área do projeto. O mapeamento é realizado seguindo um processo proposto por Petersen, Vakkalanka e Kuniarz (2015) para identificar e analisar as estratégias de ensino atualmente utilizadas e voltadas ao ensino de computação para professores. As atividades a seguir são definidas para dividir esta etapa.

Atividade 2.1: Definir o protocolo de busca;

Atividade 2.2: Executar a busca;

Atividade 2.3: Extrair e analisar as informações.

Etapa 3 - Design da unidade instrucional: engloba toda a parte de planejamento e design da unidade instrucional a ser realizada. Também envolve a análise do contexto da unidade instrucional a ser desenvolvida, identificando características e restrições em relação ao público alvo, à infraestrutura e ao contexto educacional. A definição do design da unidade, segue a metodologia utilizada no modelo ADDIE (BRANCH, 2009), na qual orienta a construção de ferramentas eficazes de treinamento e como a unidade instrucional deve suceder na prática.

Atividade 3.1: Analisar o contexto da unidade instrucional a ser desenvolvida.

Atividade 3.1.1: Analisar o contexto em termos de perfil dos aprendizes (professores);

Atividade 3.1.2: Analisar o contexto em termos de ambiente em escolas brasileiras;

Atividade 3.1.3: Analisar o contexto em termos do perfil dos alunos;

Atividade 3.2: Definir e sequenciar o conteúdo da unidade instrucional e definir uma estratégia instrucional;

Atividade 3.3: Definir e planejar a avaliação da aprendizagem.

Etapa 4 - Desenvolvimento da unidade instrucional: é realizado o desenvolvimento de todo o material didático para a aplicação da unidade instrucional.

Atividade 4.1: Desenvolver o material de apresentação a ser utilizado durante as aulas;

Atividade 4.2: Desenvolver atividades e exercícios a serem aplicadas durante as aulas;

Atividade 4.3: Desenvolver avaliações do desempenho do aluno;

Atividade 4.4: Desenvolver materiais para os professores poderem aplicar o aprendizado adquirido na unidade instrucional em sala de aula.

Etapa 5 - Aplicação e avaliação da unidade instrucional: nesta etapa são identificadas as atividades relacionadas a aplicação da unidade instrucional e análise dos dados para avaliar o que foi proposto, de forma a aplicá-la na prática por meio de uma série de estudos de casos (YIN, 2009).

Atividade 5.1: Definição da avaliação;

Atividade 5.2: Submissão do projeto a CEPSH/UFSC (Comitê de Ética em Pesquisa com Seres Humanos);

Atividade 5.3: Aplicação da unidade instrucional e coleta de dados;

Atividade 5.4: Analisar e interpretar os dados.

1.4 ESTRUTURA DO DOCUMENTO

O capítulo 2 apresenta a fundamentação teórica dos conceitos relacionados com o desenvolvimento da unidade instrucional. No capítulo 3 é apresentado um mapeamento sistemático do estado da arte em relação às unidades instrucionais existentes que ensinam habilidades e conhecimentos de computação para professores *in-service* da Educação Básica. O capítulo 4 apresenta o design da

unidade instrucional, com informações sobre o público-alvo, características das escolas brasileiras e o detalhamento do plano de ensino, rubricas de avaliação e materiais didáticos. No capítulo 5 é apresentada a aplicação e avaliação da unidade instrucional. Finalmente, o capítulo 6 apresenta as conclusões finais sobre o trabalho, apresentando os principais resultados e melhorias da unidade desenvolvida.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais conceitos referentes ao tema deste trabalho. Também é apresentado o processo de ensino e aprendizagem e a formação de professores. É também apresentado os principais ambientes de programação visual baseado em blocos.

2.1 PROCESSO DE ENSINO E APRENDIZAGEM

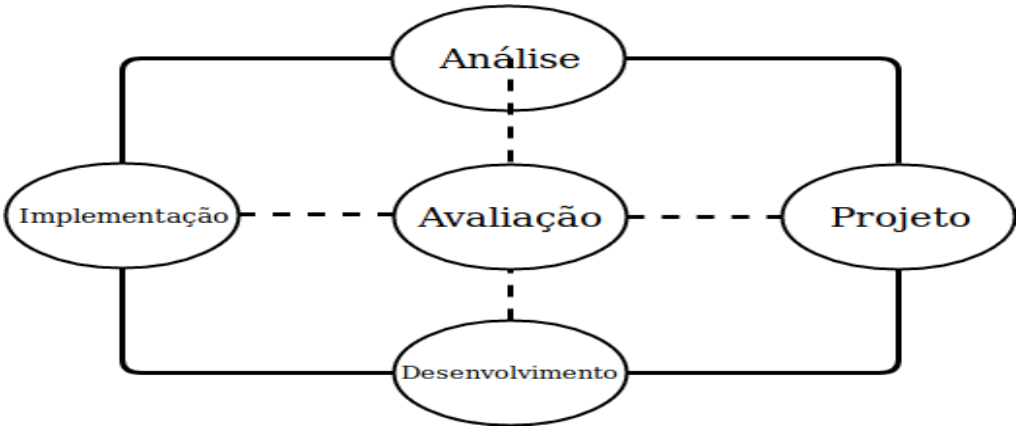
O processo de ensino e aprendizagem abrange diversos questionamentos, como a própria definição de ensinar e aprender (FREITAS, 2016). O termo ensinar é empregado com sentido de "instruir sobre", "transmitir conhecimentos a", "educar", "dar aulas", "doutrinar" e "pregar" (MICHAELIS, 2009). Enquanto o termo aprender expressa "ficar sabendo", "reter na memória" e "tomar conhecimento" (MICHAELIS, 2009). Porém, Freire (1971) condena essas definições contidas no dicionário e relata que não são compatíveis com a realidade da educação. A aprendizagem é o processo de assimilação de qualquer forma de conhecimento, desde o mais simples onde a criança aprende a brincar, até processos mais complexos onde uma pessoa aprende uma profissão (FREITAS, 2016). O ensino busca estimular, incentivar e impulsionar o processo de aprendizagem dos alunos (LIBÂNEO, 1994). Assim, o processo de ensinar define-se por obter aprendizagem do aluno e não pela intenção do professor ou por uma descrição do que ele faz em sala de aula (KUBO & BOTOMÉ, 2001). Desta forma, esses dois processos podem ser considerados dependentes, pois o objetivo de ensinar é que o outro aprenda, embora possa aprender sem um professor (KUBO & BOTOMÉ, 2001). O ato de ensinar não pode ser percebido como algo mecânico, mas deve ser um processo que permita o alcance da aprendizagem (LIBÂNEO, 1994).

Durante esse processo é necessário atenção a falta de conhecimento por parte dos alunos em alguns pré-requisitos do conteúdo a ser aprendido, ou seja, os conteúdos passados anteriormente pelo professor (FREITAS, 2016). Assim, a decisão e a definição dos objetivos de aprendizagem devem ser de forma consciente, estruturando o processo educacional (FERRAZ & BELHOT, 2010) e fazendo o professor assumir o papel de facilitador da aprendizagem (LIBÂNEO, 1994).

A estruturação do ensino e aprendizagem é resultado de um processo de design instrucional que está diretamente relacionado a escolha do conteúdo e das atividades, dos recursos disponíveis e da metodologia a ser adotada (FERRAZ & BELHOT, 2010). O desenvolvimento da UI tipicamente segue o design instrucional, se caracterizando como um processo iterativo que inclui definição de objetivos, seleção de estratégias instrucionais, seleção e criação de materiais necessários, aplicação e avaliação de UI (BRANCH, 2009). O design instrucional é utilizado para desenvolver unidades instrucionais de forma consistente e confiável (BRANCH, 2009). Assim, a unidade instrucional é definida como um conjunto de aulas planejadas para ensinar determinados objetivos de aprendizagem a um público-alvo específico (HILL, ROWAN & BALL, 2005). Também possui um conjunto de materiais instrucionais, tanto para professores quanto para alunos, desenvolvidos para oportunizar o aprendizado em um contexto específico (HILL, ROWAN & BALL, 2005).

O modelo ADDIE - *Analyze, Design, Develop, Implement and Evaluate* (Análise, Projeto, Desenvolvimento, Implementação e Avaliação) demonstra os componentes conceituais e indicam como o design instrucional deve suceder na prática (BRANCH, 2009). O modelo ADDIE é estruturado em fases (Figura 1) que facilitam a capacidade intencional de aprendizagem do aluno, aumentando assim o potencial para o seu sucesso (BRANCH, 2009).

Figura 1. Fases do modelo ADDIE.



Fonte: BRANCH (2009).

A fase de análise visa determinar as características do público-alvo e do ambiente onde ocorre a aplicação da unidade instrucional, definindo também os objetivos e as metas a serem alcançadas (BRANCH, 2009). Durante a identificação

do público-alvo é importante realizar um estudo sobre os conhecimentos, competências, atitudes, experiências, preferências e a motivação de cada aluno (BRANCH, 2009). Os recursos tecnológicos, humanos, do conteúdo e instalações no ambiente da unidade instrucional também merecem atenção (BRANCH, 2009). Uma parte significativa da aprendizagem requer uma progressão de conhecimento para o desconhecido e do simples ao complexo (BRANCH, 2009), necessitando de instrumentos para fornecer apoio didático-pedagógico.

Bloom e Simpson desenvolveram instrumentos (DRISCOLL, 2000) atualmente utilizados com o objetivo de facilitar o processo de ensino e aprendizagem e consequentemente colaborar com os educadores nas práticas educacionais (FERRAZ & BELHOT, 2010). Esses instrumentos são chamados de Taxonomia de Bloom, na qual é dividida em três domínios: cognitivo (Tabela 1), afetivo (Tabela 2) e psicomotor (Tabela 3). Os domínios cognitivo e afetivo foram estudados por Bloom e aprimorados por Simpson, surgindo o domínio psicomotor (DRISCOLL, 2000). A estruturação de cada domínio é realizada em níveis de complexidade crescente, ou seja, para adquirir um novo conhecimento, habilidade ou atitude do próximo nível, o aluno deve ter adquirido a do nível anterior (DRISCOLL, 2000).

O domínio cognitivo é caracterizado pela aquisição de novos conhecimentos e o desenvolvimento de pensamentos, ideias e percepções (BRANCH, 2009). Esse domínio também envolve a aprendizagem de conceitos que estimulam o desenvolvimento intelectual do aluno (FERRAZ & BELHOT, 2010). Para dividir as capacidades adquiridas por um aluno, os objetivos foram agrupados em seis categorias, formando uma hierarquia de dependência e complexidade (Tabela 1) (FERRAZ & BELHOT, 2010).

Tabela 1. Categorias do domínio cognitivo.

Categoria	**Descrição**
1. **Conhecimento**	Recordar o material previamente aprendido, incluindo fatos, conceitos, vocabulário e princípios
2. **Compreensão**	Aprender o conteúdo do material
3. **Aplicação**	Utilizar abstrações, regras, princípios, ideias e outras informações determinadas situações
4. **Análise**	Decompor o material em elementos ou partes constituintes

5. **Síntese**	Combinar elementos peças ou partes para formar o todo ou constituir um novo padrão ou estrutura
6. **Avaliação**	Fazer julgamentos sobre até que ponto os métodos ou materiais satisfazem os critérios existentes

Fonte: DRISCOLL (2000).

As características do domínio afetivo incluem atividades ligadas ao desenvolvimento da área emocional e afetiva (FERRAZ & BELHOT, 2010). Comportamento, atitude, responsabilidade, respeito, emoção e valores estão entre os termos que mais representam esse domínio (FERRAZ; BELHOT, 2010). A hierarquia (Tabela 2) apresenta a mesma ideia da estrutura do domínio cognitivo, com a diferença que este foi dividido em cinco categorias.

Tabela 2. Categorias do domínio afetivo.

Categoria	**Descrição**
1. Receptividade	Tornar-se sensibilizado ou disposto a receber certas informações
2. Resposta	Tornar-se envolvido ou fazer algo
3. Valorização	Exibir um compromisso com algo por causa de seu valor inerente
4. Organização	Organizar o conjunto de valores e determinar os relacionamentos, incluindo quais devem dominar
5. Caracterização	Integrando valores numa filosofia total e agindo consistentemente de acordo com essa filosofia

Fonte: DRISCOLL (2000).

A taxonomia para o domínio psicomotor foi desenvolvida por autor diferente dos pesquisadores dos domínios anteriores (DRISCOLL, 2000), mas apresenta a mesma ideia da estrutura (Tabela 3) das demais. Esse domínio inclui características ligadas ao desenvolvimento de reflexos, percepções, competências físicas e comunicação não verbal (FERRAZ & BELHOT, 2010).

Tabela 3. Categorias do domínio psicomotor.

Categoria	**Descrição**
1. Percepção	Tornar-se consciente da estimulação e necessidade de ação
2. Conjunto	Preparar para a ação
3. Resposta guiada	Responder com a ajuda de um professor ou treinador
4. Mecanismo	Responder habitualmente

5. Resposta complexa	Resolver incertezas e executar tarefas automaticamente
6. Adaptação	Alterar respostas para novas situações
7. Originalidade	Criar novos atos ou expressões

Fonte: DRISCOLL (2000).

A fase de projeto tem o propósito de definir a abordagem prática utilizada para manter um alinhamento de metas, objetivos, estratégias e avaliações aplicados na unidade instrucional (BRANCH, 2009). Utilizando os objetivos de aprendizagem sugeridos anteriormente, deve-se identificar as tarefas e os conteúdos necessários para atingi-los, baseando-se nas características dos estudantes, recursos necessários e competências dos membros da equipe (BRANCH, 2009). A organização dos conteúdos ensinados também é importante para a construção de competências e conhecimentos por parte dos alunos (BRANCH, 2009). Outra parte essencial é a medição de desempenho, fornecendo *feedback* ao professor sobre o aprendizado e o progresso do estudante (BRANCH, 2009). Dessa maneira, pode-se observar a forma como a unidade instrucional foi desenvolvida está proporcionando aprendizado aos alunos (BRANCH, 2009).

Na fase de desenvolvimento deve-se selecionar, criar e validar todos os recursos e ferramentas necessárias para realizar as aulas que serão ministradas durante a unidade instrucional (BRANCH, 2009). Ao final dessa fase é essencial a disponibilidade de um conjunto de recursos de aprendizagem (como o conteúdo, planos de aula), mídias (usada para apresentar conteúdo), orientações de interação entre professor e aluno, plano de avaliação e outros recursos (BRANCH, 2009). Também devem ser formulados as estratégias e métodos instrucionais, de acordo com o conteúdo e os objetivos de aprendizagem definidos previamente (SASKATCHEWAN EDUCATION, 1991). Essas estratégias devem acomodar a motivação dos alunos para o aprendizado, as taxas de aprendizado dos alunos e o estilo de aprendizado de cada aluno (BRANCH, 2009). Abaixo estão descritas as principais estratégias instrucionais e os métodos que cada uma utiliza (SASKATCHEWAN EDUCATION, 1991):

- **Instrução direta:** é eficiente para fornecer informações e desenvolver competências passo-a-passo. Essa estratégia funciona bem com o envolvimento ativo dos estudantes na construção do conhecimento e

geralmente é dedutiva, ou seja, o assunto é apresentado e depois ilustrado com exemplos. Utilizam métodos instrucionais como palestras, questionamento didático, exercitar e praticar, ensino explícito, prática, treinamento e demonstrações.

- **Instrução indireta:** busca envolvimento do aluno em observar, investigar, extrair inferências a partir de dados e pensar em hipóteses, tirando proveito do interesse e da curiosidade dos alunos. Também promove a criatividade e o desenvolvimento de competências. Os métodos instrucionais incluem discussão reflexiva, formação de conceito, realização de conceito, resolução de problemas e investigação guiada.
- **Instrução interativa:** é fortemente baseada na discussão e compartilhamento entre os participantes. Requer refinamento de competências de observação, escuta, o seu sucesso depende da experiência do professor e desenvolvimento de dinâmicas de grupo. Os métodos instrucionais incluem discussões de classe, em pequenos grupos ou pares de alunos.
- **Aprendizagem experimental:** é indutiva, centrada no aluno, orientada para a atividade e baseia-se na reflexão personalizada sobre uma experiência. Experimentos conduzidos, jogos e simulações são alguns métodos instrucionais.
- **Estudo independente:** incentiva os alunos a serem responsáveis pelo seu planejamento e pelo ritmo de seu próprio aprendizado, sob a orientação ou supervisão de um professor em sala de aula. O aprendizado também pode ser feito em parceria com outro aluno. O conjunto de métodos instrucionais são propositadamente fornecidos para promover o desenvolvimento da iniciativa individual do aluno, autoconfiança e autoaperfeiçoamento.

A organização e o sequenciamento das atividades é outro recurso importante para o processo de ensino e aprendizagem, pois essa é uma forma de facilitar a interpretação, construção e manifestação de conhecimentos e competências para um aluno (BRANCH, 2009). Nesse contexto, Gagné propôs nove eventos de instrução para a realização desse processo (Tabela 4).

Tabela 4. Nove Eventos de Instrução de Gagné.

Evento instrucional	**Ação**
1. Ganhar atenção	Utiliza mudança inesperada de estímulo
2. Informar os objetivos da lição aos alunos	Contar aos alunos o que eles serão capazes de fazer depois do aprendizado
3. Estimular a lembrança da aprendizagem prévia	Lembrar de conhecimentos e competências previamente aprendidas
4. Apresentar o conteúdo	Exibir o conteúdo de diferentes formas
5. Fornecer aprendizagem guiada	Sugerir a organização do conteúdo que faça sentido
6. Obter desempenho	Pedir ao aluno para realizar atividades
7. Fornecer *feedback*	Fornecer *feedback* informativo
8. Avaliar o desempenho	Exigir desempenho adicional do aluno com *feedback*
9. Melhorar a retenção e a transferência de aprendizado	Fornecer práticas e análises variadas

Fonte: DRISCOLL (2000).

A fase de implementação possui o intuito de preparar o ambiente de aprendizagem, treinamento de instrutores e a aplicação da UI na sala de aula (BRANCH, 2009). O professor necessita adquirir conhecimentos adicionais do conteúdo, praticar as estratégias a serem utilizadas e os recursos de ensino criadas nas fases anteriores e se preparar para novos desafios (BRANCH, 2009). O papel do aluno é construir sua própria aprendizagem, dispondo a interagir com os recursos produzidos (BRANCH, 2009). Durante essa fase ocorre a aplicação dos recursos desenvolvidos para a unidade instrucional, deste modo pode-se observar na prática os objetivos de aprendizagem e a utilização dos materiais (BRANCH, 2009).

A fase de avaliação tem o objetivo de examinar a qualidade da unidade instrucional (antes e depois da implementação), identificar os sucessos, e em seguida recomendar melhorias para projetos futuros que sejam semelhantes ao produzido (BRANCH, 2009). Os procedimentos utilizados nessa fase estão associados à determinação dos critérios de avaliação, à seleção das ferramentas adequadas e à realização de avaliações (BRANCH, 2009). Os critérios de avaliação se concentram em medir a capacidade do aluno de adquirir conhecimento, competências e atitudes aprendidas na unidade instrucional (BRANCH, 2009).

Com propósito de avaliar a qualidade de ensino, Kirkpatrick e Kirkpatrick (1998)

criou um modelo de avaliação (Tabela 5):

Tabela 5. Critérios do modelo de avaliação.

Critério	**Característica**
1. Reação do aluno	O que os alunos pensaram e sentiram sobre o treinamento
2. Aprendizagem	Aumento no conhecimento ou capacidade
3. Comportamento	Extensão do comportamento e melhoria de capacidade
4. Resultados	Os efeitos resultantes a partir do desempenho

Fonte: KIRKPATRICK & KIRKPATRICK (1998).

Outra abordagem propõe três níveis de avaliação: percepção, aprendizagem e desempenho. O primeiro mede as percepções dos alunos sobre o conteúdo do curso, os recursos usados e o estilo do professor (BRANCH, 2009). O segundo considera a capacidade do aluno de executar as tarefas indicadas em cada uma das metas e objetivos (BRANCH, 2009). Enquanto o terceiro verifica o conhecimento e a competência do aluno (BRANCH, 2009).

As ferramentas de avaliação são utilizadas para obter dados para influenciar um processo de tomada de decisão (BRANCH, 2009). Há uma variedade de instrumento para a coleta de dados, como pesquisa, questionário, entrevistas, observações, entre outras (BRANCH, 2009). Esses instrumentos podem ser utilizados para analisar a UI em termos de seu sucesso, podendo ser recomendadas melhorias para futuros trabalhos neste escopo. Após a escolha do instrumento apropriado, ocorre a realização da avaliação, usando como base os dados obtidos (BRANCH, 2009). Essa avaliação pode ser realizada por diferentes designs de pesquisa, variando de não experimentais (como, estudo de caso) até experimentais (Tabela 6).

Tabela 6. Tipos comuns de design de pesquisa.

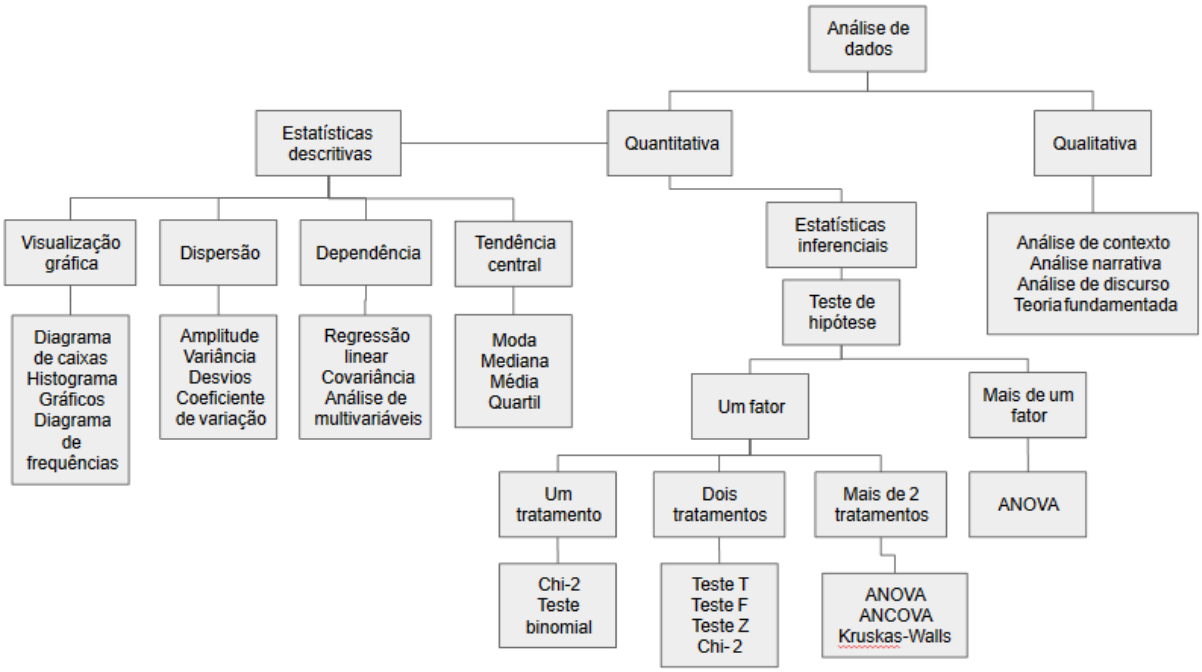
Nível de avaliação (KIRKPATRICK & KIRKPATRICK, 2006)	**Tipo de estudo**	**Design**	**Representação X=Tratamento O=Medição R=Tarefa aleatória**
1 - Reação: Avalia como os participantes se sentiram sobre o treinamento ou a experiência de aprendizado	Estudo de caso	Apenas um pós-teste	X O
2 - Aprendizagem: avalia o aumento de conhecimento ou habilidades	Estudo de caso	Apenas um pós-teste/pré-teste	O X O
	Quase-experimental	Grupo de comparação estático	X O O
		Grupo estático pré-	O X O

		teste/pós-teste	O O
		Série de vezes	O O X O O
	Experimental	Randomizado somente pós-teste	R X O R O
		Randomizado pré-teste/pós-teste	R O X O R O O
		Randomizado com grupo de controle pré-teste/pós-teste	R O X1 O R O X2 O

Fonte: SHADISH, COOK & CAMPBELL (2002) e GRESSE VON WANGENHEIM & SHULL (2009).

A medição deve ser explicitamente definida de uma forma que ligue os objetivos e os dados coletados e também constituam uma maneira genérica para analisar e interpretar os dados (WOHLIN *et al.*, 2012), com o intuito de alcançar os objetivos de avaliação. De acordo com esses objetivos e a natureza dos dados coletados pelas ferramentas de coleta, diferentes métodos de análise qualitativa ou quantitativa podem ser utilizados (Figura 2) (WOHLIN *et al.*, 2012) (FREEDMAN, PISANI & PURVES, 2007). Então, os dados analisados são interpretados para verificar se os objetivos foram atingidos.

Figura 2. Métodos de análise de dados.



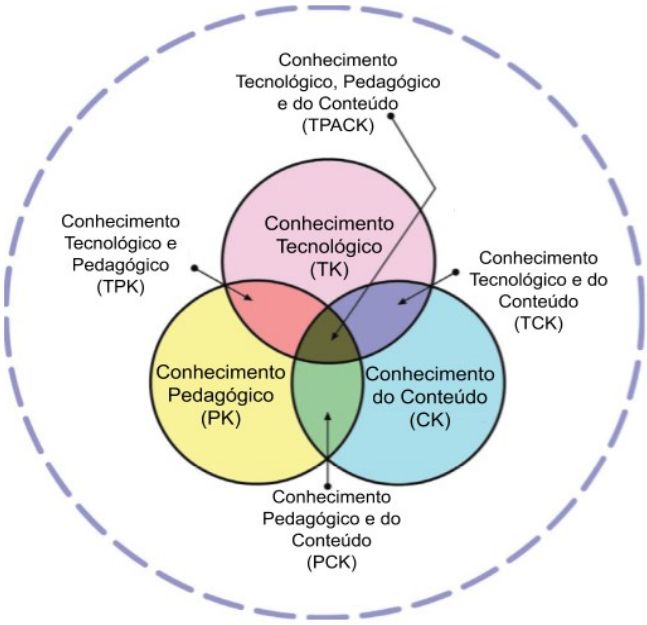
Fonte: WOHLIN *et al.* (2012) e FREEDMAN, PISANI & PURVES (2007).

2.2 FORMAÇÃO CONTINUADA DE PROFESSORES *IN-SERVICE* PARA ENSINO DE COMPUTAÇÃO

O ensino de computação na Educação Básica tem o objetivo de contribuir para a desenvolvimento intelectual dos estudantes, no entanto enfrenta-se muitos desafios, incluindo a formação de professores nessa área (NI *et al.*, 2011). A formação continuada dos professores é entendida como componente essencial da profissionalização considerando os diferentes saberes e as experiências profissionais dos professores (BRASIL, 2016) e também é crucial, para assegurar que os alunos tenham acesso de qualidade aos conceitos básicos de computação (GIANNAKOS *et al.*, 2014). Desta forma, para obter qualidade no ensino, os professores precisam dominar as competências e necessidades em relação aos aspectos centrais de seus conhecimentos (GIANNAKOS *et al.*, 2014). Verloop (2001) relata que as competências do professor estão relacionadas as experiências individuais e contextos nos quais o professor está inserido. Assim, o conhecimento dos professores inclui o conteúdo, suas estratégias de ensino e sua experiência de ensino (VAN DRIEL, BEIJAARD & VERLOOP, 2001).

Os professores precisam estar preparados de forma abrangente para apoiar o ensino de computação. Isso inclui não apenas conhecimento de conteúdo de computação (WONGSOPAWIRO, 2012), mas também componentes de tecnologia e pedagogia (MISHRA & KOEHLER, 2006). Esses componentes do conhecimento formam o *framework* chamado de TPACK (Conhecimento Tecnológico, Pedagógico e do Conteúdo) (MISHRA & KOEHLER, 2006). Esse *framework* se tornou a representação do professor e tem sido usado em muitos estudos recentes com professores (GIANNAKOS *et al.*, 2014). Isso permite guiar a construção do currículo e ajudar na criação de ambientes de aprendizagem conceitualmente coerentes (MISHRA & KOEHLER, 2006). A Figura 3 mostra um Diagrama de Venn representando os três componentes do conhecimento e as interseções desses componentes (GIANNAKOS *et al*., 2014).

Figura 3. Representação do framework TPACK.



Fonte: GIANNAKOS *et al.* (2014).

Inicialmente as bases de conhecimento da formação de professores enfatizava apenas o conhecimento de conteúdo (CK) (SHULMAN, 1986). Então observou-se a necessidade de introduzir o conhecimento da pedagogia (PK), ensinando práticas pedagógicas gerais em sala de aula (MISHRA & KOEHLER, 2006). Depois de fazer os programas de formação de professores utilizarem apenas a pedagogia ou o conteúdo, passaram a possuir um foco no conhecimento da pedagogia e do conteúdo (PCK), pois encontrou-se relação entre os dois (MISHRA; KOEHLER, 2006).

Com a evolução da ciência, novas tecnologias mudaram a natureza da sala de aula ou tem o potencial para fazer isso (MISHRA & KOEHLER, 2006). Em razão disso, o conhecimento tecnológico (TK) está se tornando cada vez mais importante, fazendo os professores possuírem a necessidade de integrar as ferramentas tecnológicas à sua prática docente (GIANNAKOS *et al.*, 2014). A tecnologia utilizada na educação ainda é vista como um conjunto separado de conhecimentos e competências, mas programas de treinamento de professores promovem a aprendizagem de competências específicas de hardware e software como suficientes para serem aplicado como abordagens da educação (MISHRA & KOEHLER, 2006). A Tabela 7 mostra as principais características de cada um dos componentes do conhecimento TPACK.

Tabela 7. Componentes do TPACK e suas características.

Componente	Descrição
Conhecimento do Conteúdo (CK)	Conhecimento e compreensão dos assuntos ensinados, incluindo os conceitos, teorias e procedimentos dentro de uma área; estruturas explicativas que organizam e conectam ideias
Conhecimento Pedagógico (PK)	Conhecimento sobre os processos e práticas ou métodos de ensino e aprendizagem; inclui técnicas para usar na sala de aula; a natureza do público-alvo; e estratégias para avaliar e compreender o aluno
Conhecimento Tecnológico (TK)	Conhecimento sobre tecnologias padrão, tais como livros, giz e quadro negro, e tecnologias mais avançadas, como a internet e vídeo digital; envolve as competências necessárias para operar tecnologias específicas

Fonte: MISHRA & KOEHLER (2006).

Anteriormente foi mencionado o conhecimento pedagógico e do conteúdo, no qual é a interseção de dois componentes do conhecimento. Mishra e Koehler (2006) evidencia que o diferencial dessa abordagem é a especificidade das relações de tecnologia, pedagogia e conteúdo. Os componentes conhecimento pode ser analisados individualmente e por pares (MISHRA & KOEHLER, 2006), demonstrando que a interseção de pares de conhecimento pode resultar em novas características para professores (Tabela 8).

Tabela 8. Interseção dos componentes do TPACK e suas características.

Interseção	Descrição
Conhecimento Pedagógico e do Conteúdo (PCK)	Inclui as abordagens de ensino que encaixam no conteúdo e como os elementos do conteúdo podem ser organizados para um melhor ensino; estratégias de ensino que incorporam representações conceituais, a fim de abordar as dificuldades e promover uma compreensão significativa
Conhecimento Tecnológico e do Conteúdo (TCK)	Conhecimento sobre a maneira na qual tecnologia e conteúdo estão reciprocamente relacionados; precisam saber não apenas o assunto que eles ensinam, mas também a maneira que a matéria pode ser mudada pela aplicação da tecnologia
Conhecimento Tecnológico e Pedagógico (TPK)	Conhecimento da existência, componentes, e capacidades de várias tecnologias; como são usadas em ensino e aprendizagem, compreender como ensinar aos alunos; também a capacidade de escolher uma ferramenta com base em sua adequação e estratégias para usá-la

Fonte: MISHRA & KOEHLER (2006).

A abordagem completa é utilizar a interseção dos três componentes, representando uma forma de conhecimento que é essencial para os professores poderem trabalhar com tecnologia (MISHRA & KOEHLER, 2006). Assim, a qualidade no ensino requer o desenvolvimento de uma compreensão diferenciada, por parte dos

professores, da relação entre tecnologia, conteúdo e pedagogia (MISHRA & KOEHLER, 2006). As principais características do TPACK são: compreensão da representação de conceitos usando tecnologias; técnicas pedagógicas que usam tecnologias de maneira construtiva para ensinar o conteúdo; usar tecnologias para construir o conhecimento existente e desenvolver novas epistemologias ou fortalecer as antigas (MISHRA & KOEHLER, 2006).

Os educadores também precisam entender como construir atividades de aprendizado adequadas, para garantir que os alunos entendam os conceitos sendo ensinados com tecnologia efetivamente integrada nesse processo de aprendizagem (GIANNAKOS *et al*., 2014). Por esse motivo, os professores necessitam possuir algumas características, que auxiliem a elaboração de roteiros para ajudar os alunos a se tornarem aprendizes capacitados (ISTE, 2019). Essas características são chamadas de padrões (Tabela 9), nas quais buscam aprofundar a prática dos professores, promover a colaboração com os colegas, desafiar a repensar as abordagens tradicionais e preparar os alunos para impulsionar seu próprio aprendizado (ISTE, 2019).

Tabela 9. Padrão ISTE para Educadores.

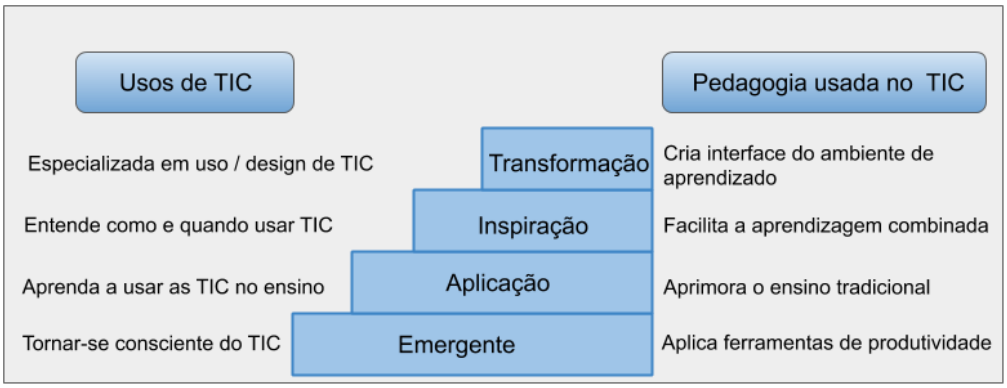
Padrão	**Descrição**
Aprendiz	Aprimoram continuamente sua prática aprendendo com os outros, explorando práticas comprovadas e promissoras que aproveitam tecnologia para melhorar o aprendizado dos alunos
Líder	Procuram oportunidades de liderança para apoiar a participação e o sucesso dos alunos com intuito de melhorar o ensino e a aprendizagem
Cidadão	Participar com responsabilidade no mundo digital e inspiram os alunos a contribuir positivamente
Colaborador	Dedicam tempo para colaborar com colegas e alunos para melhorar a prática, resolver problemas, descobrir e compartilhar recursos e ideias
Inventor	Desenvolvem atividades em ambientes autênticos, motivando experiências de aprendizado que promovam o aprendizado independente
Facilitador	Facilitam o aprendizado com tecnologia para apoiar o desempenho dos alunos
Analista	Entendem e usam dados para orientar suas instruções e ajudar os alunos a atingir suas metas de aprendizado

Fonte: ISTE (2019).

A formação continuada de professores pode visar diferentes níveis de objetivos

de aprendizagem de acordo com modelos contínuos de competência do professor, como por exemplo representado pelo TIC (Tecnologia da Informação e Comunicação) Padrão de Competência para Professores (Figura 4) (GAROFALAKIS, 2015).

Figura 4. As quatro etapas do desenvolvimento de professores em TIC.



Fonte: GAROFALAKIS (2015).

As primeiras ações da formação continuada de professores podem se concentrar na conscientização e compreensão básica da computação, trabalhando também com a percepção dos professores, aumentando sua confiança e motivação para ensinar essa tecnologia em suas classes. Na formação continuada, os próximos passos incluem um treinamento mais abrangente sobre computação, permitindo que os professores apliquem unidades de ensino existentes, bem como conhecimentos sobre como e quando usar computação nas disciplinas de forma eficaz e envolvente. Com o objetivo de transformar as unidades de ensino existentes (por exemplo, através da incorporação de uma forma multidisciplinar), as ações de formação continuada visam um nível mais elevado de competência e abordam aspectos sobre como usar e projetar unidades de ensino para a educação em computação.

A formação continuada de professores *in-service* para o ensino da computação abrange desde programas de treinamento nacionais/estaduais até a aprendizagem por meio de estudo individual (BRASIL, 2016). Existem diversos modelos de formação continuada para professores, as quais variam de altamente adaptados a altamente especializados (KOELLNER & JACOBS, 2014). Além disso, o acesso onipresente as ferramentas de computação levaram a utilização de vários métodos de unidade instrucional, incluindo, presencial, videoconferência, aprendizagem *online* (incluindo MOOC) e instrução combinada (envolvendo uma mistura das aprendizagens presenciais e *online*) (RAVITZ *et al.*, 2017). Abordagens não baseadas

exclusivamente em presenciais facilitam o alcance de um número maior (geograficamente disperso) de professores, além de permitir que eles organizem o treinamento em torno de seu próprio tempo de ensino na escola (SARDESSAI & KAMAT, 2011).

Outra forma é o desenvolvimento de práticas em comunidades (WENGER et al., 2002), nas quais os professores trabalham juntos para um objetivo comum, por exemplo, implementando o ensino de computação, compartilhando suas experiências e ajudando uns aos outros (RAVITZ *et al.*, 2017). Ser orientado é outra maneira que um professor pode desenvolver profissionalmente. O aprendizado pode ocorrer dentro do ambiente escolar, liderado por professores com competências em computação e capazes de apoiar seus colegas no aprendizado da computação e da aplicação em suas aulas. Essas formas também podem ser combinadas, por exemplo, iniciando com a formação continuada e, em seguida, ajudar os professores nas primeiras aplicações por meio de orientação/*coaching* e/ou práticas em grupo.

Além disso, tais programas são desafiados a projetar e oferecer programas de formação com o intuito dos professores não apenas ingressarem em suas salas de aula com um equilíbrio de teoria e prática, no qual lhes permita transmitir o conteúdo necessário aos alunos de maneira completa, mas também que continuem a desenvolver seus conhecimentos e competências em todas as suas carreiras de ensino profissional. A formação continuada eficaz inclui treinamento, prática e *feedback*, e fornece tempo e acompanhamento adequados. Envolve a reflexão sobre a aplicação da computação nas disciplinas, e também promove a participação ativa em comunidades de prática, organizações profissionais ou grupos de interesse (LLOYD & COCHRANE, 2006).

2.3 ENSINO DE COMPUTAÇÃO NA EDUCAÇÃO BÁSICA

Segundo a Constituição Federal de 1996 (BRASIL, 1996):

> A educação, direito de todos e dever do Estado e da família, será promovida e incentivada com a colaboração da sociedade, visando ao pleno desenvolvimento da pessoa, seu preparo para o exercício da cidadania e sua qualificação para o trabalho. (Artigo 2, Lei nº 9.394/96).

O estado possui o dever de garantir a infraestrutura necessária, para que os cidadãos ainda crianças, sejam inseridos na escola (MEC, 2019) seguindo a Base Nacional Comum Curricular (BNCC) que possui a seguinte característica:

> É um documento de caráter normativo que define o conjunto orgânico e progressivo de aprendizagens essenciais que todos os alunos devem desenvolver ao longo das etapas e modalidades da Educação Básica, de modo a que tenham assegurados seus direitos de aprendizagem e desenvolvimento. (MEC, 2019).

A Educação Básica é dividida em três etapas, a Educação Infantil, o Ensino Fundamental e o Ensino Médio (MEC, 2019). A partir, de 4 e 5 anos, é obrigatório as crianças estarem matriculadas na escola, mas podem frequentar a Educação Infantil com idade inferior (MEC, 2019). Estudantes entre 6 e 14 anos frequentam o Ensino Fundamental, o qual tem 9 anos de duração (MEC, 2019). O Ensino Médio é a etapa final da Educação Básica (MEC, 2019).

A computação e as tecnologias digitais de informação e comunicação estão cada vez mais presentes na vida de todos. Isso faz com que além do suporte para elaboração de currículos para as escolas, a BNCC, através das competências gerais da educação, reconhecer que os alunos estão envolvidos com tecnologia (MEC, 2019). Uma das competências da BNCC expressa a importância das tecnologias digitais no dia-a-dia dos estudantes:

> Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva. (MEC, 2019).

Há no Brasil, iniciativas de elaborar um currículo com os assuntos relacionados a tecnologia e computação a serem ensinados aos alunos. A principal, é da Sociedade Brasileira de Computação (SBC), na qual entende que é fundamental e estratégico para o Brasil, que conteúdos de Computação sejam ministrados na Educação Básica (SBC, 2017). Com o intuito de organizar os conteúdos as ser ensinados, a SBC desenvolveu um currículo voltado a Educação Básica, no qual apresenta três eixos

(Tabela 10), que são divididos em pilares (Tabela 11). Esses eixos são direcionados para o ensino de computação, e englobam novas competências cognitivas e novas competências usando Computação para ensinar (SBC, 2017). Em 2019, o eixo de pensamento computacional (Tabela 10) foi incorporado na etapa do Ensino Fundamental da BNCC, bem como no Ensino Médio (BNCC, 2019).

Tabela 10. Eixos de ensino de Computação e seus pilares.

Eixo fundamental	**Descrição**	**Pilares**
Pensamento computacional	Envolve a capacidade de compreender, definir, modelar, comparar, solucionar, automatizar e analisar problemas, usando abstrações e técnicas para encontrar soluções	Abstração Automação Análise
Mundo digital	Codificar informação e organizá-la de forma que possa ser armazenada e recuperada quando necessário, e assim entender qual é a estrutura da Internet	Codificação Processamento Distribuição
Cultura digital	Compreender as relações interdisciplinares da computação com outras áreas do conhecimento, buscando promover o uso de expressão de soluções e manifestações culturais de forma contextualizada e crítica	Computação e Sociedade Fluência Digital Ética Digital

Fonte: SBC (2017) e SBC (2018).

Tabela 11 Pilares Básicos e suas características para o ensino de computação.

Pilar	**Descrição**
Abstração	Compreender, utilizar modelos e representações adequadas para descrever informações, processos e técnicas para construir soluções algorítmicas
Automação	Ser capaz de descrever as soluções por meio de algoritmos de forma que máquinas possam executar partes ou todo algoritmo proposto, bem como de construir modelos computacionais para sistemas complexos
Análise	Analisar criticamente os problemas e soluções para identificar não somente se existem soluções que podem ser automatizadas, mas também ser capaz de avaliar a eficiência e a correção destas soluções
Codificação	Entender como informações podem ser descritas e armazenadas
Processamento	Compreender como a informação é processada por computadores e os diferentes níveis de relação entre hardware e software
Distribuição	Entender como se dá a comunicação entre diferentes dispositivos digitais, como os dados são transmitidos, como é garantida a integridade e segurança no mundo digital, entender a estrutura da Internet
Computação e Sociedade	Compreender o impacto e decorrências da revolução digital e dos avanços do mundo digital na humanidade

Fluência Digital	Utilizar de forma eficiente e crítica ferramentas que auxiliem a obter, analisar, sintetizar e comunicar informações de formatos e fins diversos
Ética Digital	Analisar de forma crítica questões éticas e morais que surgiram com o mundo digital

Fonte: SBC (2017).

Além dos eixos e dos pilares básicos mostrados, a SBC apresenta algumas competências específicas de computação (Tabela 12) que os alunos devem desenvolver durante a Educação Básica (SBC, 2018). Essas competências tem como objetivo complementar a formação adquirida pelos alunos nas outras áreas do conhecimento e também estabelece relação com as competências gerais da Educação Básica, já definidas pela Base Nacional Comum Curricular (BNCC). Dessa forma, a BNCC tem a intenção do estudante ao terminar todas as etapas definidas pelo currículo de referência nacional deve ter a compreensão de fundamentos tecnológicos e aprender a relacionar a teoria com a prática, se apropriando de linguagens das tecnologias digitais e tornando-se fluentes na sua utilização (MEC, 2019). As competências que os alunos devem adquirir são descritas abaixo:

Tabela 12. Competências específicas de computação.

Competência	**Descrição**
Interpretação e transformação do mundo	Aplicar conhecimentos de computação para compreender o mundo e ser um agente ativo e consciente de transformação do mundo digital, capaz de entender e analisar criticamente os impactos sociais, culturais, econômicos, legais e éticos destas transformações
Aplicação de computação em diversas áreas	Compreender a influência dos fundamentos da computação nas diferentes áreas do conhecimento, incluindo o mundo artístico-cultural, sendo capaz de criar e utilizar ferramentas computacionais em diversos contextos, reconhecendo que a computação contribui no desenvolvimento do raciocínio, pensamento computacional, espírito de investigação, criatividade e capacidade de argumentar
Formulação, execução e análise do processo de resolução de problemas	Utilizar conceitos, técnicas e ferramentas computacionais para identificar e analisar problemas cotidianos, sociais e de todas áreas, modelá-los e resolvê-los, individual e/ou cooperativamente, usando representações e linguagens adequadas para descrever processos e informação, validando estratégias e resultados
Desenvolvimento de projetos	Desenvolver e/ou discutir projetos de diversas naturezas envolvendo computação, com base em princípios éticos, democráticos, sustentáveis e solidários, valorizando a diversidade de opiniões de indivíduos e de grupos sociais, sem preconceitos de qualquer natureza
Computação é uma ciência	Compreender os fundamentos da computação e reconhecê-la como uma ciência que contribui para explicar e transformar o mundo, solucionar

	problemas de diversas áreas do conhecimento e para alicerçar descobertas, com impactos no mundo cotidiano e do trabalho

Fonte: SBC (2018).

Os objetos de conhecimento e competências são divididos por ano do Ensino Fundamental e no Ensino Médio, com base nos eixos apresentados anteriormente (SBC, 2018). Nos Anos Iniciais do Ensino Fundamental é importante que os alunos aprendam a solucionar problemas e descrever a solução em forma de algoritmo, já nos Anos Finais devem aperfeiçoar e dominar as principais técnicas para construir soluções algorítmicas, assim como entender o funcionamento da Internet (SBC, 2018). Enquanto no Ensino Médio a ênfase é na aplicação das competências e conhecimentos adquiridos na etapa do Ensino Fundamental (SBC, 2018). Na Tabela 13 estão destacadas as competências referentes ao ensino de computação para o Ensino Fundamental Anos Finais.

Tabela 13. Objetos de conhecimento e competências do Ensino Fundamental Anos Finais.

Ano	**Objeto**	**Competências**
6	Tipos de dados	Reconhecer que entradas e saídas de algoritmos são tipos de dados e formalizar esse conceito, como conjuntos
	Introdução à generalização	Identificar que um algoritmo pode ser uma solução genérica para um conjunto de instâncias de um mesmo problema
	Linguagem visual de programação	Compreender a definição de problema como relação entre entrada e saída, identificando seus tipos e utilizar uma linguagem visual para descrever soluções de problemas
	Técnicas de solução de problemas: decomposição	Identificar problemas de diversas áreas do conhecimento e criar soluções usando decomposição de problemas
	Fundamentos de transmissão de dados	Entender o processo de transmissão de dados
	Proteção de dados	Atribuir propriedade aos dados de uma pessoa, identificar problemas de segurança e sugerir formas de proteger dados
7	Automatização	Compreender que envolve a definição de dados e o processo
	Estruturas de dados: registros e vetores	Formalizar o conceito de registros e vetores
	Técnicas de solução de problemas: decomposição e reuso	Criar soluções para problemas envolvendo a definição de dados usando estruturas estáticas e algoritmos; sua implementação e depuração em uma linguagem

	Programação: decomposição e reuso	Identificar subproblemas comuns em problemas maiores e a possibilidade do reuso de soluções
	Internet	Entender como é a estrutura e funcionamento da internet
	Armazenamento de dados	Compreender e utilizar diferentes formas de armazenamento \de dados
8	Estruturas de dados: listas	Formalizar o conceito de listas dinâmicas, conhecer algoritmos de manipulação e pesquisa sobre listas
	Técnicas de solução de problemas: recursão	Identificar o conceito de recursão em diversas áreas e empregar esse conceito
	Programação: listas e recursão	Identificar problemas de diversas áreas e criar soluções, usando algoritmos sobre listas e recursão
	Paralelismo	Compreender o conceito, identificando partes de uma tarefa que podem ser realizadas simultaneamente
	Fundamentos de sistemas distribuídos	Compreender conceitos de armazenamento e processamento distribuídos; e protocolos para a transmissão
9	Estruturas de dados: grafos e árvores	Formalizar os conceitos de grafo e árvore e conhecer os algoritmos básicos de tratamento
	Técnica de construção de algoritmos: Generalização	Identificar problemas similares e a possibilidade do reuso de soluções, usando a técnica de generalização
	Programação: generalização e grafos	Construir soluções de problemas usando a técnica de generalização, grafos e árvores
	Segurança digital	Compreender vírus, *malware* e técnicas de criptografia

Fonte: SBC (2018).

Com o objetivo de analisar as diretrizes da SBC em relação ao sequenciamento de conteúdo, Alves *et al.* (2019) realizou uma análise em larga escala utilizando a Teoria da Resposta ao Item (ANDRAD, TAVARES & VALLE, 2000) para analisar os conteúdos em relação à complexidade e compara estes resultados com as diretrizes da SBC. Esta análise foi realizada em 2 etapas:

Etapa 1. Avaliação da proficiência em pensamento computacional por meio da análise do projeto App Inventor criado pelo aluno. A avaliação foi realizada de forma automatizada usando a ferramenta CodeMaster 2.0 (ALVES, 2019), por meio da análise com sucesso 88.812 aplicativos públicos disponíveis na Galeria do App Inventor em maio de 2018.

Etapa 2. Análise via Teoria de Resposta ao Item dos resultados da avaliação usando o modelo de Resposta Gradual (SAMEJIMA, 1969). A partir dos parâmetros

calibrados, os itens foram posicionados em uma escala de proficiência (Figura 5) que permite a distinção entre o que é mais fácil e mais difícil do ponto de vista dos aprendizes (ANDRADE, TAVARES & VALLE, 2000).

Figura 5. Posicionamento dos itens na escala.

Pontos da escala			Escala			
5,5	Listas B4					Mais difícil
5,0						
4,5						
4,0						
3,5	Persistência B4					
3,0	Laços B4					
2,5	Laços B2	Laços B3				
2,0	Nomeação B4	Listas B3	Persistência B2	Persistência B3	Condicionais B4	
1,5	Listas B2	Abstração B3	Abstração B4			
1,0	Strings B3	Condicionais B3	Sincronização B2	Abstração B2		
0,5	Operadores B3	Operadores B4	Nomeação B3	Condicionais B2		
0,0	Operadores B2	Variáveis B3	Nomeação B2	Eventos B4		
-0,5	Variáveis B2	Strings B2	Eventos B3			
-1,0						
-1,5	Eventos B2					Mais fácil

Fonte: ALVES et al. (2019).

Os itens posicionados na escala de proficiência são analisados em relação a sua dificuldade. As relações de ordem do posicionamento dos itens são comparadas com o sequenciamento das diretrizes da SBC discutindo similaridades e diferenças na Figura 6.

Figura 6 .Comparação do sequenciamento da SBC com a TRI.

Sequenciamento do eixo Pensamento Computacional das diretrizes da SBC no Ensino Fundamental			Sequenciamento baseado na TRI ← Mais fácil Mais difícil →										
			I01. Eventos	I02. Variáveis	I03. Strings	I04. Operadores	I05. Nomeação	I06. Condicionais	I07. Sincronização	I08. Abstração	I09. Laços	I10. Listas	I11. Persistência
Mais fácil →	1 Ano	Organização de objetos											
		Algoritmo: definição											
	2 Ano	Identificação de padrões de comportamento											
		Modelos de objetos								X			
		Algoritmos: construção e simulação									X		
	3 Ano	Definição de problemas											
		Algoritmo: seleção				X					X		
		Introdução à lógica											
	4 Ano	Estruturas de dados estáticas: registros e vetores										X	
		Algoritmo: repetição									X		
	5 Ano	Estruturas de dados dinâmicas: listas e grafos										X	
		Algoritmos sobre estruturas dinâmicas										X	
	6 Ano	Tipos de dados			X								
		Linguagem visual de programação						X			X		
		Introdução à generalização								X			
		Técnicas de solução de problema: decomposição								X			
	7 Ano	Automatização								X			
		Estruturas de dados: registros e vetores										X	
		Técnicas de solução de problemas: decomposição e reuso								X			
		Programação: decomposição e reuso								X			
← Mais difícil	8 Ano	Estruturas de dados: listas										X	
		Programação: listas e recursão										X	
		Técnicas de solução de problema: recursão											
		Paralelismo							X				
	9 Ano	Estruturas de dados: grafos e árvores											
		Programação: generalização e grafos								X			
		Técnica de construção de algoritmos: Generalização								X			

Fonte: ALVES et al. (2019).

Os resultados indicam que as diretrizes curriculares da SBC possuem um bom alinhamento com a BNCC e abrange uma grande variedade de conteúdo, porém, não aborda todos os conceitos referentes a algoritmos e programação, como o conceito

de eventos. Os resultados também mostram que as diretrizes da SBC não seguem a dificuldade percebida pelo aluno, por exemplo o ensino de conceitos de recursão e grafos já no Ensino Fundamental.

Também há esforços de nível internacional para definir as diretrizes de currículo para o ensino de computação na Educação Básica. Um exemplo chama-se *K-12 Computer Science Standards* da *Computer Science Teachers Association* (CSTA), no qual defende a aplicação da computação como uma ferramenta para aprender e se expressar para a participação ativa dos estudantes em um mundo que é cada vez mais influenciado pela tecnologia (CSTA, 2016).

Com o objetivo de representar áreas específicas de importância disciplinar em vez de ideias abstratas e gerais (CSTA, 2016), foi proposto a divisão do que se deseja ensinar em alguns conceitos fundamentais. Para uma estruturação um pouco mais específica, foram identificados subconceitos usados para criar progressões de aprendizagem aos alunos, nas quais sejam coerentes, abrangendo todas as etapas da Educação Básica (CSTA, 2016). Os conceitos fundamentais são: Sistemas Computacionais (Tabela 14), Redes e Internet (Tabela 15), Dados e Análise (Tabela 16), Algoritmos e Programação (Tabela 17) e Impactos da computação (Tabela 18).

Os sistemas computacionais são recursos nos quais as pessoas estão frequentemente interagindo (CSTA, 2016). Esses sistemas coletam, armazenam e analisam diferentes tipos de informações que podem influenciar a capacidade humana de maneira positiva ou negativa (CSTA, 2016). A compreensão de sistema computacionais e seus subconceitos (Tabela 14) apresentam diversas utilidades, como para solucionar problemas e expandir as competências dos humanos no uso dos dispositivos (CSTA, 2016).

Tabela 14. Subconceitos de Sistemas Computacionais.

Subconceito	**Aprendizado**
Dispositivos	Aplicações em dispositivos de computação comuns e a interação entre humanos e dispositivos
Hardware e software	Compreender a interação entre hardware e software, e usá-los para representar e processar informações
Solução de problemas	Identificar o problema, processos sistemáticos de solução de problemas e como desenvolver seus próprios estratégias de solução de problemas

Fonte: CSTA (2016).

Os sistemas computacionais não funcionam isoladamente, precisam de redes os conectar a outros dispositivos e então compartilhar informações e recursos. As informações compartilhadas nas redes necessitam de segurança e confiabilidade (CSTA, 2016), necessitando aprendizado de subconceitos relacionados à Internet (Tabela 15).

Tabela 15 Subconceitos de Redes e Internet.

Subconceito	**Aprendizado**
Redes de comunicação e organização	A conexão entre pessoas, lugares e coisas e a compreensão de envio e recebimento de informações na rede
Ciber-segurança	Compreender a interação entre hardware e software, e usá-los para representar e processar informações

Fonte: CSTA (2016).

Sistemas computacionais também devem ter a capacidade de processar dados, pois atualmente a rápida expansão da quantidade de dados digitais gerados no mundo é uma realidade (CSTA, 2016). A efetivação de previsões mais precisas utilizando dados coletados e armazenados são importantes para entender melhor o mundo, com isso há necessidade de utilizar os melhores métodos para armazenar uma alta quantidade de dados e compreender como esses dados são analisados (CSTA, 2016).

Tabela 16. Subconceitos de Dados e Análise.

Subconceito	**Aprendizado**
Coleção	A coleta e o uso de dados, efeitos da coletar dados com ferramentas computacionais e automatizadas
Armazenamento	Como armazenar dados e avaliar diferentes métodos de armazenamento
Visualização e transformação	Como usar transformações para simplificar os dados
Inferência e modelos	Uso de dados para fazer previsões simples e modelos e simulações podem ser usados

Fonte: CSTA (2016).

Algoritmo é uma sequência de etapas projetadas para realizar uma tarefa específica e são traduzidos em programas que fornecem instruções para dispositivos de computação (CSTA, 2016). Controlam todos os sistemas de computação e o

aprendizado de seus subconceitos (Tabela 17) capacitam as pessoas a se comunicar com o mundo de novas maneiras e resolvendo problemas (CSTA, 2016).

Tabela 17. Subconceitos de Algoritmos e Programação.

Subconceito	**Aprendizado**
Algoritmos	O desenvolvimento, combinação, decomposição de algoritmos e avaliação de algoritmos concorrentes
Variáveis	Como usar diferentes tipos de dados e formas de organizar grandes conjuntos de dados em estruturas de dados
Controle	Execução sequencial e estruturas de controle
Modularidade	Algoritmos e programas que podem ser projetados ao quebrar tarefas em partes menores e recombinar as soluções existentes
Desenvolvimento de programas	Como e por que as pessoas desenvolvem programas e decisões envolvendo algumas restrições

Fonte: CSTA (2016).

A computação afeta muitos aspectos do mundo, através de comportamentos e interações sociais, influenciando novas práticas culturais (CSTA, 2016). O estudante deve compreender as os impactos da computação (Tabela 18) no futuro da sociedade (CSTA, 2016).

Tabela 18. Subconceitos de Impactos da computação.

Subconceito	**Aprendizado**
Cultura	Compensações associadas à computação e possíveis impactos futuros nas sociedades globais
Interações Sociais	Conexão de pessoas e o impacto em instituições e carreiras de diversos setores
Segurança, lei e ética	Fundamentos da cidadania digital, uso apropriado de mídia digital e questões legais e éticas

Fonte: CSTA (2016).

O currículo apresentado pelo CSTA (2016) também aborda práticas fundamentais, as quais se sobrepõem intencionalmente àquelas em outras disciplinas e usam linguagem semelhante para ajudar os professores a fazer conexões entre ciência da computação e disciplinas. As práticas fundamentais buscam promover uma cultura de computação inclusiva; colaborar com a computação; reconhecer e definir problemas computacionais; desenvolver e usar abstrações; criar artefatos

computacionais; testar e redefinir esses artefatos e comunicar a computação (CSTA, 2016).

A aplicação desses conceitos e práticas fundamentais deve ser realizadas em níveis para ensinar computação na Educação Básica (CSTA, 2016). Com o propósito do aluno evoluir sua aprendizagem, de acordo com os níveis (Figura 7) que já estudou, no decorrer do tempo os assuntos serão mais aprofundados (CSTA, 2016).

Figura 7. Níveis de ensino de computação.

Fonte: adaptado do CSTA (2016).

Com o objetivo de analisar as diretrizes da CSTA em relação ao sequenciamento de conteúdo, foi realizado uma análise em larga escala utilizando a Teoria da Resposta ao Item (ANDRADE, TAVARES & VALLE, 2000) para analisar os conteúdos em relação à complexidade e compara estes resultados com as diretrizes da CSTA (ALVES, 2019). Esta análise foi realizada em 2 etapas (Avaliação da proficiência e Análise via Teoria de Resposta o Item), da mesma forma como foi realizada a análise das diretrizes da SBC. Então, os parâmetros calibrados, os itens foram posicionados em uma escala de proficiência (Figura 8) que permite a distinção entre o que é mais fácil e mais difícil do ponto de vista dos aprendizes (ANDRADE, TAVARES & VALLE, 2000).

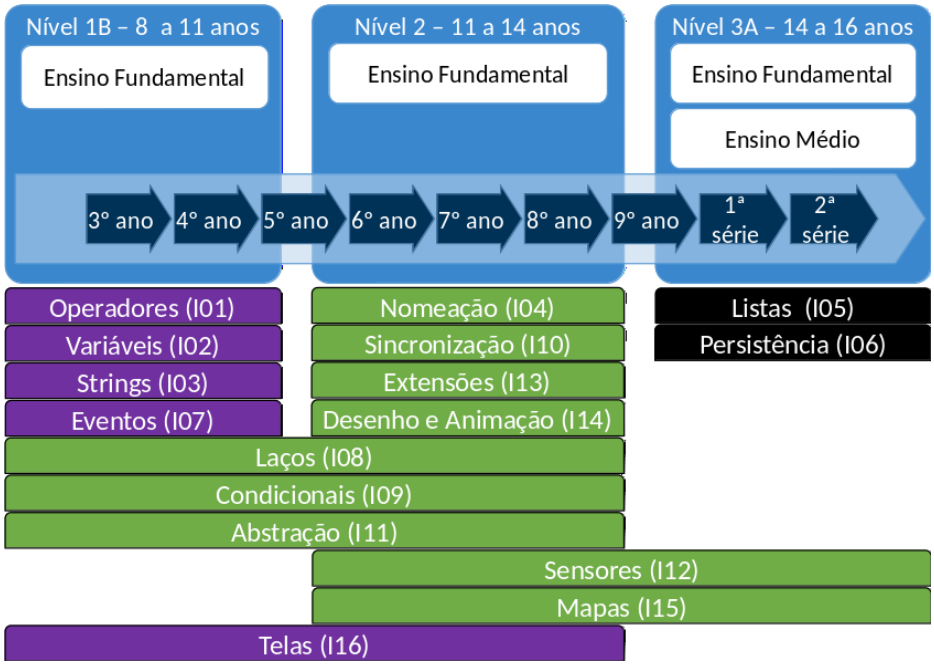
Figura 8. Posicionamento dos itens na escala.

5,5	LISTAS B4						
5,0							MAIS DIFÍCIL
4,5	SENSOR. B4						
4,0							
3,5	PERSIST. B4						
3,0	LAÇOS B4	SENSOR. B3					
2,5	LAÇOS B2	LAÇOS B3					
2,0	NOMEA. B4	LISTAS B3	PERSIST. B2	PERSIST. B3	CONDIC. B4		
1,5	LISTAS B2	ABST. B3	ABST. B4	DES.ANIM. B3	DES. ANIM. B4	TELAS B4	
1,0	STRINGS B3	CONDIC. B3	SINCRO. B2	ABST. B2	SENSOR. B2	DES. ANIM. B2	TELAS B3
0,5	OPERAD.B3	OPERAD.B4	NOMEA. B3	CONDIC. B2			
0,0	OPERAD. B2	VARIÁVEIS B3	NOMEA. B2	EVENTOS B4			
-0,5	VARIÁV. B2	STRINGS B2	EVENTOS B3				
-1,0							
-1,5	EVENT. B2						
-2,0							MAIS FÁCIL
-2,5	TELAS B2						

Fonte: ALVES (2019).

Os itens posicionados na escala de proficiência são analisados em relação a sua dificuldade. As relações de ordem do posicionamento dos itens são comparadas com o sequenciamento das diretrizes da CSTA na Figura 9. Itens roxos são considerados mais fáceis por serem indicados a serem ensinados já do Ensino Fundamental – Anos Iniciais, já os itens pretos são considerados difíceis pois são indicados a serem ensinados no Ensino Médio ou no último ano do Ensino Fundamental. Os itens verdes são considerados de dificuldade mediana pois devem ser ensinados no Ensino Fundamental – Anos Finais.

Figura 9. Dificuldade dos itens de acordo com o *K-12 Computer Science Standards*.



Fonte: ALVES (2019).

Os resultados indicam que as diretrizes curriculares da CSTA possuem um sequenciamento adequado, quanto a dificuldade percebida pelo aluno e abrange grande variedade de conceitos, porém não é alinhado a BNCC. Isso ocorre, pois, as diretrizes do currículo CSTA foram feitas com base na realidade dos Estados Unidos, enquanto que a BNCC foi feita com base na realidade brasileira.

2.4 AMBIENTES DE PROGRAMAÇÃO PARA ENSINAR COMPUTAÇÃO

O conhecimento de conceitos básicos da computação é fundamental para o cidadão, pois podem ser aplicados na resolução de problemas de diferentes áreas do conhecimento, ainda que elas não estejam diretamente relacionadas à computação (CSTA, 2011). Saber resolver problemas, juntamente com a habilidade em algoritmos e programação, proporciona conhecimento para programar vários tipos de software, tais como jogos, animações ou aplicativos móveis (LYE & KOH, 2014).

Alguns ambientes de programação auxiliam no desenvolvimento de software. Ambientes que utilizam linguagens de programação textuais como Java ou C++ têm uma representação que se assemelha a maneira de pensar do computador (SMITH, CYPHER & TESLER, 2000). Porém, as linguagens textuais possuem sintaxes complexas que podem dificultar o aprendizado de pessoas iniciantes (KELLEHER & PAUSCH, 2005). Por outro lado, as linguagens de programação visual baseada em blocos usam representações mais próximas da linguagem humana (LYE & KOH, 2014), proporcionando ao aprendiz se concentrar na lógica e nas estruturas envolvidas na programação, não exigindo a aprendizagem de sintaxe e semântica complexa (KELLEHER & PAUSCH, 2005). As instruções utilizadas (Figura 10) nos programas devem ser representadas por blocos, então o aprendiz deve arrastar e conectar blocos para criar funcionalidades no software que deseja construir (MITb, 2019). Neste sentido, ambientes de programação visual baseados em blocos, como App Inventor (MITa, 2019) e Scratch (MITb, 2019) são geralmente usados com iniciantes.

Figura 10. Exemplo de um programa utilizando blocos.

Fonte: adaptado do CODE.ORG (2019).

Um importante ambiente de programação voltado ao ensino de programação no Ensino Básico é o Code.org. O ambiente Code.org (CODE.ORG, 2019) pode ser utilizado para ensinar conceitos básicos de computação, pois a programação é realizada por meio da conexão de blocos. Este ambiente oferece diversas atividades fechadas para o usuário programar um algoritmo que soluciona algum problema específico. A avaliação do código desenvolvido pelo aluno é automaticamente avaliada e com feedback imediato. O Code.org foi criado e mantido de forma *online* (*https://code.org/*) por uma organização sem fins lucrativos para expandir o ensino de ciência da computação nas escolas. Está disponibilizada gratuitamente em português sob a licença *Creative Commons Attribution-NonCommercial-ShareAlike 4.0 Unported*.

A interface de usuário do Code.org (Figura 11) possui área para visualização da execução do algoritmo, vídeo com explicações e dicas de como resolver o exercício, visualização do status da sequência de exercícios, enunciado e área de programação.

Figura 11. Interface de usuário do Code.org.

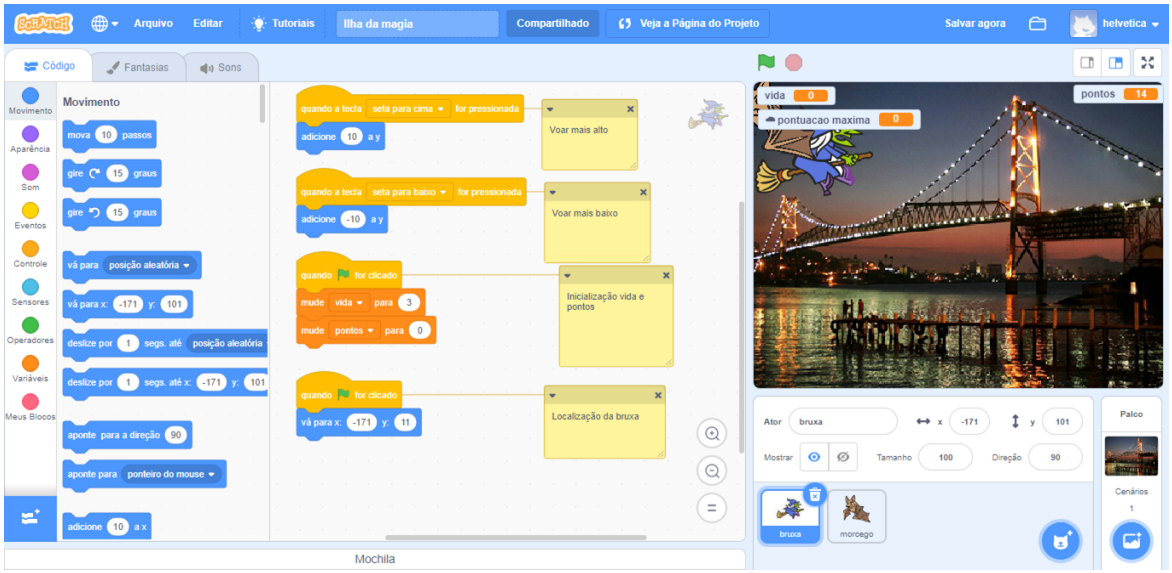
Fonte: CODE.ORG (2019).

O Code.org permite que o professor crie uma conta com possibilidade de criar turmas e monitorar o andamento dos alunos em cada exercício. O Code.org também possui diversos tutoriais *online*, os quais fazem parte de um movimento global chamado Hora do Código, idealizado para mostrar que qualquer pessoa é capaz de aprender programação. Os tutoriais estão disponíveis em mais de 45 idiomas inclusive português brasileiro e são para todas as idades e níveis de experiência.

Outro ambiente utilizado para ensinar programação no Ensino Básico é o Scratch (MITb, 2019). O Scratch (MITa, 2019) também é um ambiente intuitivo de programação visual visando a programação de animações e jogos. Esse ambiente de programação foi criado e mantido de forma *online* (*https://appinventor.mit.edu/*) e *offline* pelo MIT (*Massachusetts Institute of Technology*). Está disponibilizada gratuitamente em português sob a licença *Creative Commons Attribution-ShareAlike 2.0*.

A interface de usuário do Scratch possui uma parte (Figura 12) que engloba a programação do comportamento do jogo, na qual possui 9 tipos de blocos (Movimento, Aparência, Som, Eventos, Controle, Sensores, Operadores, Variáveis, Meus Blocos) e ao lado para projetar a interface do usuário do jogo (MITa, 2019).

Figura 12 Interface de usuário do Scratch.



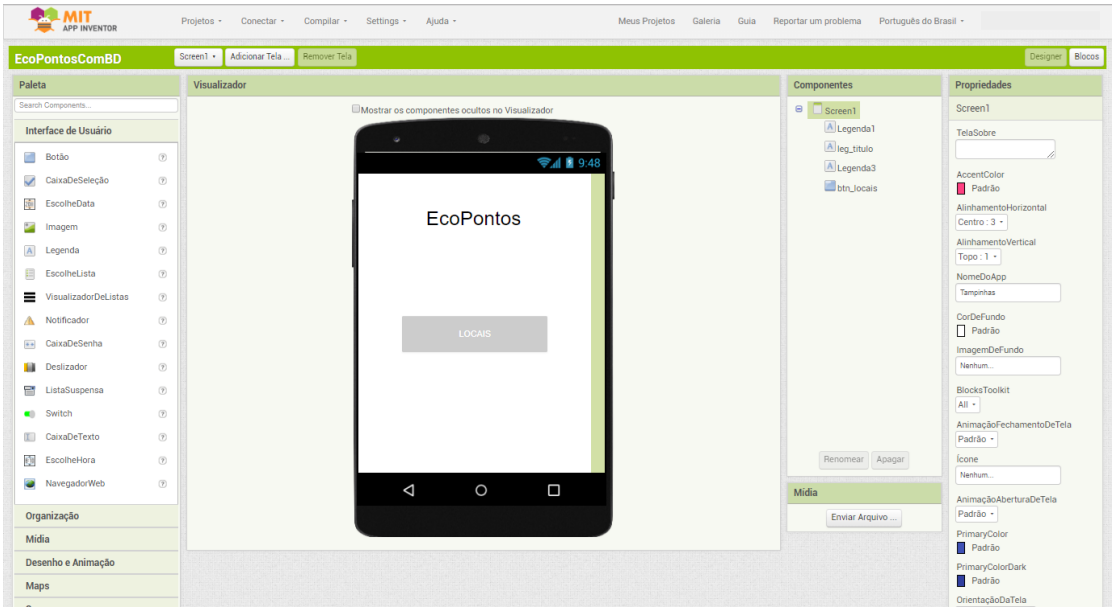
Fonte: MITb (2019).

Uma característica importante do Scratch é a possibilidade de testar a animação ou o jogo durante o seu desenvolvimento, permitindo ajustes durante à construção. Além do mais, o Scratch possui uma área para compartilhar os seus projetos, encontrar outros projetos que podem ser reutilizados, colaborar e aprender com outros desenvolvedores de projetos e criar estúdios para organizar seus projetos e/ou turmas (MITb, 2019). Assim como o Code.org, o Scratch permite que o professor crie uma conta com possibilidade de criar turmas e monitorar o andamento dos alunos no decorrer do desenvolvimento dos jogos.

O MIT App Inventor (MITa, 2019) também é um ambiente intuitivo de programação visual que permite qualquer pessoa criar apps funcionais para *Android*. Esse ambiente de programação foi criado pela *Google* e hoje é mantido de forma *online* (*https://appinventor.mit.edu/*) e *offline* pelo MIT. Está disponibilizada gratuitamente em português sob a licença *Creative Commons Attribution-ShareAlike 3.0*.

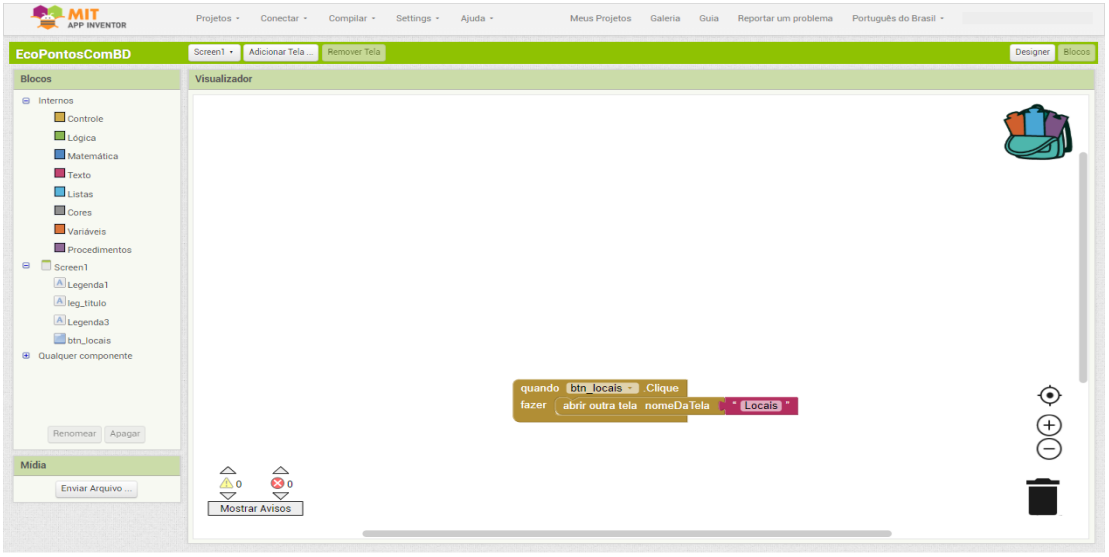
A interface de usuário do App Inventor é baseada em duas partes: *Designer* (Figura 13) para projetar a interface do usuário do aplicativo organizando os componentes na tela e Blocos (Figura 14) para programar o comportamento do aplicativo através da junção de blocos (MITa, 2019).

Figura 13. *Designer* no App Inventor.



Fonte: MITa (2019).

Figura 14. Blocos do App Inventor.



Fonte: MITa (2019).

Uma das principais características do App Inventor é a possibilidade do desenvolvedor poder testar o aplicativo durante a sua construção. Isto permite que o programa seja feito de forma incremental, diminuindo consideravelmente o número de erros produzidos com os testes realizados no mesmo momento (POKRESS & DOMINGUEZ VEIGA, 2013). Ao mesmo tempo que modificações são realizadas nas partes de *Designer* e Blocos do ambiente, as mudanças podem ser vistas diretamente

no *smartphone*, permitindo ao programador ter um *feedback* imediato do seu trabalho (MISSFELDT FILHO, 2019). Deste modo o programador pode testar e encontrar erros no programa enquanto ele está em construção (MISSFELDT FILHO, 2019).

Quando o app estiver pronto, pode ser enviado diretamente ao smartphone conectado ou exportado no formato *apk* para distribuição ou upload na *Google Play Store* (MISSFELDT FILHO, 2019). Caso o usuário deseje, ele pode postar seu aplicativo desenvolvido em uma galeria para que outros usuários possam visualizá-lo ou também utilizá-lo como base para outra aplicação (MISSFELDT FILHO, 2019).

3. ESTADO DA ARTE E PRÁTICA

Com o objetivo de identificar e caracterizar unidades instrucionais existentes, é realizado um estudo do estado da arte e prática da formação de professores da educação básica voltado ao ensino de computação. O estudo é realizado utilizado o processo de mapeamento sistemático proposto por Petersen, Vakkalanka e Kuniarz (2015).

3.1 DEFINIÇÃO DO PROTOCOLO DE MAPEAMENTO

O objetivo deste mapeamento é investigar a questão de pesquisa: Quais unidades instrucionais existem para a formação continuada de professores do Ensino Fundamental Anos Finais de outras áreas a ensinar? Em razão da abrangência dessa questão de pesquisa, foram formuladas as seguintes perguntas de análise:

PA1. Quais UI existem?

PA2. Qual(is) competências de computação e de pedagogia são ensinadas na UI?

PA3. Quais são as características instrucionais da UI?

PA4. Para quais contextos as UI são projetadas / aplicadas?

PA5. Como a UI foi desenvolvida?

PA6. Como a qualidade da UI foi avaliada?

Foi utilizado a base *Scopus* (*www.scopus.com*) para efetuar a busca nas bases das principais editoras científicas. Também foram conduzidas pesquisas no *Google* (*www.google.com*) para minimizar o risco de omitir artigos descrevendo UI não encontrados nas outras fontes.

Com base nas perguntas de análise, foram definidos os critérios de inclusão e exclusão utilizados para a seleção de artigos relevantes. Os critérios de inclusão são:

- Contém descrição de UI para professores;
- Tem como aprendizes professores "in-service" do Ensino Fundamental Anos Finais;
- Focado no ensino de computação - especificamente algoritmos e programação;
- Utiliza linguagem de programação baseada em blocos.

Os critérios de exclusão são:

- Disponíveis apenas em resumos ou apresentações em Powerpoint;

- Não está no contexto de ensino de computação;
- Tem foco no ensino na graduação ou no ensino de computação para alunos da Educação Básica.

A fim de determinar os termos para executar a busca, os conceitos relacionados a questão de pesquisa foram determinados, conforme apresentado na Tabela 19.

Tabela 19. Palavras-chave.

Termos	**Sinônimos**	**Tradução (inglês)**
unidade instrucional	curso	*course, MOOC, training*
professores		*teacher*
ensino fundamental	educação básica	*school*
ensino	aprendizagem	*learning, teaching*
computação	programação, ciência da computação	*programming, computer science*

Fonte: elaborada pela autora.

A partir de buscas de teste foi ajustada a *string* de pesquisa utilizando os termos identificados. Na Tabela 20, é apresentada a *string* utilizada.

Tabela 20. *String* de busca.

(course OR MOOC OR training) AND (teacher) AND (school OR "k-12") AND (learning OR teaching) AND ("computer science" OR programming)

Fonte: elaborada pela autora.

3.2 EXECUÇÃO DA BUSCA

A busca foi realizada em outubro de 2018 pela primeira autora e revisada por outros pesquisadores do GQS/INCoD/INE/UFSC, conforme definido. A busca no *Scopus*, retornou 764 artigos dos quais foram considerados, pela ordem de relevância, os 200 primeiros para a análise. Também foram realizadas pesquisas no *Google*. Dos resultados da busca no *Google*, foram analisadas as 50 primeiras publicações. A partir dos resultados da busca, foi realizada a seleção de artigos relevantes em duas etapas utilizando os critérios de inclusão e exclusão. Na primeira etapa, os artigos potencialmente importantes foram selecionados com base na leitura e análise do título, resumo e palavras-chave. Na segunda etapa foi realizada a seleção com base

na análise do texto completo dos artigos considerados relevantes na etapa anterior.

Tabela 21. Quantidade de artigos por etapa de seleção por repositório.

Fonte de dados	Resultados da pesquisa	Artigos analisados	Seleção depois do 1° estágio	Seleção depois do 2° estágio
Scopus	764	200	54	12
Google	246.000.000	200	5	4

Fonte: elaborada pela autora.

Durante a primeira e segunda etapas da busca, vários artigos foram desconsiderados por relatarem apenas informações sobre a organização de um curso para professores, sem considerar o conteúdo a ser ensinado (CAO & HE, 2009) e outros, por retratar a formação apenas para professores do ensino médio (HAMLEN *et al.*, 2018) (MORELLI *et al.*, 2015). Alguns artigos foram desconsiderados por não apresentarem detalhes da UI (HODHOD *et al.*, 2016) e outros por não se enquadrarem nos critérios de inclusão/exclusão.

Também foi realizada uma análise dos artigos secundários indicados nas referências nos artigos primários encontrados na seleção dos artigos, resultando na identificação de mais um trabalho relevante (CHO *et al.*, 2014).

3.3 ANÁLISE DOS RESULTADOS

Os artigos foram lidos de forma completa e os dados extraídos pelos autores. A extração de dados foi dificultada em alguns casos, pela falta de um protocolo estruturado para as publicações nessa área, resultando em artigos desprovidos de alguns detalhes sobre a UI. Isso impôs aos pesquisadores inferir algumas informações apresentadas no artigo de maneira implícita, incluindo, por exemplo método de análise de dados, idioma da UI e a necessidade de competências prévias de computação. Com base nas perguntas de análise foram extraídas as informações dos artigos conforme apresentado na Tabela 22.

Tabela 22. Informações a serem extraídas referentes ao ensino de computação.

Pergunta de análise	Dados a extrair	Descrição
PA1. Quais UI existem?	Nome	O nome ou o autor da UI
	Referência	Referência bibliográfica

PA2.Qual(is) competências de computação e pedagogia são ensinados na UI?	Objetivo geral de aprendizagem	Identificação do objetivo de aprendizagem da UI em geral
	Estágio alvo de competência	Nível de aprendizagem
	Descrição geral	Breve visão geral da UI, apresentando suas principais características
	Conceitos e práticas de computação	Conceitos e práticas de computação
	Competências pedagógicas	Estratégias instrucionais, atividades, materiais, métodos de avaliação, lições aprendidas para ensinar computação
	Competências tecnológicas	Preparação de infraestrutura (instalação de software)
PA3. Quais são as características instrucionais da UI?	Modo de ensino	Identificação do modo de ensino (distância/online)
	Duração da UI	Duração da UI (número de horas/aulas)
	Ambiente de programação	Linguagem de programação/plataformas usadas na UI
	Método instrucional	Método(s) de instrução utilizado(s) na UI
	Organização das seções	Como os tópicos ensinados na UI foram divididos
	Material instrucional	Material instrucional usado (slides, vídeos, exercícios)
	Métodos de avaliação	Método/instrumento usado para avaliar a aprendizagem do público-alvo
	Inclusão de suporte	Disponibilidade de auxílio após o término da UI
	Língua	Línguas que a UI está disponível
	Licença	Licença de uso
PA4. Quais são as características de contexto da UI?	Área de conhecimento dos professores que participam na UI	Disciplina em que os professores operam (matemática, ciências, história, etc)
	Nível de ensino dos professores que participam na UI	Etapa educativa que os docentes atuam (infantil, fundamental, médio)
	Pré-requisitos	Pré-requisitos dos professores no que diz respeito às competências de computação (iniciante ou avançado)
PA5. Como a UI foi desenvolvida?	Método de desenvolvimento da UI	Identificação do método adotado para o desenvolvimento da UI
PA6. Como a qualidade da UI é avaliada?	Design de pesquisa	Indicação do tipo de estudo adotado para a avaliação da UI
	Fator(es) avaliado(s)	Indicação dos fatores avaliados
	Método(s) de coleta de dados	Indicação do(s) método(s) de coleta de dados adotado(s) para a avaliação da UI
	Tamanho de amostra	Número de pontos de dados utilizados para a avaliação da UI
	Replicação do estudo	Indicação de possíveis replicações da avaliação em vários contextos
	Método(s) de análise de dados	Indicação do(s) método(s) de análise de dados utilizado(s) para a avaliação da UI
	Resultados	Descrição dos principais resultados, pontos fortes e fracos da UI

Fonte: elaborada pela autora.

Em seguida é apresentada a análise dos dados referentes a cada uma das perguntas de análise.

PA1. Quais UI existem?

Ao total foram identificadas 16 unidades instrucionais (Tabela 23) voltadas ao ensino de computação para professores *in-service* da Educação Básica, especificamente do 6º ao 9º do Ensino Fundamental.

Tabela 23. Unidades instrucionais.

Citação	**Referência bibliográfica**
(PARTANEN, MANNILA & PORANEN, 2016)	PARTANEN, T.; MANNILA, L.; PORANEN, T. Learning programming online: a racket-course for elementary school teachers in Finland. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli, Finlândia, 2016, pp. 178-179.
(MARTINEZ *et al.*, 2016)	MARTINEZ, M.; GOMEZ, M. J.; MORESI, M.; BENOTTI, L. Lessons Learned on Computer Science Teachers Professional Development. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, Arequipa, Peru, 2016, pp. 77-82.
(KAY & MOSS, 2012) (KAY *et al.*, 2014)	KAY, J. S., MOSS, J. G.. Using robots to teach programming to K-12 teachers. In: Proceedings of the 2012 Frontiers in Education Conference Proceedings, Seattle, EUA, 2012, pp. 1-6. KAY, J. S.; MOSS, J. G.; ENGELMAN, S.; MACKLIN, T. Sneaking in through the back door: introducing k-12 teachers to robot programming. In: Proceedings of the 45th ACM technical symposium on Computer science education, Atlanta, EUA, 2014, pp. 499-504.
(COOPER *et al.*, 2011) (COOPER *et al.*, 2015)	COOPER, S.; DANN, W.; LEWIS, D.; LAWHEAD, P.; RODGER, S.; SCHEP, M.; STALVEY, R. H. A pre-college professional development program. In: Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, Darmstadt, Alemanha, 2011, pp. 188-192. COOPER, S., RODGER, S. H.; SCHEP, M.; STALVEY, R. H.; DANN, W. Growing a K-12 Community of Practice. In: Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, EUA, 2015, pp. 290-295.
(LIU *et al.*, 2015)	LIU, J.; WILSON, J.; HEMMENWAY, D.; XU, Y.; LIN, C. Oh SNAP! A one-week summer computing workshop for K-12 teachers. In: Proceedings of the 2015 10th International Conference on Computer Science & Education (ICCSE), Cambridge, Reino Unido, 2015, pp. 663-668.
(LIU *et al.*, 2013)	LIU, J.; LIN, C. H.; POTTER, P.; HASSON, E. P.; BARNETT, Z.; XU, Y. Singleton M. Going mobile with app inventor for android: a one-week computing workshop for K-12 teachers. In: Proceedings of the 44th ACM technical symposium on Computer science education, Denver, EUA, 2013, pp. 433-438.
(VIVIAN, FALKNER & FALKNER, 2014)	VIVIAN, R.; FALKNER, K.; FALKNER, N. Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. Research in Learning Technology, 2014.

(HAMNER *et al.*, 2016)	HAMNER, E.; CROSS, J.; ZITO, L.; BERNSTEIN, D.; MUTCH-JOBES, K. Training teachers to integrate engineering into non-technical middle school curriculum. In: Proceedings 2016 IEEE Frontiers in Education Conference (FIE), Erie, EUA, 2016, pp. 1-9.
(HADEN *et al.*, 2016)	HADEN, P.; GASSON, J.; WOOD, K.; PARSONS, D. Can you learn to teach programming in two days? In: Proceedings of the Australasian Computer Science Week Multiconference, Canberra, Austrália, 2016.
(LIU *et al.*, 2014)	LIU, J.; LIN, C. H.; WILSON, J.; HEMMENWAY, D.; HASSON, E. P.; BARNETT, Z.; XU, Y Making games a "snap" with Stencyl: a summer computing workshop for K-12 teachers. In: Proceedings of the 45th ACM technical symposium on Computer science education, 2014, Atlanta, EUA pp. 169-174.
(RUSU, 2015)	RUSU, A. Introducing gaming tools for computing education in STEM related curricula. In: Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE), El Paso, EUA, 2015, pp. 1-8.
(GELDREICH, TALBOT & HUBWIESER, 2018)	GELDREICH, K.; TALBOT, M.; HUBWIESER, P. Off to new shores: preparing primary school teachers for teaching algorithmics and programming. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education, Potsdam, Alemanha, 2018, pp. 1-6.
(VON WANGENHEIM *et al.*, 2015)	VON WANGENHEIM, A.; GRESSE VON WANGENHEIM, C.; PACHECO, F. S.; HAUCK, J. C. R.; FERREIRA, M. N. F. Motivating Teachers to Teach Computing in Middle School – A Case Study of a Physical Computing Taster Workshop for K-12 Teachers. International Journal of Computer Science Education in Schools, vol. 35, n. 1, pp. 368-373, 2017.
(ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009)	ALIMISIS, D.; FRANGOU, S.; PAPANIKOLAOU K. A Constructivist Methodology for Teacher Training in Educational Robotics. In: the 9th IEEE International Conference on Advanced Learning Technologies, Riga, Letônia, 2009, pp. 25-28.
(CHO *et al.*,2014)	CHO, S.; PAUCA, V. P.; JOHNSON, D.; JAMES Y. V. Computational Thinking for the Rest of Us: A Liberal Arts Approach to Engaging Middle and High School Teachers with Computer Science Students. In: Proceedings of the Society for Information Technology & Teacher Education International Conference, 2014, Jacksonville, EUA, pp. 79-86.
(LIU *et al.*, 2011)	LIU, J.; LIN, C. H.; HASSON, E. P.; BARNETT, Z.; XU, Y. Introducing computer science to K-12 through a summer computing workshop for teachers. In: Proceedings of the 42nd ACM technical symposium on Computer science education, Dallas, EUA, 2011, pp. 389-394.

Fonte: elaborada pela autora.

Um número significativo de UI encontradas foi publicado nos últimos 10 anos, indicando um aumento de importância dessa área. Observando o crescimento de iniciativas para ensinar computação nas escolas, verifica-se a necessidade de professores qualificados para essa função. Um dos principais motivos para o aumento das UI existentes entre 2014 e 2016 (apresentado na Figura 15) possivelmente tem

relação com a inclusão de programação e computação como um elemento integrado em todas as disciplinas, como está acontecendo na Finlândia (PARTANEN, MANNILA & PORANEN, 2016), Austrália e Inglaterra (VIVIAN, FALKNER & FALKNER, 2014).

Figura 15. Quantidade de UI com foco no ensino de computação para professores por ano de publicação.

Fonte: elaborada pela autora.

PA2. Qual(is) competências de computação e pedagogia são ensinadas na UI?

Dentro do foco da pesquisa, todas as unidades instrucionais encontradas visam ensinar práticas de programação para que os participantes possam aplicar computação em suas salas de aula. Algumas buscam promover a habilidade e confiança (KAY & MOSS, 2012; HAMNER *et al.*, 2016), enquanto outra tem o intuito de explicar técnicas pedagógicas referentes ao ensino de programação (HADEN *et al.*, 2016). Vivian, Falkner e Falkner (2014) também menciona a importância dos participantes entenderem os impactos culturais da tecnologia.

Com o intuito de alcançar os objetivos propostos dos trabalhos encontrados, as competências a serem adquiridas pelos participantes foram classificados em conceitos (e práticas) de computação, competências pedagógicas e tecnológicas. Os conceitos de computação ensinados por várias UI focam em algoritmos e programação (condicionais, loops e variáveis), e outras incluem desenvolvimento e uso de abstrações. Há exceções que ensinam também colaboração na computação (MARTINEZ *et al.*, 2016), sistemas de computador (VIVIAN, FALKNER & FALKNER, 2014), criação (LIU *et al.*, 2014) e testes (LIU *et al.*, 2011) de artefatos computacionais, definição de problemas computacionais (HADEN *et al.*, 2016) e análise de dados (LIU

et al., 2013).

As competências pedagógicas instruídas geralmente são vinculadas com o conhecimento pedagógico. O conhecimento pedagógico é ensinado, em 12 UI, por meio do desenvolvimento de materiais curriculares (LIU *et al.*, 2011), de metáforas e analogias adequadas para os conceitos (RUSU, 2015), explicação de estratégias instrucionais (VON WANGENHEIM *et al.*, 2017) e demonstração de práticas educativas específicas (HADEN *et al*., 2016). Os requisitos curriculares também foram considerados por Partanen, Mannila e Poranen (2016).

As competências tecnológicas ensinadas são relacionadas com o conhecimento tecnológico. Porém, observa-se que 7 UI não relatam a inclusão desse tipo de competência. Nas demais UI, o conhecimento tecnológico é coberto por meio de instruções de instalação de *software* (LIU *et al.*, 2011; VON WANGENHEIM *et al.*, 2017), montagem de robôs LEGO (ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009) e simulação da execução dos programas em um emulador de *Android* (LIU *et al.*, 2013).

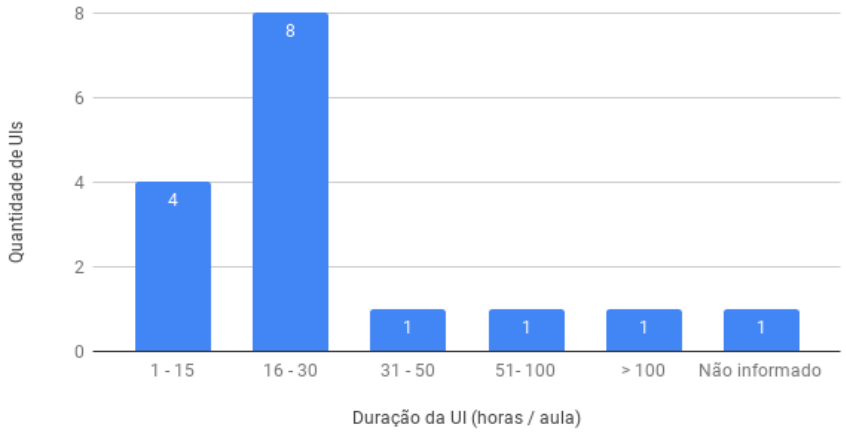
A fim de atingir os objetivos definidos pelas unidades instrucionais, algumas apresentam aspectos que foram encontrados em apenas uma ou em poucas UI selecionadas. As características distintas encontradas são: apoio após o término da UI (COOPER *et al.*, 2015), aprendizado com emulador *Android* (LIU *et al.*, 2013), utilização de exemplos com jogos (LIU *et al.*, 2014; RUSU, 2015) e ensino de instruções de instalação do *software* (LIU *et al.*, 2011; VON WANGENHEIM *et al.*, 2017). Algumas UI envolveram os participantes em atividades práticas na maior parte do tempo (CHO *et al.*, 2014). Enquanto isso, algumas focaram em cursos de curta duração, com aulas nos finais de semana e/ou feriados (HADEN *et al*., 2016). Duas UI são projetadas para o modo de ensino *online* (PARTANEN, MANNILA & PARANEN, 2016; VIVIAN, FALKNER & FALKNER, 2014) utilizando múltiplas linguagens e/ou plataformas (MARTINEZ *et al.*, 2016; VON WANGENHEIM *et al.*, 2017).

Com todos esses conceitos e competências ensinados aos participantes das unidades instrucionais, a maioria das UI visa atingir o nível de aprendizagem de aplicação, conforme o modelo (Figura 4). Somente uma UI enfoca no nível emergente (Figura 4).

PA3. Quais são as características instrucionais da UI?

A maioria das UI identificadas utilizam o modo de ensino presencial (14 UI) com somente duas UI realizadas de forma *online*. A duração das UI encontradas varia de 1 a 30 horas/aula (Figura 16). Apenas três UI tem uma duração maior do que 30 horas/aula.

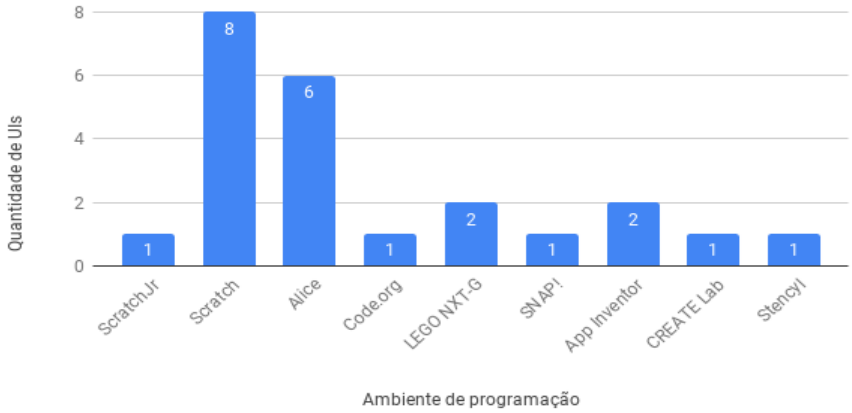
Figura 16. Tempo de duração da UI em horas / aula.



Fonte: elaborada pela autora.

O ensino de conceitos de computação é tipicamente voltado a algoritmos e programação de *software*. Deste modo, as UI aderem a ambientes de programação para aplicar os conceitos e proporcionar experiências práticas. Um modo intuitivo é utilizar ambientes de programação visual baseados em blocos, como Scratch e Alice (desenvolvimento de *software desktop*), os mais utilizados nas UI encontradas (Figura 17). ScratchJr, Code.org, SNAP!, App Inventor e Stencyl também são ambientes de programação baseados em blocos. LEGO NXT-G e CREATE *Lab Visual Programmer* também são baseados em blocos utilizados na programação de robótica. De forma mais pontual podem ser também utilizadas linguagens de programação baseadas em textos (*como Racket e Python) e outras ferramentas (como Chatbot e UNC++Duino).*

Figura 17. Ambientes de programação.



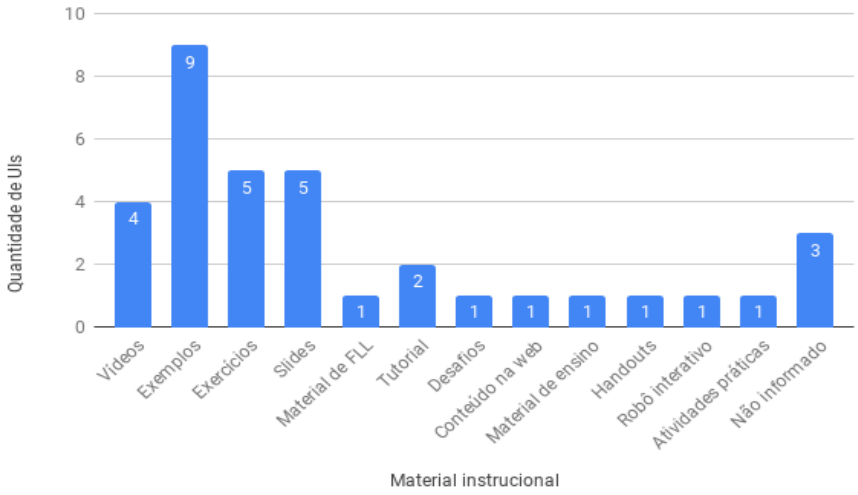
Fonte: elaborada pela autora.

As unidades instrucionais encontradas geralmente apresentam uma organização bem definida dos tópicos a serem ensinados. Algumas UI abordam metodologias que utilizam experiência prática por meio de tutoriais, com tutores para tirar dúvidas (PARTANEN, MANNILA & PORANEN, 2016; COOPER *et al.*, 2015; VON WANGENHEIM *et al.*, 2017). Outras intercalam entre aulas expositivas e práticas, geralmente em pares, como intuito de introduzir conceitos de computação e depois utilizar o conhecimento aprendido para resolver exercícios e promover discussões. Esses exercícios podem variar de atividades práticas utilizando um ambiente de programação até o desenvolvimento de materiais e currículos para serem aplicados na sala de aula. Em algumas UI, os participantes também devem apresentar os artefatos criados (HAMNER *et al.*, 2016; VIVIAN, FALKNER & FALKNER, 2014; COOPER *et al.*, 2015; PARTANEN, MANNILA & PORANEN, 2016).

Nas UI encontradas, são utilizados diversos tipos de materiais instrucionais (Figura 18). Os materiais mais utilizados são exemplos que contenham programas completos (HADEN *et al.*, 2016) planos de aula (COOPER *et al.*, 2015) e jogos (LIU *et al.*, 2014). Exercícios e slides também são muitas vezes empregados em aplicações que intercalam aulas expositivas e práticas (VON WANGENHEIM *et al.*, 2017). Vídeos são apresentados para motivar os participantes e introduzir conceitos de computação (PARTANEN et al., 2016), principalmente quando o curso é *online* (VIVIAN, FALKNER & FALKNER, 2014). Tutoriais guiando passo-a-passo à programação usando determinados ambientes são adicionados nas aulas práticas (COOPER *et al.*, 2015; VON WANGENHEIM *et al.*, 2017). A variedade dos materiais instrucionais utilizados

é apresentada na Figura 18).

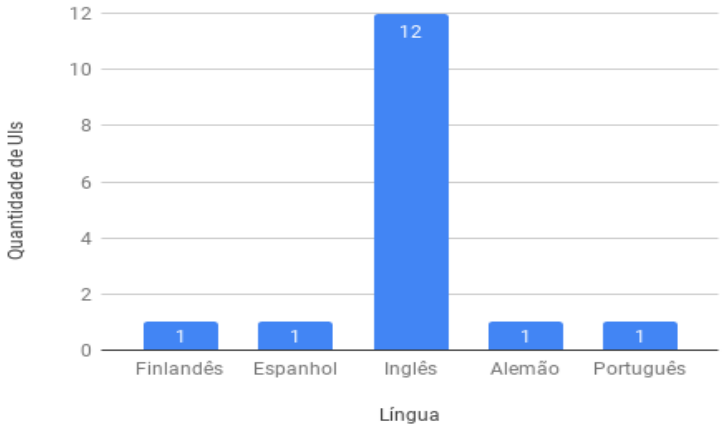
Figura 18. Material instrucional.



Fonte: elaborada pela autora.

A licença para o uso desses materiais foi informada em apenas três publicações: *Koodiaapinen MOOC* (PARTANEN, MANNILA & PORANEN, 2016), *Creative Commons* (VON WANGENHEIM *et al.*, 2017) e uma indica que o uso é grátis (COOPER *et al.*, 2015). A língua usada nos materiais e o curso como todo, foram inferidas de acordo com as informações dos países e pesquisadores que aplicaram a UI. Encontram-se: finlandês, espanhol, alemão, português e em grande parte inglês (Figura 19).

Figura 19. Idioma utilizado no desenvolvimento dos materiais.



Fonte: elaborada pela autora.

A aprendizagem e o conhecimento que os participantes estão adquirindo durante a aplicação são importantes para o sucesso da unidade instrucional.

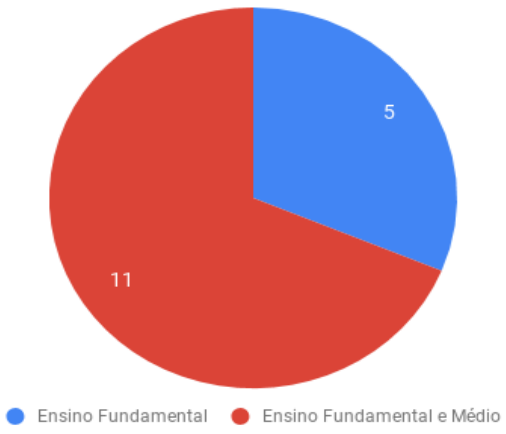
Referente à avaliação da aprendizagem nas UI, vários artigos indicam a estratégia utilizada, embora cinco publicações não informaram o método e uma não avaliou. As outras onze UI recorreram a autoavaliação por meio de questionários (antes e depois da aplicação) e exercícios de programação, revisão por pares de artefatos de programação e desempenho em lições sobre aspectos pedagógicos (PARTANEN, MANNILA & PORANEN, 2016) Apresentações de artefatos criados e *feedback* do tutor mediante observações também foram utilizados.

Oferecer suporte após o término da UI pode ser uma maneira de passar confiança aos participantes e fazer com que utilizem, em suas escolas, o conteúdo que aprenderam. No entanto, onze publicações não informam se disponibilizam esse tipo de suporte e uma proporciona na forma *online*, mas apenas durante o curso (ALIMISIS, FRANGOU & PAPANOKOLAOU, 2009). Outras quatro oferecem visitas durante o ano seguinte (COOPER *et al.*, 2015), suporte nos semestres seguintes (LIU *et al.*, 2011), possibilidade de procurar apoio (GELDREICH, TALBOT & HUBWIESER, 2018) ou trocas de experiências por meio da Comunidade *Google+* (VIVIAN, FALKNER & FALKNER, 2014).

PA4. Quais são as características de contexto da UI?

A maioria das UI encontradas tem o foco na Educação Básica como um todo e não um nível específico, envolvendo como participantes professores de diferentes níveis escolares (Figura 20).

Figura 20. Nível escolar no qual os professores participantes lecionam.



Fonte: elaborada pela autora.

Observando a atual falta de professores formados especificamente na área de

computação e a falta da inclusão de conteúdo de computação no currículo base em muitos países (VON WANGENHEIM *et al.*, 2017), uma solução proposta por todas as UI analisadas é uma abordagem interdisciplinar, proporcionando aos professores fazerem a integração entre o ensino de computação e outras disciplinas consideradas tradicionais, como matemática, história e geografia. Algumas unidades instrucionais também buscam abranger os professores que fornecem apoio de TI nas escolas.

As competências prévias dos participantes não são informadas em várias publicações, porém de forma geral, professores *in-service* de outras áreas não têm um conhecimento prévio da computação. Uma das unidades (VON WANGENHEIM *et al.*, 2017) somente cita a necessidade de saber usar um computador e outra enfatiza a disposição dos professores em ensinar computação (MARTINEZ *et al.*, 2016).

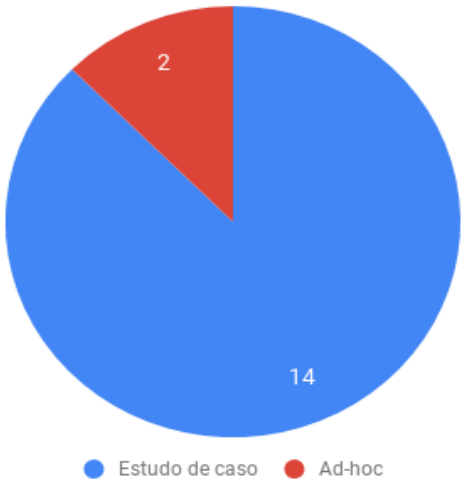
PA5. Como a UI foi desenvolvida?

No desenvolvimento das UI observou-se a escassez de informações sobre o processo utilizado, encontrando cinco publicações que nem abordam essa questão (MARTINEZ *et al.*, 2016; KAY & MOSS, 2012; COOPER *et al.*, 2015; LIU *et al.*, 2015; RUSU, 2015). Encontramos informações da metodologia de pesquisa utilizada somente referente a três unidades instrucionais (VIVIAN, FALKNER & FALKNER, 2014; VON WANGENHEIM *et al.*, 2017; GELDREICH, TALBOT & HUBWIESER, 2018). A UI de Vivian, Falkner e Falkner (2014) foi desenvolvida em três etapas (revisão de computação, desenvolvimento e design, implementação e revisão), iniciando com uma revisão quase-sistemática de trabalhos de pesquisa sobre computação na educação. Outras duas se baseiam no *design* instrucional (VON WANGENHEIM *et al.*, 2017; GELDREICH, TALBOT & HUBWIESER, 2018), sendo que um trabalho seguiu o modelo ADDIE (VON WANGENHEIM *et al.*, 2017). As UI restantes fundamentam-se em modelos de treinamento já existentes na literatura (HAMNER *et al.*, 2016), trabalhos realizados por universidades (LIU et al., 2014), experiências anteriores (PARTANEN, MANNILA & PORANEN, 2016), exemplos encontrados em livros (HADEN *et al*., 2016) ou uma ideia introduzida por Resnick (ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009).

PA6. Como a qualidade da UI é avaliada?

A avaliação é uma etapa importante no processo de desenvolvimento de uma unidade instrucional e busca avaliar se os objetivos de aprendizagem são atingidos e identificar oportunidades de melhoria. Nos trabalhos encontrados, quatorze UI foram avaliadas por meio de estudo de caso (Figura 21). Na maioria dos estudos, esses dados são obtidos por meio de questionários antes e/ou após o ensino. Duas UI foram avaliadas de forma *ad-hoc*, menos rigorosa, permitindo maior flexibilidade no processo e avaliação da UI (VIVIAN, FALKNER & FALKNER, 2014) (COOPER *et al.*, 2015).

Figura 21. Tipos de estudo.

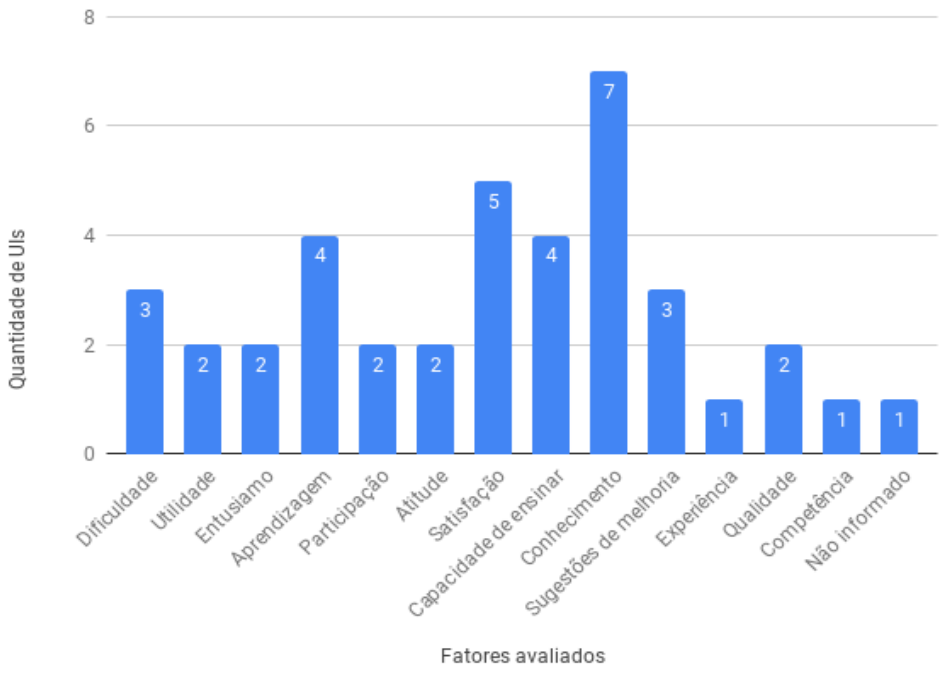


Fonte: elaborada pela autora.

As UI observadas, predominam a avaliação de variados fatores (Figura 22). O conhecimento em computação e a satisfação dos participantes são os fatores mais avaliados, demonstrando a importância do professor sentir que o esforço está trazendo benefícios para seu conhecimento e possuir a compreensão do conteúdo. Alguns estudos também buscam avaliar a aprendizagem (MARTINEZ *et al.*, 2016; COOPER *et al.*, 2015; VON WANGENHEIM *et al.*, 2017; ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009) e a capacidade do professor utilizar o que aprendeu em suas aulas (KAY & MOSS, 2012, HADEN *et al.*, 2016; RUSU, 2015; ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009). Outros fatores avaliados são o nível de dificuldade da UI (PARTANEN, MANNILA & PORANEM, 2016; LIU *et al.*, 2015; VON WANGENHEIM *et al.*, 2017), a utilidade dos conteúdos apresentados (PARTANEN, MANNILA & PORANEM, 2016; CHO *et al.*, 2014), o entusiasmo dos participantes (PARTANEN,

MANNILA & PORANEM, 2016; VON WANGENHEIM *et al.*, 2017), sua participação nas aulas (MARTINEZ *et al.*, 2016; VIVIAN, FALKNER & FALKNER, 2014), a atitude (KAY & MOSS, 2012; RUSU, 2015), experiência (VIVIAN, FALKNER & FALKNER, 2014) e competência dos professores (LIU *et al.*, 2011), além da qualidade da unidade desenvolvida (RUSU, 2015) (ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009).

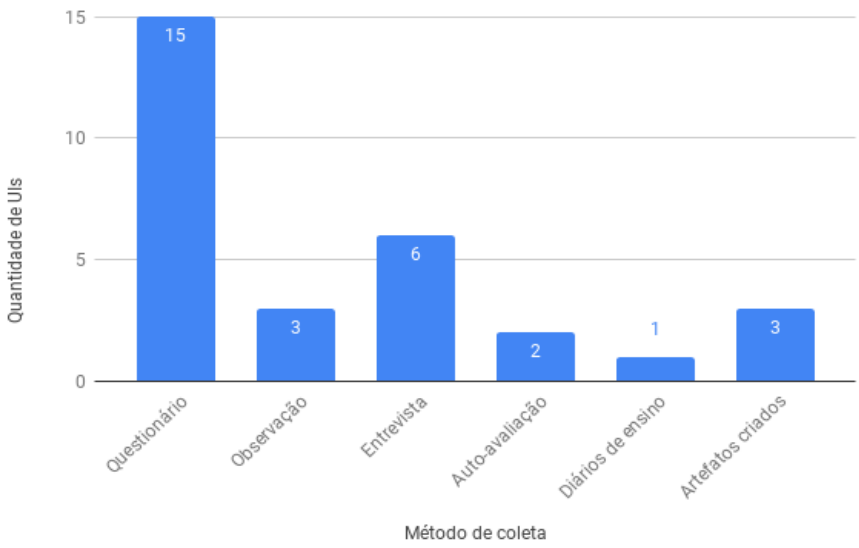
Figura 22. Fatores avaliados.



Fonte: elaborada pela autora.

Os questionários são a principal forma para efetuar a coleta de dados, mas outros métodos também são utilizados (Figura 23). Em algumas UI, os participantes fornecem um *feedback* durante uma entrevista com os pesquisadores (LIU *et al.*, 2015) ou a um avaliador externo (LIU *et al.*, 2011). Também foi aplicada avaliação por pares baseada na criação de um recurso de ensino e/ou um plano de aula (VIVIAN, FALKNER & FALKNER, 2014) e observações do comportamento dos professores em sala de aula (MARTINEZ *et al.*, 2016), com o propósito de extrair conclusões sobre a qualidade da UI, a partir do desempenho dos professores nessas avaliações. Em outros trabalhos, foram usados autoavaliação do grau de conhecimento (LIU *et al.*, 2013) e diário de ensino (GELDREICH, TALBOT & HUBWIESER, 2018).

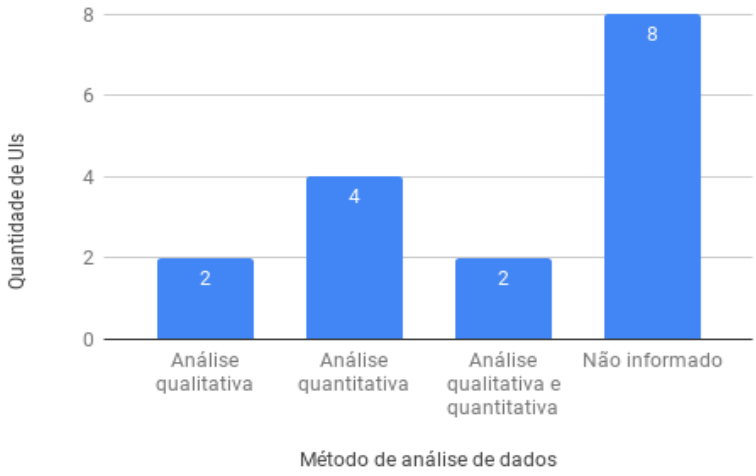
Figura 23. Método de coleta de dados.



Fonte: elaborada pela autora.

Os métodos adotados para a análise dos dados não foram mencionados na metade das publicações (Figura 24).A análise quantitativa foi utilizada por quatro UI, adotando testes de hipótese e/ou estatística descritiva (KAY & MOSS, 2012; COOPER et al., 2015; LIU et al., 2013; CHO et al., 2014). A análise qualitativa também foi utilizada para a avaliação de duas unidades instrucionais (RUSU, 2015; MARTINEZ *et al.*, 2016). Outras duas avaliações mencionaram o uso de análise qualitativa e quantitativa (VON WANGENHEIM *et al.*, 2017; GELDREICH, TALBOT & HUBWIESER, 2018).

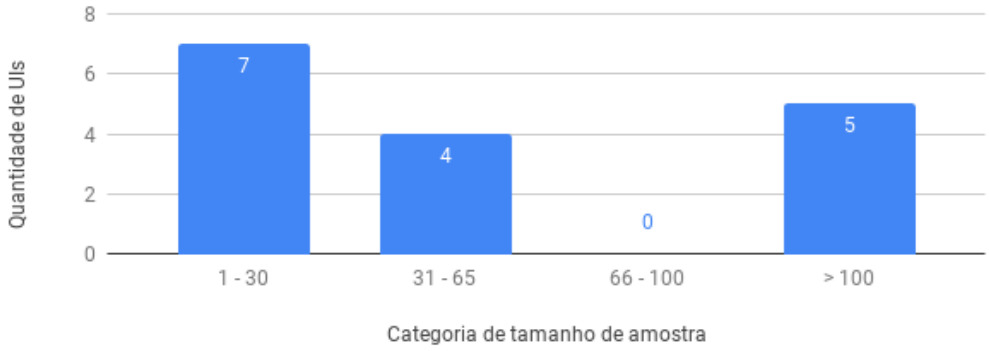
Figura 24. Método de análise de dados.



Fonte: elaborada pela autora.

As amostras utilizadas pelas pesquisas em geral variam de 1 a 65 participantes, possivelmente correspondendo a uma turma de aplicação da UI (Figura 25). Entretanto, cinco publicações apresentam um tamanho maior que 100 professores (PARTANEN, MANNILA & PORANEN, 2016; MARTINEZ *et al.*, 2016; COOPER *et al.*, 2015; VIVIAN, FALKNER & FALKNER, 2014; HAMNER *et al.*, 2016).

Figura 25. Tamanho da amostra.



Fonte: elaborada pela autora.

As unidades instrucionais selecionadas não têm seus estudos replicados em mais de um contexto na metade dos casos (8 UI). Enquanto em seis UI os estudos foram replicados em mais do que uma turma/contexto (CHO *et al.*, 2014; RUSU, 2015; HADEN *et al.*, 2016; HAMNER *et al.*, 2016; KAY & MOSS, 2012; MARTINEZ *et al.*, 2016; PARTANEN, MANNILA & PORANEN, 2016). Em duas publicações essa informação da quantidade de replicações não foi encontrada (ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009) (LIU *et al.*, 2011).

O *feedback* das unidades instrucionais, em geral, foi positivo, indicando que o nível de dificuldade e a carga de trabalho estavam razoáveis (PARTANEN, MANNILA & PORANEN, 2016). O conhecimento em programação foi aprimorado (KAY & MOSS, 2012; COOPER *et al.*, 2015; LIU *et al.*, 2015; LIU *et al.*, 2013; LIU *et al.*, 2014; LIU *et al.*, 2011), com o desenvolvimento da capacidade de transferir o treinamento para suas próprias salas de aula (HADEN *et al.*, 2016; RUSU, 2015). No entanto, observou-se que essas UI foram insuficientes para preparar professores sem experiência anterior (MARTINEZ *et al.*, 2016), os quais se sentem desconfortáveis para responder perguntas sobre computação aos alunos (GELDREICH, TALBOT & HUBWIESER, 2018). Os resultados também destacam que, para permitir que o professor aplique oficinas de forma eficaz, cursos de treinamento mais longos e suporte contínuo são

necessários (VON WANGENHEIM *et al.*, 2017). Alguns participantes avaliaram a computação como benéfica para o ensino (HAMNER *et al.*, 2016), no entanto, muitos professores não conseguiram completar o curso (VIVIAN, FALKNER & FALKNER, 2014). Outro ponto observado é que aprender a pedagogia é tão importante quanto aprender conceitos de computação, porque simplesmente ensinar algoritmos e definições pode não promover uma aprendizagem significativa da disciplina (MARTINEZ *et al.*, 2016).

3.4 DISCUSSÃO

Em geral, o número de unidades instrucionais encontradas, visando a formação de professores *in-service* referentes ao ensino de competências de computação para o Ensino Fundamental Anos Finais, foi muito pequeno (apenas 16 no total). Isso mostra a escassez de trabalhos nessa área e a necessidade de desenvolvimento de mais UI para esse fim.

Em termos de conceitos e práticas de computação ensinados nas UI, observou-se que a maioria foca em conteúdos relacionados a algoritmos e programação. As publicações também destacaram o ensino de competências pedagógicas aos professores, enquanto competências tecnológicas foram abordadas somente em poucas UI.

As unidades instrucionais geralmente possuem uma organização dos tópicos bem definida, dividindo o conteúdo em seções. Vários métodos são utilizados para ensinar esse conteúdo, mas a maioria emprega abordagens de aprendizagem ativa. Tipicamente ocorre a introdução de um conceito, demonstrações de exemplos e em seguida os participantes desenvolvem algum artefato. Em outras UI, os professores realizam a aula com base em tutoriais e/ou resolução de problemas. O conteúdo é explicado aos professores utilizando diferentes materiais instrucionais, como vídeos, slides, tutoriais e exemplos, proporcionando caminhos de aprendizagem flexíveis, a mistura de conteúdo e exemplos (VIVIAN, FALKNER & FALKNER, 2014). A aprendizagem e o desempenho dos professores nessas aulas são medidos por meio de questionários, observações e artefatos criados.

Os diversos ambientes e plataformas de programação usados também

chamam a atenção, abrangendo desde robótica utilizando o *LEGO NXT-G* (ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009) até aplicativos, como App Inventor (LIU *et al.*, 2013) e sistemas para *desktop*, como Scratch (GELDREICH, TALBOT & HUBWIESER, 2018). Algumas UI utilizam múltiplas linguagens/plataformas, pois acreditam que ensinar programação não é sobre o uso de uma plataforma ou o aprendizado de uma linguagem de programação específica, mas sobre como aprender conceitos de computação (MARTINEZ *et al.*, 2016).

De forma geral, observa-se uma falta de informações sobre a base teórica e o método de desenvolvimento das UI. Apenas três UI apresentam informações sobre a metodologia utilizada para desenvolvê-las sistematicamente. Enquanto as outras não apresentam nenhuma informação sobre a base teórica empregada no desenvolvimento. Em razão da falta de rigor científico, a maioria das UI foi avaliada somente por estudos de casos com pequenas amostras e/ou de forma *ad-hoc*.

3.4.1 Ameaças à validade

Com a intenção de minimizar os impactos de ameaças à validade desse estudo de mapeamento, foram tomadas várias ações descritas em Petersen, Vakkalanka e Kuniarz (2015). Mas o mapeamento pode apresentar desvios, como o viés comum de que os resultados positivos têm maior probabilidade de serem publicados do que os negativos (KITCHENHAM, 2004). Ainda assim, é considerado que esse fator tem apenas uma pequena influência, pois a pesquisa buscou as características de uma unidade instrucional para ensinar computação a professores.

Uma ameaça é a omissão de estudos relevantes. Com o intuito de minimizar esse risco foram realizadas pesquisas em várias bases de dados, utilizando também sinônimos na *string* de busca. Para reduzir esse risco ainda mais foram realizadas também buscas informais no *Google*, além da análise das referências dos artigos encontrados.

A ameaça à seleção de publicações foi reduzida por meio da definição e documentação de um protocolo para a escolha criteriosa de artigos, assim como pela definição detalhada de critérios de inclusão/exclusão. Mesmo encontrando dificuldades na extração de informações das publicações relevantes pela falta de um

protocolo de relato nessa área, o processo foi realizado de forma cuidadosa em conjunto com outros pesquisadores do GQS/INCoD/INE/UFSC até que um consenso foi obtido.

4. DESIGN DA UNIDADE INSTRUCIONAL “APRENDA A ENSINAR ALGORITMOS E PROGRAMAÇÃO”

A unidade instrucional é voltada a ensinar conceitos básicos da computação principalmente com enfoque em algoritmos e programação por meio de atividades de programação com Code.org (CODE.ORG, 2019), desenvolvimento de jogos com Scratch e desenvolvimento de apps com App Inventor. A UI também aborda conhecimento tecnológico, pedagógico e do conteúdo, conforme sugerido por Mishra e Koehler (2006). A unidade instrucional é alinhada ao *framework* da CSTA (2016) e as diretrizes do currículo da SBC (2018) e busca habilitar professores *in-service* a ensinar algoritmos e programação com o Code.org e as unidades instrucionais “UNIfICA - UNidade Instrucional Interdisciplinar de Computação e HistóriA” (ALVES, 2016) e “Faça o seu app” (MISSFELDT FILHO, 2019; PINHEIRO, 2019; FERREIRA, 2019; ALVES, 2019; JUSTEN, 2019). A UNIfICA é voltada ao ensino de práticas de computação/programação e ao pensamento computacional para criar jogos e reforçar os conhecimentos dos alunos na disciplina de história. E a UI “Faça o seu app” é voltada ao ensino de competências de algoritmos e programação e de engenharia de *software* (ES), design *thinking* e *User eXperience* (UX) a alunos do Ensino Fundamental.

Em conformidade com o design instrucional (BRANCH, 2009), a unidade instrucional é definida com base na definição e análise do contexto em termos do público-alvo (professores *in-service* do Ensino Fundamental Anos Finais), incluindo os alunos da Educação Básica (especificamente Ensino Fundamental Anos Finais), os quais são indiretamente atingidos pela UI, e características das escolas. Os resultados dessa análise de contexto são apresentados na seção 4.1. Também são definidos os objetivos de aprendizagem e plano de ensino da UI nas seções 4.1.4 e 4.2.2, respectivamente.

4.1 ANÁLISE DE CONTEXTO

4.1.1 Análise do perfil dos professores

O público-alvo da unidade instrucional a ser projetada consiste em professores *in-service* do Ensino Fundamental Anos Finais. A maioria dos professores possui idade entre 35 e 54 anos, e de aproximadamente 11 à 30 anos de carreira (TODOS PELA EDUCAÇÃO, 2018). O nível de escolaridade dos educadores é um aspecto importante na educação, pesquisas mostram que, dos professores atuantes na Educação Básica, 77,5% possuem nível superior completo (INEP, 2016). Desses docentes com graduação, 90% têm curso de licenciatura (INEP, 2016) em áreas como matemática e história.

Atualmente é evidente a discrepância (Figura 26) entre a procura por cursos de licenciatura considerados “tradicionais” (como matemática, história e português) e no curso de licenciatura em computação (INEP, 2015). O número de matrículas em cursos de formação de professor de computação (informática) representa apenas 0,8% do total, muito menor do que de outras matérias, como por exemplo matemática, que possui 5,8% do total (Figura 26) (INEP, 2015). Quanto a evasão, verifica-se um número maior nos cursos de computação em comparação com a média nacional (HOED, 2016). Vale ressaltar que estes dados se referem apenas aos cursos de licenciatura, não incluindo outros cursos de bacharelado em Ciência da Computação e Engenharia da Computação (INEP, 2015). Com o cenário de inferior número de matrículas, associado uma das maiores taxas de evasão, pode-se concluir que há poucos professores de computação na educação básica, em comparação com outras disciplinas. Então, para inserir computação na Educação Básica em grande escala, em geral será ministrado por professores que não foram formados na área de computação.

Figura 26. Comparação entre o número de estudantes matriculados nos cursos de licenciatura.

Matrícula de graduação em Licenciatura por curso – Brasil 2015

N	Curso/Nome OCDE	Matrículas [1]	Percentual (%)	Percentual Acumulado (%)
1	Pedagogia	648.998	44,3	44,3
2	Formação de professor de educação física	149.011	10,2	54,4
3	Formação de professor de biologia	88.294	6,0	60,4
4	Formação de professor de história	86.661	5,9	66,3
5	Formação de professor de matemática	84.522	5,8	72,1
6	Formação de professor de língua/literatura vernácula (português)	80.737	5,5	77,6
7	Formação de professor de geografia	50.723	3,5	81,1
8	Formação de professor de língua/literatura estrangeira moderna	48.383	3,3	84,4
9	Formação de professor de língua/literatura vernácula e língua estrangeira moderr	39.081	2,7	87,0
10	Formação de professor de química	35.892	2,5	89,5
11	Formação de professor de física	25.102	1,7	91,2
12	Formação de professor de filosofia	20.046	1,4	92,6
13	Formação de professor de artes visuais	17.609	1,2	93,8
14	Formação de professor de sociologia	15.220	1,0	94,8
15	Formação de professor de música	14.855	1,0	95,8
16	Formação de professor de ciências	13.183	0,9	96,7
17	Formação de professor de computação (informática)	12.210	0,8	97,5
18	Formação de professor de artes (educação artística)	6.692	0,5	98,0
19	Formação de professor de teatro (artes cênicas)	4.898	0,3	98,3
20	Formação de professor das séries finais do ensino fundamental	3.359	0,2	98,6

Fonte: Mec/Inep; Tabela elaborada por Inep/Deed
Nota: (1) Não constam dados de cursos de Área Básica de Ingresso

Fonte: INEP (2015).

A maioria dos professores afirmam utilizam programas e aplicativos pedagógicos durante suas aulas (Figura 27), mostrando que alguns professores sabem utilizar celulares e computadores, mas geralmente não possuem conhecimento aprofundado em computação.

Figura 27. Uso de programas e aplicativos para fins pedagógicos.

Fonte: INEP (2017).

Alguns educadores são contratados por tempo determinado, num período de mais ou menos um ano (TODOS PELA EDUCAÇÃO, 2018). Mas, em geral, os professores são contratados por tempo indeterminado, no caso de escolas públicas a maior parte é concursado (TODOS PELA EDUCAÇÃO, 2018). Apesar disso, 37% dos

educadores trabalham em duas ou mais escolas (TODOS PELA EDUCAÇÃO, 2018), enfrentando diversos ambientes de trabalho, e ainda tendo que se locomover de uma escola para outra. Na rede particular de ensino esse número sobe para 55% (TODOS PELA EDUCAÇÃO, 2018).

4.1.2 Análise do perfil dos alunos - Ensino Fundamental Anos Finais

Nos anos iniciais do ensino fundamental, segundo a BNCC, os alunos desenvolvem experiências com a leitura e a escrita, muitas vezes já iniciada na família e na Educação Básica (MEC, 2019). É nessa etapa também, que os alunos aprendem a resolver problemas, a compreender e interpretar o mundo, entre outras assuntos para que desenvolvam o raciocínio (MEC, 2018). Sendo assim, se espera que no 1º ou 2º ano do Ensino Fundamental as crianças já sejam alfabetizadas (MEC, 2019). Depois, quando adolescentes, frequentam o Ensino Fundamental Anos Finais. E, espera-se um domínio dos assuntos aprendidos anteriormente, normas da língua portuguesa e saibam compreender alguns aspectos da língua inglesa (MEC, 2019). Atualmente o pensamento computacional faz parte da Base Nacional Comum Curricular.

A maioria os alunos do Ensino Fundamental Anos Finais já realizaram alguma atividade na internet, como pesquisar curiosidades, assistir vídeos, usar redes sociais, mapas e mandar mensagens (CETIC, 2017). A internet é usada mais de uma vez por dia por 88% dos alunos (CETIC, 2017). Isso mostra que os alunos dessa faixa etária (11 - 14 anos) sabem usar dispositivos eletrônicos, como celular, tablet e computadores.

Os adolescentes nasceram na era dos dispositivos eletrônicos, não conhecem o mundo sem a tecnologia, fazem várias tarefas ao mesmo tempo e não encontram nenhuma dificuldade em apreender a lidar com as novidades tecnológicas, estas são características da geração Z (KAMPF, 2011; TOLEDO, 2012). Esta geração atualmente presente nas escolas, é caracterizado por estudantes, os quais são impacientes e querem que as situações sejam resolvidas de imediato (TOLEDO, 2012). Estão ligados no mundo virtual, por isso demonstram resistência ao modelo de educação atual (TOLEDO, 2012), exigindo que as escolas e os professores se adaptem a novas práticas, para serem estimulados a estudarem, não querendo

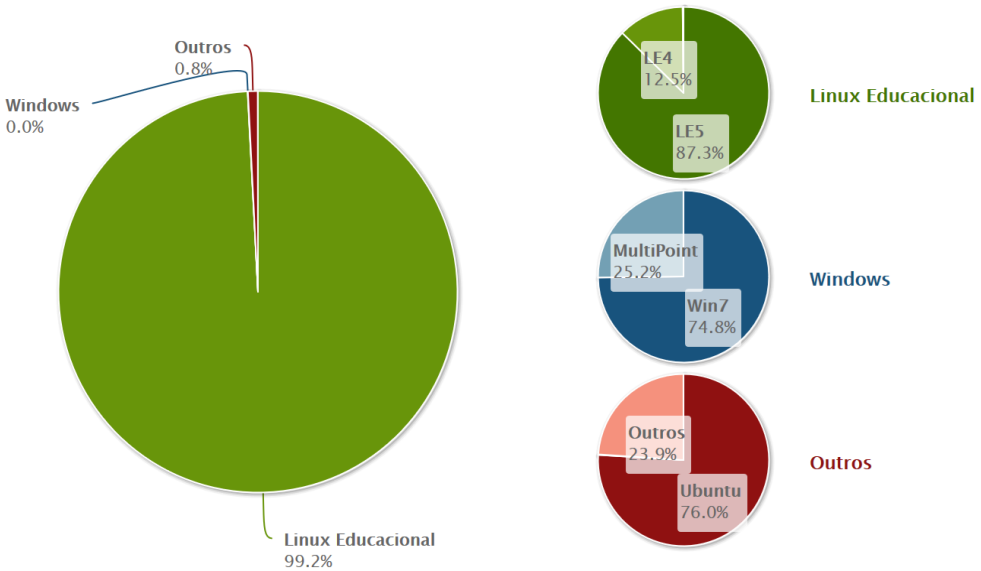
apreender com a forma antiga, de palavras e quadro negro, mas com tecnologia (KAMPF, 2011; TOLEDO, 2012).

4.1.3 Análise das características das escolas

Ao ensinar computação nas escolas é necessário que as mesmas tenham uma infraestrutura adequada, principalmente em relação a recursos básicos para que os alunos tenham acesso ao computador (ou outro dispositivo) e a internet. Alguns diretores, coordenadores pedagógicos e professores afirmam que o número insuficiente de computadores por aluno, equipamentos ultrapassados (antigos), baixa velocidade da internet, ausência de suporte técnico e manutenção são alguns dos fatores que dificultam o uso de TI nas escolas (CETIC, 2017).

A opinião sobre a infraestrutura das escolas pelos profissionais que estão diariamente no ambiente, mostram a dificuldade das escolas em manter laboratórios de informática. Isso se reflete nos números, sendo um recurso disponível em 38% (INEP, 2018), no qual para aplicar alguns conceitos computacionais esperados pela SBC (2018) e pela CSTA (2016) é um recurso indispensável. O sistema operacional desses computadores, em sua maioria, é o Linux Educacional (Figura 28), tendo como minoria o uso de *windows*, *ubuntu*, *debian*, e outros sistemas (PROINFO/MEC, 2018).

Figura 28. Sistema operacional dos computadores da escola pública.



Fonte: PROINFO/MEC (2018).

Um outro indicativo agravante, citado pelos professores como fator que dificulta o uso de TI - se refere a baixa velocidade de conexão à internet nas escolas (Figura 29), dificultando a utilização dos computadores para atividades *online.*

Figura 29. Velocidade da principal conexão à internet da escola.

(%)	Total de escolas		Escolas públicas		Escolas particulares	
	2015	2016	2015	2016	2015	2016
Até 999 Kbps	5	5	7	6	1	0
De 1 a 2 Mbps	32	25	38	27	23	20
De 3 a 4 Mbps	11	12	12	12	9	11
De 5 a 10 Mbps	21	22	16	19	28	31
11 Mbps ou mais	10	11	4	6	24	28
Não sabe/ Não respondeu	21	24	24	30	15	11

Fonte: CETIC (2016).

Apesar da existência de leis em alguns estados que proíbem o uso de celular no ambiente da escola, é fácil identificar seu uso nas escolas por estudantes, professores e funcionários (MARTIN et al., 2014). No estado de Santa Catarina possui Lei nº 14.363, de 25 de janeiro de 2008, proíbe o uso de telefone celular nas salas de aula das escolas públicas e privadas de todo o estado (RODRIGUES; TOSCHI, 2018). Enquanto que na esfera federal o Projeto de Lei nº 2.806, de 2011 foi rejeitado em 2014 (RODRIGUES; TOSCHI, 2018). Portanto, no âmbito federal é permitido utilizar celular no ambiente escolar, mas em Santa Catarina e em outros estados e/ou municípios que possuem leis proibindo o uso, não é permitido.

4.1.4 Objetivos de aprendizagem

O objetivo geral da UI é formar professores *in-service* para o ensino de computação nos Anos Finais do Ensino Fundamental. Esses professores, em geral, têm conhecimento de conteúdo da área de estudo que ensinam (por exemplo, história, artes, ciências, etc.), conhecimento pedagógico geral, bem como conhecimentos e habilidades profissionais gerais, mas normalmente não possuem competências em

computação (GRESSE VON WANGENHEIM *et al.*, 2017). Então, torna-se necessário ensinar conhecimento de conteúdo e pedagogia de computação, bem como aspectos tecnológicos.

Os objetivos de aprendizagem definidos para a UI são apresentados na Tabela 24, contendo o nome, a área de conhecimento e a fonte utilizada como base para a criação do objetivo. Cada objetivo foi derivado de metas de aprendizagem presentes em currículos das áreas, tendo como principal fonte o guia do currículo CSTA (2016) referente ao nível de Ensino Fundamental nos Anos Finais, já que não foram encontradas diretrizes curriculares para formação de professores *in-service*. Os objetivos relacionados ao conhecimento pedagógico e tecnológico foram derivados de características definidas pelo *framework* descrito por Mishra e Koehler (2006).

Tabela 24. Objetivos de aprendizagem da UI.

ID	**Objetivo de aprendizagem**	**Área de conhecimento**	**Fonte**
Conhecimento do conteúdo			
OA1	Compreender algoritmos como um conjunto de instruções passo-a-passo para realizar tarefas	Algoritmo e Programação	(CSTA, 2016: 1B-AP-08)
OA2	Conhecer diferentes ambientes de programação	Algoritmo e Programação	(FERREIRA, 2019)
OA3	Criar interface de aplicativo usando App Inventor	Algoritmo e Programação	(DA CRUZ, 2019) (PINHEIRO, 2019)
OA4	Criar programas que usam condicionais e laços	Algoritmo e Programação	(CSTA, 2016: 1B-AP-10; 2-AP-12)
OA5	Criar programas que usam eventos, variáveis, operadores e sensores	Algoritmo e Programação	(CSTA, 2016: 2-AP-11) (BRENNAN; RESNICK, 2012) (SHERMAN *et al.*, 2014) (SHERMAN; MARTIN, 2015)
OA6	Criar programas que usam aparência e som	Algoritmo e Programação	(BRENNAN; RESNICK, 2012)
OA7	Criar programas que usam movimento	Algoritmo e Programação	(BRENNAN; RESNICK, 2012)
OA8	Criar programas que usam nomeação, *strings*, mapa, abstração, bibliotecas externas (API), listas e persistência de dados	Algoritmo e Programação	(CSTA, 2016: 2-AP-11; 1B-AP-10; 2- AP-14; 3A-AP-14; 2-AP-16) (SHERMAN *et al.*,

			2014; SHERMAN; MARTIN, 2015)
OA9	Conhecer, compartilhar e usar aplicativos e jogos do App Inventor e Scratch	Algoritmo e Programação	(CSTA, 2016: 1B-AP-12, 1B-AP-17, 2-AP-16)
OA10	Identificar e resolver problemas criando sistemas de software interativos	Algoritmo e programação / Engenharia de Usabilidade	(CSTA, 2016: 1B-CS-03, 3A-AP-13) (AIGA, 2008)
OA11	Explicar o conceito de um ciclo de vida de software e forneça um exemplo, ilustrando suas fases, incluindo as entregas que são produzidas	Engenharia de *Software*/ Engenharia de Usabilidade	(ACM/IEEE, 2013) (UXQB, 2018)
OA12	Analisar o contexto de sistemas de software interativo em termos de usuários, tarefas, dispositivos e ambientes de uso	Engenharia de Usabilidade	(AIGA, 2008) (ISO 9241-220, 2019) (CSTA, 2016: 1B-IC-19) (ACM/IEEE, 2013)
OA13	Especificar requisitos de sistemas de software interativos em termos de funcionalidade e usabilidade	Engenharia de *Software* / Engenharia de Usabilidade	(CSTA, 2016: 2-AP-19)(ACM/IEEE, 2013)
OA14	Criar protótipos de sistemas de software interativos em diferentes níveis (esboços, baixa fidelidade, alta fidelidade, funcional)	Engenharia de Usabilidade	(CSTA, 2016: 2-AP-19, 1B-AP-13, 2-AP-13) (ACM/IEEE, 2013)
OA15	Desenvolver artefatos computacionais iterativamente de forma colaborativa, seguindo um cronograma	Engenharia de Software	(CSTA, 2016: 2-AP-18)
OA16	Modelar processos criando e seguindo algoritmos/mapas de navegação para concluir tarefas	Algoritmo e Programação	(CSTA, 2016: 1A-AP-08)
OA17	Usar fluxogramas, pseudocódigo e/ou mapas de navegação para resolver problemas complexos	Algoritmo e Programação	(CSTA, 2016: 2-AP-10)
OA18	Projetar o design visual (cores, tipografia, ícones, imagens.) do sistema de software interativo	Engenharia de Usabilidade	(ACM/IEEE, 2013) (CSTA, 2016: 2-IC-21) (GARRET, 2011)
OA19	Procurar e incorporar o *feedback* dos membros da equipe e dos usuários para refinar uma solução que atenda às necessidades do usuário	Algoritmo e Programação /Engenharia de *Software* /Engenharia de Usabilidade	(CSTA, 2016: 2-AP-15)
OA20	Testar e refinar um sistema de software interativo para funcionalidade e usabilidade	Engenharia de *Software*	(CSTA, 2016: 2-AP-17, 1B-AP-15, 3A-AP-21)

OA21	Recomendar melhorias no design de dispositivos de computação, com base nos resultados de verificação e validação	Engenharia de *Software*	(CSTA, 2016: 2-CS-01)
Conhecimento pedagógico			
OA22	Conhecer e entender guias de currículo para o ensino de computação na Educação Básica		(MISHRA; KOEHLER, 2006)
OA23	Conhecer métodos de ensino e aprendizagem de computação e como ele se aplica em sala de aula		(MISHRA; KOEHLER, 2006)
OA24	Compreender estratégias para avaliar a compreensão do aluno sobre computação		(MISHRA; KOEHLER, 2006)
OA25	Identificar como o ensino de computação pode ser inserido no ensino da Educação Básica (extracurriculares, oficinas, interdisciplinar)		(MISHRA; KOEHLER, 2006)
Conhecimento tecnológico			
OA26	Compreender habilidades para tecnologias específicas e como lidar com novas tecnologias		(MISHRA; KOEHLER, 2006)
OA27	Revisar/criar a infraestrutura técnica necessária para realizar aulas de implementação de atividades fechadas de programação, jogos e aplicativos		(MISHRA; KOEHLER, 2006)
OA28	Compreender questões práticas de tecnologia no ensino de computação (problemas que podem acontecer e como agir caso correr)		(MISHRA; KOEHLER, 2006)

Fonte: elaborada pela autora.

4.2 PROJETO DA UNIDADE INSTRUCIONAL

O curso foi projetado para ser aplicado na forma presencial em aproximadamente 40 horas/aula, com as aulas a ser realizadas por um instrutor que possui conhecimento no assunto apresentando o conteúdo. Um ambiente de aprendizagem (*moodle*) pode ser utilizado como suporte. Visa-se que o instrutor deve possuir uma formação em computação para ministrar a UI desenvolvida com o objetivo de ser um curso para ensinar computação para professores atuantes em qualquer área do conhecimento

(exceto computação) e alinhada aos guias de currículo e complexidade dos conceitos (ALVES *et al.*, 2019). Mas o conteúdo do curso inclui não apenas conhecimento de conteúdo de computação, mas também componentes de tecnologia e pedagogia. A unidade instrucional foi projetada como resultado de trabalhos acadêmicos, no âmbito da Iniciativa Computação na Escola (ALVES, 2016; MISSFELDT FILHO, 2019; PINHEIRO, 2019; FERREIRA, 2019; ALVES, 2019; JUSTEN, 2019).

A UI utiliza diferentes métodos instrucionais, os quais variam desde instrução direita até atividades práticas, atividades abertas e atividades fechadas de programação. Também são utilizadas atividades baseadas no desenvolvimento de projetos adotando a abordagem de ação computacional (TISSENBAUM, SHELDON & ABELSON, 2019). A ação computacional propõe que ao mesmo tempo em que os estudantes aprendem sobre computação, também têm a oportunidade de criar soluções computacionais (como apps e jogos) que tenham impacto direto em suas vidas e em suas comunidades (TISSENBAUM, SHELDON & ABELSON, 2019). Assim, a unidade instrucional prevê que os professores desenvolvam seus próprios projetos de jogos e criem soluções computacionais que possam ser utilizadas em suas aulas, motivando a interdisciplinaridade da aplicação.

Com base na análise de contexto e também considerando as necessidades citadas no guia de currículo CSTA (2016) referente ao ensino de computação no Ensino Fundamental Anos Finais é definido a estratégia instrucional. Isso inclui a definição dos conteúdos, ambientes de programação, métodos instrucionais e os materiais didáticos. Também é definido como ocorre a avaliação dos participantes da UI. Como resultado é definido o plano de ensino.

4.2.1 Ambientes de programação

Com o intuito de atingir os objetivos de aprendizagem serão utilizados três ambientes de programação baseada em blocos, o Code.org, o Scratch e o App Inventor. Cada ambiente será utilizado para ensinar determinados conteúdos de algoritmos e programação, como mostrado na Figura 30.

Figura 30. Conteúdos ensinados para cada ambiente de programação.

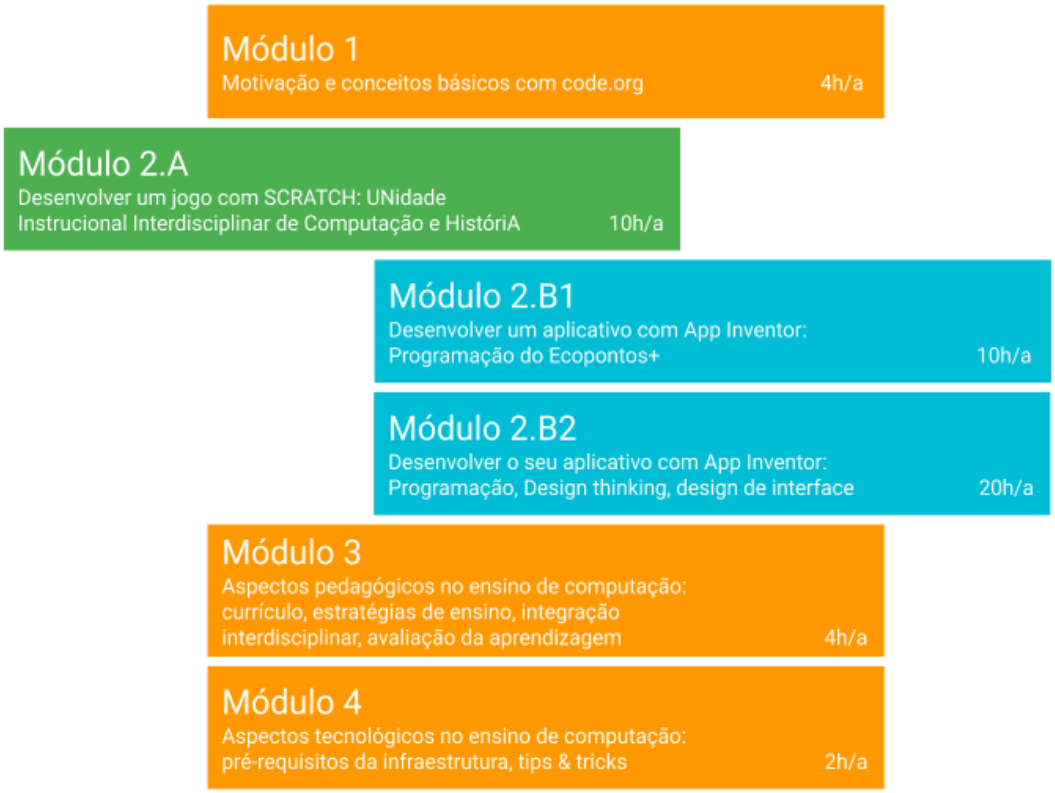
Conteúdos / Ambientes	*Eventos*	*Nomeação*	*Strings*	*Mapa*	*Variáveis*	*Condicionais*	*Operadores*	*Abstrações*	*Sensores*	*Bibliotecas externas (API)*	*Laços*	*Listas*	*Persistência*	*Movimento*	*Aparência*	*Som*
code.org						X					X			X		
Scratch	X				X	X	X		X		X			X	X	X
App Inventor	X	X	X	X	X	X	X	X	X	X	X	X	X			

Fonte: elaborada pela autora.

4.2.2 Plano de ensino

A construção do plano de ensino é baseada na análise do público-alvo, do ambiente das escolas e dos objetivos de aprendizagem definidos anteriormente. A UI foi dividida em módulos como mostrado na Figura 31.

Figura 31. Módulos da unidade instrucional.



Fonte: elaborada pela autora.

O plano de ensino de cada módulo é apresentado a seguir (Tabela 25) (Tabela 26) (Tabela 26) (Tabela 27) (Tabela 28) (Tabela 29) (Tabela 30).

Tabela 25. Plano de ensino do Módulo 1.

Aula (duração)	**Conteúdo em geral**	**ID objetivo de aprendizagem**	**Método instrucional**	**Avaliação**
1 (45min)	Motivação sobre ensino de computação e o desenvolvimento de aplicativos Conceitos básicos de computação: algoritmos/ programação	OA1	Ensino explícito, demonstrações, exercitar e praticar, condução de experimentos	-
2 (45min)	Conceitos básicos (Atividades desplugadas)	OA1	Ensino explícito, demonstrações, condução de experimentos, atividades desplugadas, instrução direta	-
3 (1h30min)	Conceitos básicos de programação com Code.org	OA2, OA4, OA7	Ensino explícito, demonstrações, condução de experimentos, atividades fechadas de programação, instrução direta	Artefatos desenvolvidos

Fonte: elaborada pela autora.

Tabela 26. Plano de ensino do Módulo 2.A.

Aula (duração)	**Conteúdo em geral**	**ID objetivo de aprendizagem**	**Método instrucional**	**Avaliação**
1 (3h)	Introdução ao Scratch	OA1, OA4, OA5, OA6, OA7	Ensino explícito, demonstrações, exercitar e praticar, condução de experimentos	-
2 (3h)	Concepção de um jogo interdisciplinar	OA1, OA4, OA5, OA6, OA9, OA10, OA11, OA15	Ensino explícito, demonstrações, condução de experimentos, atividades abertas de programação, instrução direta	Artefatos desenvolvidos

3 (1h30min)	Apresentação dos projetos e discussão	OA9, OA10	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos

Fonte: elaborada pela autora.

Tabela 27. Plano de ensino do Módulo 2.B1.

Aula (duração)	**Conteúdo em geral**	**ID objetivo de aprendizagem**	**Método instrucional**	**Avaliação**
1 (45min)	O ambiente App Inventor Programar e testar conceitos básicos de computação: Eventos, Nomeação e *Strings*	OA1, OA2, OA3, OA4, OA8	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos
2 (45min)	Programar e testar conceitos de computação: Mapa e Variáveis	OA1, OA2, OA3, OA5, OA8	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos
3 (45min)	Programar e testar conceitos básicos de computação: Condicionais e Operadores	OA1, OA2, OA4, OA5	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos
4 (45min)	Programar e testar conceitos de computação: Abstração	OA1, OA2, OA8	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos
5 (45min)	Programar e testar conceitos de computação: Sensores (GPS)	OA1, OA2, OA5	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos
6 (45min)	Programar e testar conceitos básicos de computação: Laços e Listas	OA1, OA2, OA4, OA8	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos
7 (45min)	Programar e testar conceitos de computação: API	OA1, OA2, OA8	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos
8 (1h30min)	Programar e testar conceitos de computação: Persistência	OA1, OA2, OA8	Ensino explícito, demonstrações, condução de experimentos	Artefatos desenvolvidos
9 (45min)	Compartilhamento do aplicativo	OA9	Ensino explícito, condução de experimentos	Artefatos desenvolvidos

	Experimentação de outros aplicativos da Galeria App Inventor			

Fonte: elaborada pela autora.

Tabela 28. Plano de ensino do Módulo 2.B2.

Aula (duração)	**Conteúdo em geral**	**ID objetivo de aprendizagem**	**Método instrucional**	**Avaliação**
1 (1h30min)	Descoberta - Identificar necessidade e analisar contexto	OA10, OA11, OA12, OA15	Ensino explícito, demonstrações, resolução de problemas, instrução direta	Artefatos desenvolvidos
2 (1h30min)	Ideação - Imaginar solução e especificar requisitos	OA10, OA13, OA15	Ensino explícito, demonstrações, instrução direta	Artefatos desenvolvidos
3 (1h30min)	Construção e teste - Criar e testar *sketches*	OA14, OA15	Ensino explícito, demonstrações, atividades abertas de programação, instrução direta	Artefatos desenvolvidos
4 (1h30min)	Construção e teste - Criar e testar o *wireframe*	OA14, OA15	Ensino explícito, demonstrações, atividades abertas de programação, instrução direta	Artefatos desenvolvidos
5 (3h)	Construção e teste - Criar design visual	OA15, OA18	Ensino explícito, demonstrações, atividades abertas de programação, instrução direta	Artefatos desenvolvidos
6 (3h)	Construção e teste - Programar e realizar teste Compartilhamento do aplicativo	OA9, OA15, OA16, OA17, OA19, OA20, OA21	Ensino explícito, demonstrações, atividades abertas de programação, instrução direta	Artefatos desenvolvidos

Fonte: elaborada pela autora.

Tabela 29. Plano de ensino do Módulo 3.

Aula (duração)	**Conteúdo em geral**	**ID objetivo de aprendizagem**	**Método instrucional**	**Avaliação**
1 (45min)	Diretrizes de currículo referente ao ensino de computação na Educação	OA22	Ensino explícito	Exercícios/ quiz

	Básica			
2 (45min)	Ambientes para ensinar computação	OA25, OA26	Ensino explícito	Exercícios/ quiz
3 (45min)	Como inserir o ensino de computação na disciplina/ escola	OA23, OA25	Ensino explícito	Exercícios/ quiz
4 (45min)	Como avaliar a aprendizagem do aluno	OA24	Ensino explícito	Exercícios/ quiz

Fonte: elaborada pela autora.

Tabela 30. Plano de ensino do Módulo 4.

Aula (duração)	**Conteúdo em geral**	**ID objetivo de aprendizagem**	**Método instrucional**	**Avaliação**
1 (1h30min)	Preparar e testar a infraestrutura para realizar as aulas	OA26, OA27, OA28	Ensino explícito	-
1 (1h30min)	Como agir quando acontece problemas técnicos durante as aulas	OA26, OA27, OA28	Ensino explícito	-

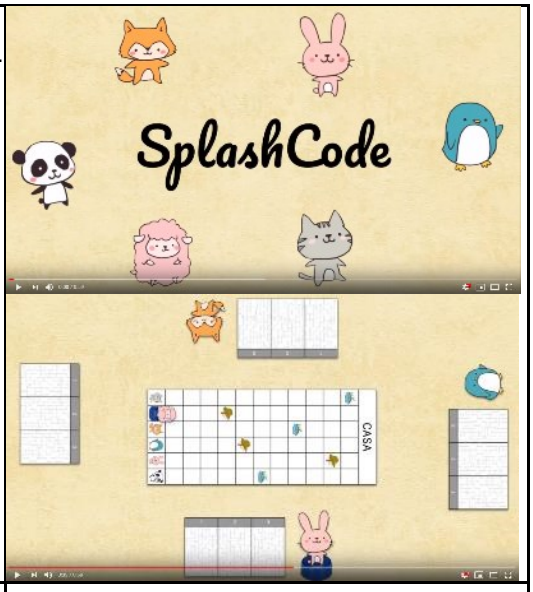
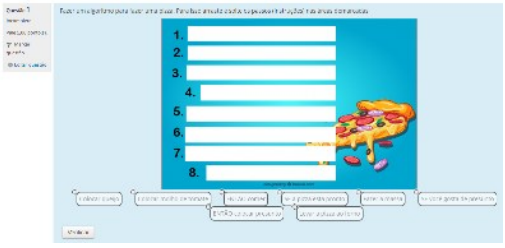
Fonte: elaborada pela autora.

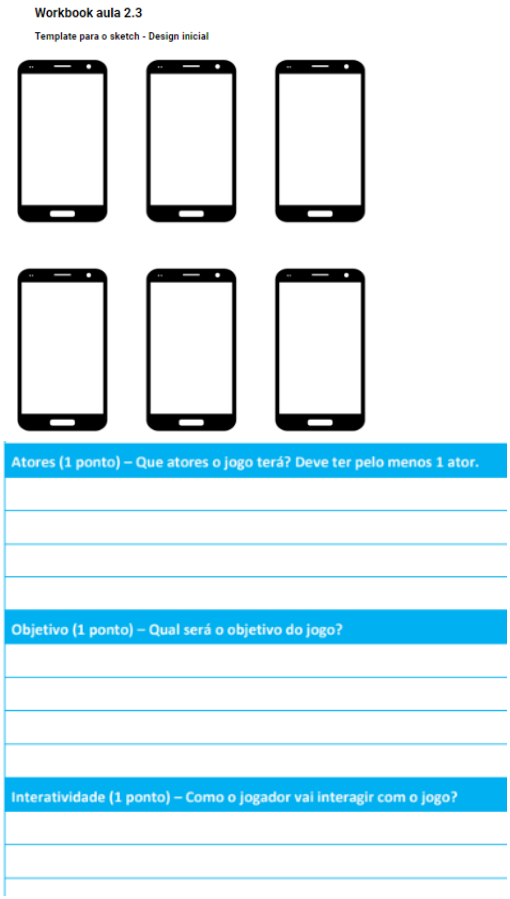
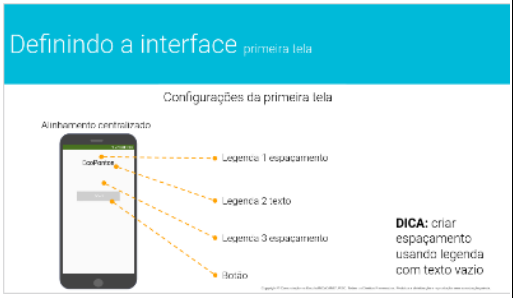
4.2.3 Desenvolvimento do material didático

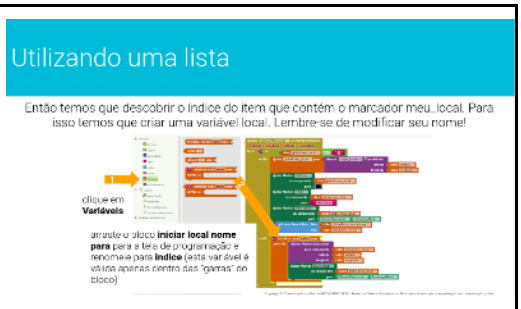
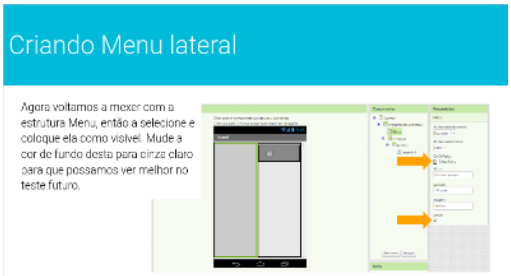
Os materiais didáticos estão sendo desenvolvidos com base no plano de ensino. A Tabela 31 apresenta alguns exemplos de materiais em desenvolvimento. Alguns materiais didáticos foram desenvolvidos com base nas UI para o Educação Básica (MISSFELDT FILHO, 2019; PINHEIRO, 2019; FERREIRA, 2019; ALVES, 2019).

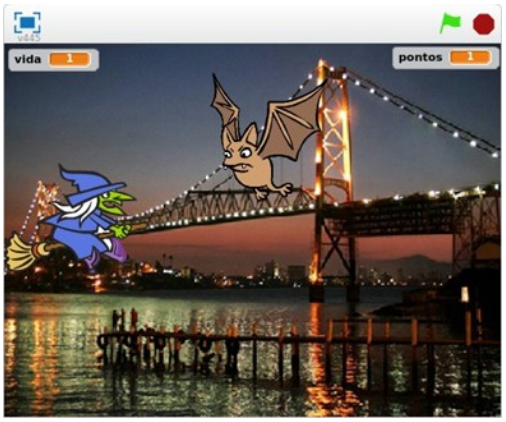
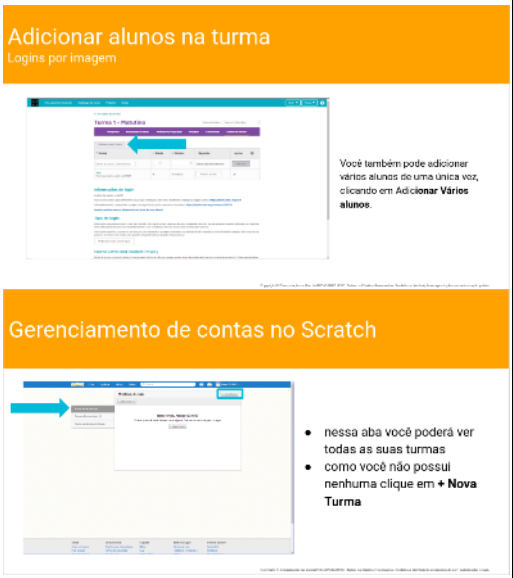
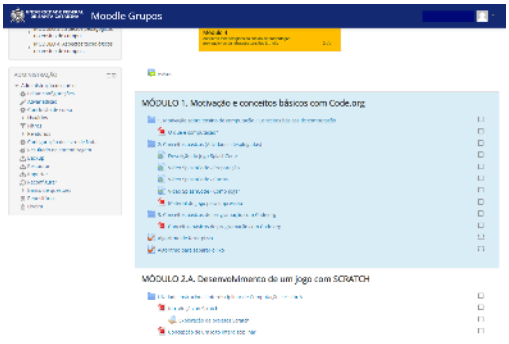
Tabela 31. Materiais didáticos em desenvolvimento.

Material	**Descrição**	**Ilustração**
Jogo *SplashCode* (GRESSE VON WANGENHEIM *et al.*, 2019)	Jogo de tabuleiro para a aplicação dos conceitos de algoritmos. Os jogadores utilizam as cartas de movimento para fazer os personagens chegarem até o campo *casa*, desviando dos obstáculos.	

Vídeos	Vídeos de auxílio para aprender a como jogar o jogo *SplashCode*. Existe um conjunto de três vídeos curtos: “preparação”, “cartas” e “como jogar”.	
Slides com conteúdo	Slides com o conteúdo de todos os conceitos abordados na UI.	
Exercícios digital	Exercícios para os professores colocarem em prática o conteúdo presente nos slides.	

Exercícios desplugados	Atividade de criar o círculo cromático.	
Folha de atividades	Guia para a produção dos artefatos propostos no processo de desenvolvimento de apps e de jogos.	
Tutoriais EcoPontos+	Desenvolvimento e tutorial para construção do app EcoPontos. Este app tem por objetivo apresentar informações sobre os locais e coletas de resíduos sólidos. Durante o desenvolvimento do app pelo tutorial, vários outros conceitos de computação são abordados, como variável,	

	eventos e outros.	
Tutoriais de funcionalidades específicas do App Inventor	Tutoriais para desenvolvimento de funcionalidades características de diversos apps no App Inventor, como banco de dados, menu lateral, *login* e mapas.	 
Tutoriais de criação de jogos	Desenvolvimento e tutorial de como utilizar diversos conceitos de computação no Scratch.	
Exemplos de jogos para inserção interdisciplinar	Códigos exemplos de jogos interdisciplinares feitos no Scratch	

		
Tutoriais de funcionalidades úteis para o professor	Tutoriais para utilização de algumas funcionalidades encontradas nos ambientes de programação Code.org, Scratch e App Inventor.	
Moodle do curso	Moodle é utilizado para facilitar o acesso aos materiais de aula, em razão do curso ser à distância.	

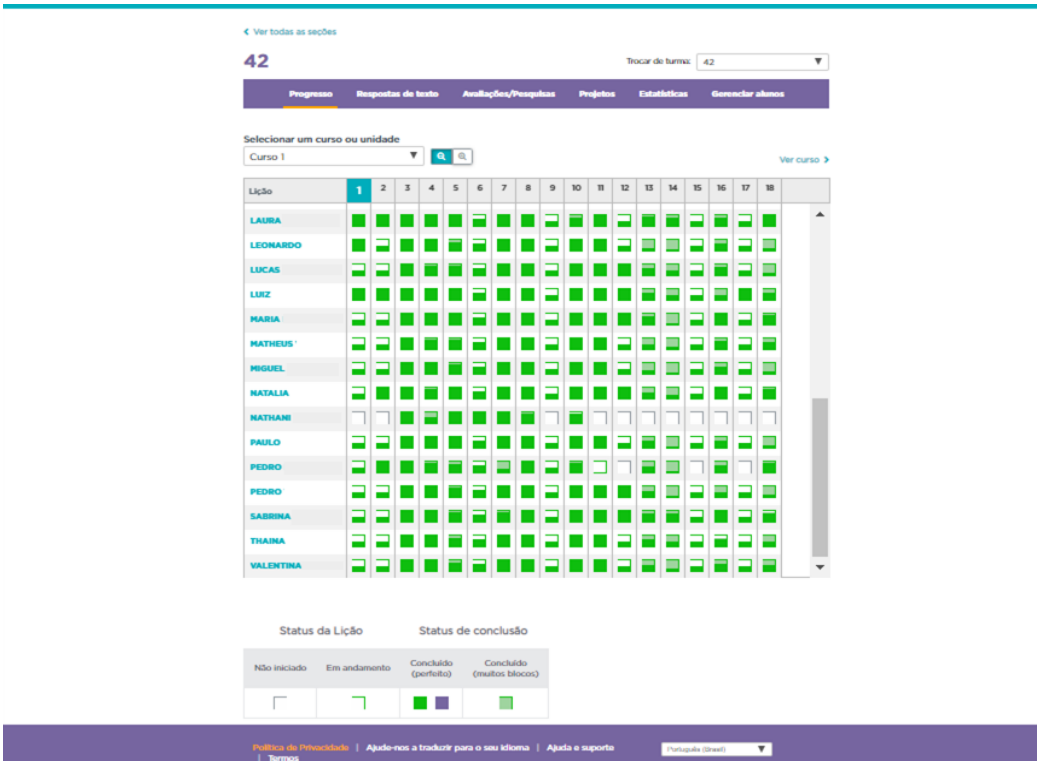
Fonte: elaborada pela autora.

4.2.4 Avaliação do aluno

A avaliação do aluno é realizada na forma de avaliação de desempenho. As avaliações por desempenho serão realizadas por meio de rubricas e ferramentas de análise de progresso dos exercícios no Code.org (CODE.ORG, 2019), projetos Scratch (SCRATCH, 2019) e App Inventor (MITa, 2019).

Os projetos realizados no Code.org serão avaliados de acordo com a tela de Progresso, disponível na conta para Educadores. Na tela de Progresso (Figura 32) visualizam-se as atividades que foram concluídas pelos professores (cor verde claro ou escuro) e as que não foram realizadas (sem cor). Nas atividades concluídas, pode-se observar também se foi concluída de forma completa (cor verde escuro) ou com mais blocos do que o necessário, neste caso a atividade é marcada com a cor verde claro. Dessa forma, a tela de Progresso pode ser utilizada para descobrir as lições que os professores realizaram e as atividades que os professores tiveram mais dificuldades. Então, para avaliar esse projeto é criado uma seção no Code.org, para acompanhar o status de conclusão das atividades dos professores, e assim atribuir uma nota para cada professor. A pontuação será de 0 (mínima) até o número de lições que o curso possuir (máxima).

Figura 32. Tela que indica o progresso das lições no Code.org.

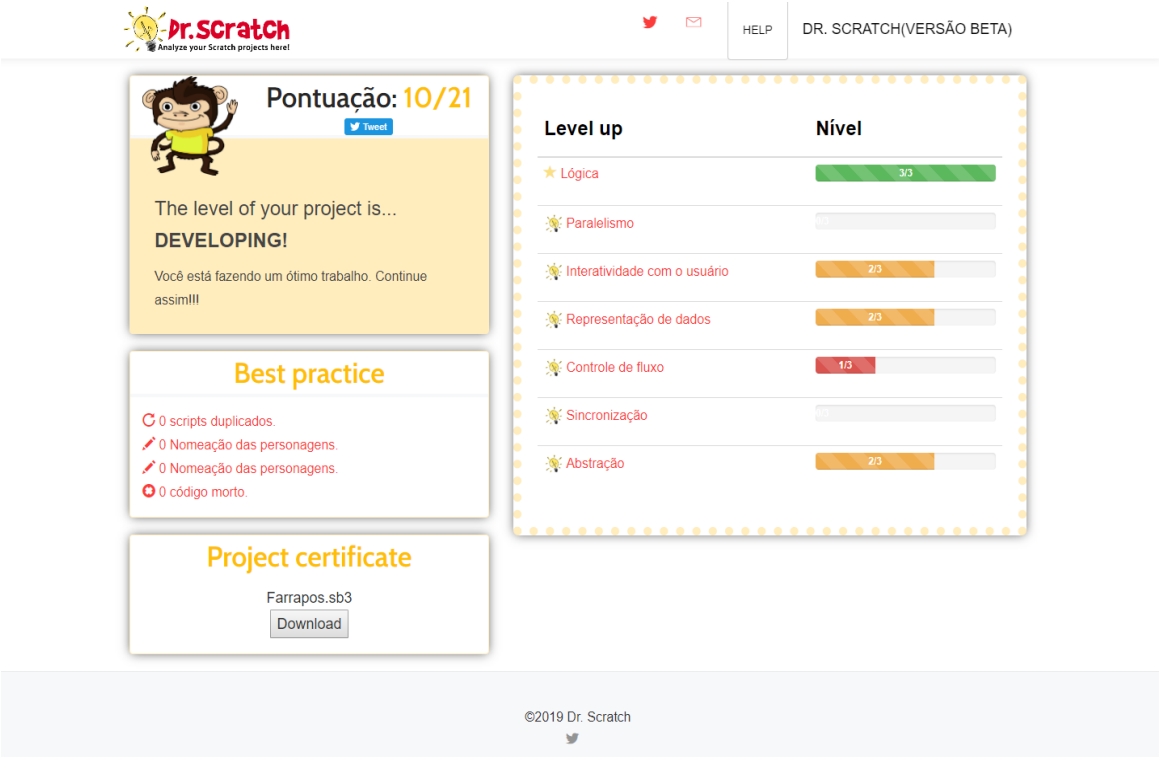


Fonte: CODE.ORG (2019).

Os jogos desenvolvidos no Scratch são avaliados utilizando a ferramenta Dr. Scratch (Figura 33) (MORENO-LEÓN & ROBLES, 2015), desenvolvido pela associação Programamos, Universidade Rey Ruan Carlos e outros contribuidores.

Essa ferramenta é disponível online e gratuita (www.drscratch.org), na qual permite avaliar automaticamente conceitos de pensamento computacional a partir dos projetos desenvolvidos com o Scratch.

Figura 33. Projeto avaliado pelo *Dr. Scratch.*



Fonte: MORENO-LEÓN e ROBLES (2015).

A ferramenta Dr. Scratch utiliza uma rubrica para avaliar cada jogo. A rubrica utiliza os critérios definidos na Tabela 32 e o cálculo da nota é realizado por meio da soma das notas dos critérios, gerando a nota final numa escala de 0 a 21.

Tabela 32. Rubrica da ferramenta Dr. Scratch.

Critério	**0 pontos**	**1 ponto**	**2 pontos**	**3 pontos**
Lógica	Não utilizou nenhum comando de lógica.	Utilização do comando “Se, então”.	Utilização do comando “Se, então; senão”.	Utilização de comandos de operações lógicas que combinam as condições.

Paralelismo	Não utilizou nenhum comando de paralelismo.	2 scripts iniciando com “bandeira verde”.	2 scripts com o comando “quando a tecla for pressionada” utilizando a mesma tecla ou 2 scripts com o comando “quando este ator for clicado” utilizando o mesmo ator.	2 scripts de recebimento de mensagens ou criação de clones ou 2 scripts de sensores ou 2 scripts de mudança de pano de fundo.
Interatividade com o usuário	Não utilizou nenhum comando de interatividade com o usuário.	Utilização do comando da “bandeira verde”.	Utilização de comandos de teclas/atores pressionadas, perguntas ao usuário, e botão do mouse.	Utilização de comandos de sensor de câmera e vídeo.
Representação de dados	Não utilizou nenhum comando de representação de dados.	Modificação de propriedades dos atores como coordenadas, tamanho e aparência.	Operações com variáveis.	Operações com listas.
Controle de fluxo	Não utilizou nenhum comando de controle de fluxo.	Programação de uma sequência de blocos.	Utilização dos comandos “repita x vezes” e “sempre”.	Utilização do comando “repita até que”.
Sincronização	Não utilizou nenhum comando de sincronização.	Utilização do comando “espere”.	Utilização de comandos de envio/recebimento de mensagens.	Utilização de comandos de “espere até” ou “quando o fundo muda para”.
Abstração	Não utilizou nenhum comando de abstração.	Programação de mais de 1 script.	Utilização e programação de funções por meio do comando “defina”.	Utilização de clones.

Fonte: MORENO-LEÓN e ROBLES (2015).

Além da avaliação dos jogos criados pelos professores, uma rubrica foi desenvolvida para avaliar os artefatos (Folha de atividades) criados durante o Módulo 2.A referente as aulas de implementação de um jogo no Scratch (Tabela 33). Essa rubrica possibilita avaliar os professores em termos do desempenho na criação destes artefatos com uma pontuação de 0 (mínima) a 10 (máxima). Os artefatos textuais serão avaliados de acordo com o tipo de jogo implementado e o tutor das aulas ficará responsável por utilizar a rubrica para avaliar os professores participantes.

Tabela 33. Rubrica para avaliação dos artefatos textuais produzidos na implementação de um jogo Scratch.

Tipo de jogo	**Item**	**0 pontos**	**0,5 pontos**	**1 ponto**	**2 pontos**	**3 pontos**	**4 pontos**
Ação	Atores - Que atores o jogo terá? Deve ter pelo menos 1 ator.	Não apresentou nada	-	Apresentou 1 ator	-	-	-
	Objetivo - Qual será o objetivo do jogo?	Não apresentou nada	-	Apresentou o objetivo de forma incompleta	Apresentou o objetivo de forma clara	-	-
	Interatividade - Como o jogador vai interagir com o jogo?	Não apresentou nada	-	Apresentou a interatividade de forma incompleta	Apresentou a interatividade de forma clara	-	-
	Regras - Quais serão as regras do jogo? Deve ter pelo menos 3 regras.	Não apresentou nada	-	Apresentou 1 regra	Apresentou 2 regras	Apresentou 3 ou mais regras	-
	Derrota - O que faz o jogador perder o jogo?	Não apresentou nada	-	Apresentou a condição de derrota de forma incompleta	Apresentou a condição de derrota de forma clara	-	-
							Máximo de pontos atingível: 10 pontos
Aventura	Atores - Que atores o jogo terá? Deve ter pelo menos 3 atores.	Não apresentou nada	-	Apresentou 1 ator	Apresentou 2 atores	Apresentou 3 ou mais autores	-
	Objetivo - Qual será o objetivo do jogo?	Não apresentou nada	Apresentou o objetivo de forma incompleta	-	Apresentou o objetivo de forma clara	-	-
	Interatividade - Como o jogador vai interagir com o jogo?	Não apresentou nada	-	Apresentou a interatividade de forma incompleta	Apresentou a interatividade de forma clara	-	-
	Sequência da história - Escreva as falas de cada ator. Deve ter pelo menos 12 falas.	Apresentou 1 fala ou menos	-	Apresentou 2 a 4 falas	Apresentou 5 a 8 falas	Apresentou 9 a 11 falas	Apresentou 12 falas ou mais
							Máximo de pontos atingível: 10 pontos
Quiz	Atores - Que atores o jogo terá? Deve ter	Não apresentou nada	-	Apresentou 1 ou mais autores	-	-	-

	pelo menos 1 ator.						
	Objetivo - Qual será o objetivo do jogo?	Não apresentou nada	Apresentou o objetivo de forma incompleta	Apresentou o objetivo de forma clara	-	-	-
	Interatividade - Como o jogador vai interagir com o jogo?	Não apresentou nada	Apresentou a interatividade de forma incompleta	Apresentou a interatividade de forma clara	-	-	-
	Regras - Quais serão as regras do jogo? Deve ter pelo menos 2 regras.	Não apresentou nada	-	Apresentou 1 regra	Apresentou 2 ou mais regras	-	-
	Derrota - O que faz o jogador perder o jogo?	Não apresentou nada	Apresentou a condição de derrota de forma incompleta	Apresentou a condição de derrota de forma clara	-	-	-
	Perguntas - Escreva as perguntas e respostas do jogo. Deve ter pelo menos 12 perguntas.	Apresentou 1 pergunta ou menos		Apresentou 2 a 4 perguntas	Apresentou 5 a 8 perguntas	Apresentou 9 a 11 perguntas	Apresentou 12 ou mais perguntas
	Máximo de pontos atingível: 10 pontos						

Fonte: ALVES (2016).

A apresentação do jogo também será avaliada pelo tutor. A nota será dada com base na avaliação da explicação dos professores sobre os blocos de comandos e o tema escolhido, com uma pontuação de 0 (mínima) a 10 (máxima).

Tabela 34. Rubrica para avaliação da apresentação do jogo Scratch.

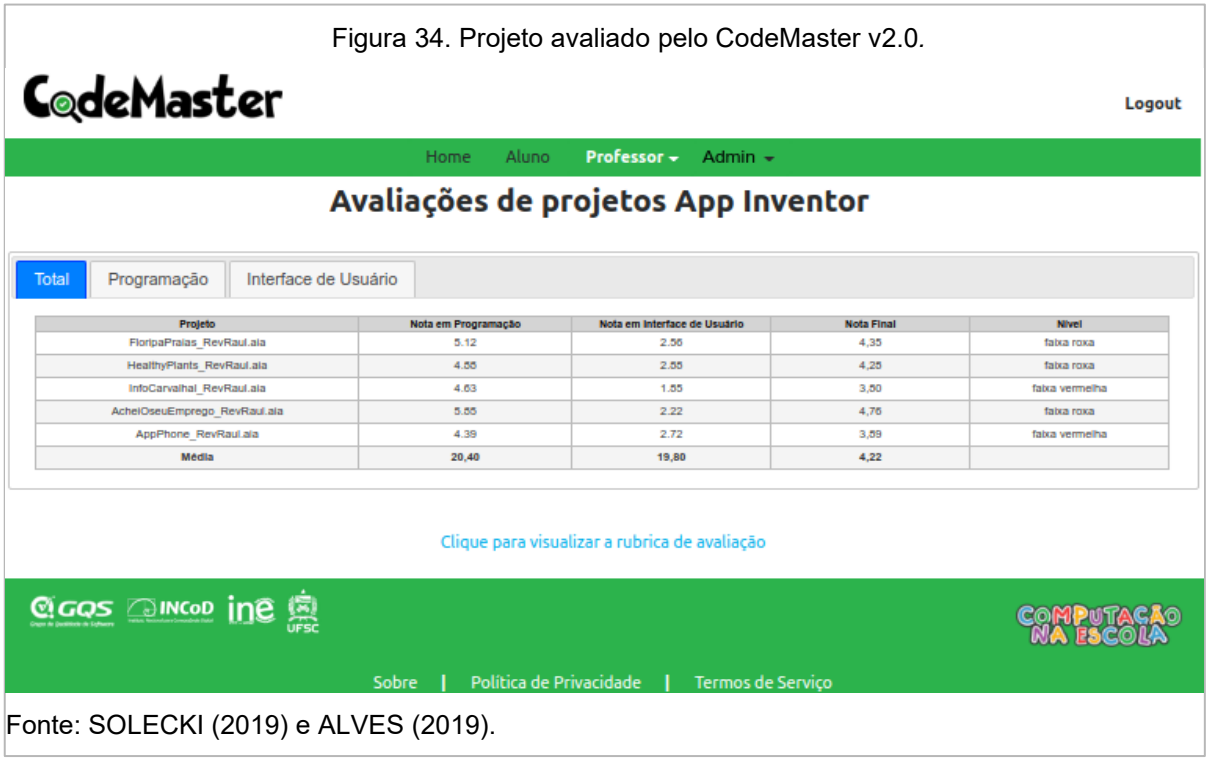
Tópico	**Insuficiente (0 pontos)**	**Regular (2 pontos)**	**Bom (3,5 pontos)**	**Excelente (5 pontos)**
Bloco de comandos	O aluno não soube explicar nenhum comando de forma correta.	O aluno soube explicar corretamente, com alguma dificuldade, poucos comandos.	O aluno soube explicar corretamente, sem dificuldades, alguns comandos passo-a-passo.	O aluno soube explicar corretamente, sem dificuldades, todos os comandos passo-a-passo, isto é, de forma sistemática.
Tema Escolhido	O aluno não apresentou nenhum argumento lógico com relação ao	O aluno apresentou argumentos lógicos com pouca relação	O aluno apresentou alguns argumentos lógicos que	O aluno apresentou argumentos lógicos que relacionam o jogo

	tema escolhido e não demonstrou nenhum conhecimento sobre o modo de vida de diferentes grupos.	com o tema escolhido e demonstrou pouco conhecimento sobre o modo de vida de diferentes grupos.	relacionam o jogo com o tema escolhido e demonstrou algum ou pouco conhecimento sobre o modo de vida de diferentes grupos.	com o tema escolhido e demonstrou conhecimento sobre o modo de vida de diferentes grupos.
				Máximo de pontos atingível: 10 pontos

Fonte: ALVES (2016).

Os aplicativos desenvolvidos no módulo 2.B2 com o App Inventor são avaliados utilizando a ferramenta CodeMaster v2.0 (Figura 34) (SOLECKI, 2019; ALVES, 2019). A ferramenta foi desenvolvida pela Iniciativa Computação na Escola e está disponível de forma *online* e gratuita (http://apps.computacaonaescola.ufsc.br:8080/). O CodeMaster v2.0 é utilizado para avaliar os conceitos de pensamento computacional e o design visual do aplicativo.

Figura 34. Projeto avaliado pelo CodeMaster v2.0.



Fonte: SOLECKI (2019) e ALVES (2019).

O CodeMaster v2.0 (SOLECKI, 2019; ALVES, 2019) rubrica avalia para cada app desenvolvido uma série de questões sobre o desempenho em relação aos conceitos do pensamento computacional baseada sobretudo no currículo da CSTA

(2016) e do design visual do aplicativo. A Figura 35 e Figura 36 ilustram um extrato da rubrica de design visual e pensamento computacional, respectivamente.

Figura 35. Extrato da rubrica de design visual.

Critério	0 pt	1 pt	2pt
Layout			
L1. O dimensionamento é fixo ou responsivo?	Fixo	Responsivo	
L2. Quantas telas usam organizadores?	Menos de 90%	90% ou mais	
L3. Todos os componentes alvos de toque têm largura e altura maior ou igual a 48 pixels?	Não	Sim	
L4. Todos os botões têm o mesmo formato?	Não	Sim	
L5. Botões agrupados na interface sempre têm o mesmo tamanho?	Não	Sim	
L6. Quantos elementos têm a tela com menos elementos (min) e a tela com mais elementos (max)?	$min < 2$ ou $max \geq 20$	$min \geq 2$ e $max \in [10, 19]$	$min \geq 2$ e $max \leq 9$
Tipografia			
T1. Quantos componentes têm família da fonte sem serifa?	Menos de 90%	90% ou mais	
T2. Quantos botões com texto têm tamanho da fonte igual a 14?	Menos de 90%	90% ou mais	
T3. Quantos componentes (exceto botões) têm tamanho da fonte igual a um dos tamanhos recomendados?	Menos de 90%	90% ou mais	
T4. Há texto em itálico?	Sim	Não	
T5. O texto de todos os botões é centralizado?	Não	Sim	

Fonte: SOLECKI (2019).

Figura 36. Extrato da rubrica de pensamento computacional.

PA2. Qual o nível de desempenho em representação de dados com relação às práticas do pensamento computacional? (CSTA, 2016; 2017)				
Item	**0 pontos**	**1 ponto**	**2 pontos**	**3 pontos**
Variáveis: verificar se criou ou modificou valores de variáveis.	Sem uso de variáveis.	Modificação ou uso de variáveis predefinidas.	Criação e operação com variáveis	
Strings: verificar se criou ou modificou valores de strings.	Sem uso de strings.	Uso do comando de criação de string para alterar textos de elementos.	Criação e operação com strings.	
Nomeação: verificar se os nomes de variáveis são alterados do padrão.	Nenhum ou poucos nomes são alterado do padrão. (menos do que 10%)	De 10 a 25% dos nomes são alterados do padrão.	De 26 a 75% dos nomes são alterados do padrão.	Mais de 76% dos nomes são alterados do padrão.
Listas: verificar se são usadas listas.	Não usa listas.	Usa uma lista unidimensional.	Usa mais de uma lista unidimension al.	Usa uma lista de tuplas (map).
Persistência de dados: verificar se são usados componentes de persistência de dados	Dados são armazenados em variáveis ou propriedades de componentes e não tem persistência	Usa persistência em arquivo (File ou Fusion Tables).	Usa algum dos bancos de dados locais do App Inventor (TinyDB).	Usa uma base de dados web tinywebdb ou Firebase do App Inventor (firebase ou TinyWebDB).

Fonte: ALVES (2019).

Também são avaliados os artefatos (Folha de atividades) criados durante o Módulo 2.B2 referente a aprendizagem de engenharia de software no desenvolvimento de aplicativos com o App Inventor (Tabela 37). E assim, avaliar os professores em termos do desempenho na criação destes artefatos com uma

pontuação de 0 (mínima) a 48 (máxima). A avaliação é realizada por meio da análise das Folhas de atividades pelo tutor.

Tabela 35. Rubrica para avaliação dos artefatos textuais produzidos na implementação de um aplicativo no App Inventor.

Folha de atividades	**Item**	**0 pontos**	**1 ponto**	**2 pontos**	**3 pontos**
Aula 2.B2.1	Descrição da necessidade/ problema	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Caracterização dos usuários	Não soube caracterizar a persona	Caracterizou incompleto	Caracterizou com pouca clareza	Caracterizou com clareza
	Características dos celulares dos usuários	Não soube caracterizar a persona	Caracterizou incompleto	Caracterizou com pouca clareza	Caracterizou com clareza
	Características do ambiente que o usuário utiliza o app	Não soube caracterizar a persona	Caracterizou incompleto	Caracterizou com pouca clareza	Caracterizou com clareza
Aula 2.B2.2	Descrição da solução	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Histórias de usuário	Não soube contar	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Resultados de usabilidade	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
Aula 2.B2.3	Design de interface inicial	Não soube realizar	Realizou incompleto	Realizou com pouca clareza	Realizou com clareza
	Resultados do teste do design da interface	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Design de interface final	Não soube realizar	Realizou incompleto	Realizou com pouca clareza	Realizou com clareza
Aula 2.B2.6	Resultados do teste funcional	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
	Resultados do teste de usabilidade	Não soube descrever	Descreveu incompleto	Descreveu com pouca clareza	Descreveu com clareza
					Total de pontos atingível: 48

Fonte: adaptado de MISSFELD FILHO (2019).

A nota da avaliação deste artefato é calculada com base na pontuação total recebida pelo professor nas Folhas de atividade, conforme indicado abaixo:

$$\text{Nota} = \left(\frac{\text{somatório da pontuação recebida}}{48} \right) \text{X } 10$$

5. APLICAÇÃO E AVALIAÇÃO DA UNIDADE INSTRUCIONAL

Nesse capítulo é apresentado o planejamento da avaliação, definindo como a UI será aplicada e avaliada. Também será a descrita a aplicação da UI e ao final, é apresentado a análise dos dados obtidos durante a aplicação.

5.1 DEFINIÇÃO DA AVALIAÇÃO DA UNIDADE INSTRUCIONAL

A avaliação da UI tem o propósito de avaliar a percepção da qualidade em termos de qualidade da unidade instrucional, experiência de computação e percepção de aprendizagem do ponto de vista dos participantes (professores) no contexto do ensino de computação para professores da Educação Básica.

Pergunta de pesquisa

A formação de professores *in-service* da Educação Básica, referente ao ensino de computação, contribui para o desenvolvimento de competências e motivação/interesse em ensinar computação dos professores de outras áreas de conhecimento e apresenta qualidade suficiente?

Design de pesquisa

Estudo de caso: *one-shot time series post-test* (X O X O X O X O X O).

Medição

Para a definir a medição da avaliação é utilizada a abordagem GQM (BASILI et al., 1994), que consiste em sistematicamente decompor a pergunta de pesquisa em perguntas de análise e medidas e o desenvolvimento de instrumentos de medição. Assim, a partir da pergunta de pesquisa foram derivadas as perguntas de análise.

Perguntas de análise

PA1. O design instrucional da formação de professores *in-service* da Educação Básica referente ao ensino de computação **proporciona aprendizagem de competências em relação aos conteúdos de computação**?

PA2. O design instrucional da formação de professores *in-service* da Educação Básica referente ao ensino de computação **promove uma experiência de aprendizagem agradável e divertida**?
PA3. O design instrucional da formação de professores *in-service* da Educação Básica referente ao ensino de computação **facilita a aprendizagem (usabilidade da unidade instrucional)**?
PA4. O design instrucional da formação de professores *in-service* da Educação Básica referente ao ensino de computação **proporciona o interesse dos professores para ensinar conteúdos de computação** em suas disciplinas?

Em relação a pergunta de análise PA1 espera-se que os professores adquiram conhecimento em conteúdo de computação, bem como em aspectos pedagógicos e tecnológicos necessários para ensinar computação, conforme indicado nos objetivos de aprendizagem dessa unidade instrucional (Tabela 36). Já as demais perguntas de análise se referem a qualidade da unidade instrucional. Assim as perguntas PA2, PA3 e PA4 são avaliadas utilizando um questionário pós-módulo (APÊNDICE II) e seguindo o modelo dETECT (GRESSE VON WANGENHEIM *et al*., 2019), sendo esperadas que os professores realizem atividades agradáveis que facilitam a aprendizagem, que despertam uma percepção positiva e interesse em ensinar conteúdo de computação nas aulas de suas disciplinas.

A partir das perguntas de análise são derivados as medidas e os respetivos instrumentos de coleta de dados (Tabela 36).

Tabela 36. Síntese da definição da avaliação da unidade instrucional.

Pergunta de análise	**Medida**	**Objetivo de aprendizagem**	**Instrumentos de coleta de dados**
PA1. A UI proporciona aprendizagem de competências em relação aos conteúdos de computação?	M1.1 Grau de aprendizagem referente a capacidade realizar atividades fechadas de programação	OA1, OA2, OA4, OA7, OA10	- Status de conclusão da lição no Code.org (Figura 32 na seção de 4.2.4) - Questionário pós-módulo (Pergunta 8)
	M1.2 Grau de aprendizagem referente a capacidade de fazer jogos	OA1, OA2, OA4, OA6 OA7, OA9, OA10, OA11, OA15	- Dr. Scratch (pensamento computacional) - Rubrica para avaliar apresentação do jogo

			feito no Scratch (Tabela 34 na seção 4.2.4) - Rubrica para avaliar as Folhas de atividades (Tabela 33 na seção 4.2.4) - Questionário pós-módulo (Pergunta 8)
	M1.3 Grau de aprendizagem referente a capacidade de fazer aplicativos para smartphones	OA1, OA2, OA3, OA4, OA5, OA8, OA9, OA10, OA11, OA12, OA13, OA14, OA15, OA16, OA17, OA18, OA19, OA20, OA21	- CodeMaster v2.0 (pensamento computacional e UI design) - Rubrica para avaliar as Folhas de atividades (Tabela 35 na seção 4.2.4) - Questionário pós-módulo (Pergunta 8)
	M1.4 Grau de aprendizagem referente à capacidade de descrever um algoritmo em uma sequência de instruções	OA1, OA10	- Status de conclusão da lição no Code.org (Figura 32 na seção 4.2.4) - Questionário pós-módulo (Pergunta 10)
	M1.5 Grau de habilidade para ensinar os conteúdos aprendidos a outras pessoas	OA9, OA22, OA23, OA24, OA25, OA26, OA27, OA28	- Questionário pós-módulo (Pergunta 10)
PA2. A UI promove uma experiência de aprendizagem agradável e divertida?	M2.1 Grau de diversão das aulas	-	- Questionário pós-módulo (Pergunta 3)
	M2.2 Grau de percepção da passagem do tempo durante as aulas	-	- Questionário pós-módulo (Pergunta 4)
	M2.3 Grau da interação social (compartilhar a experiência com outras pessoas)	-	- Questionário pós-módulo (Pergunta 10)
	M2.4 Opinião subjetiva sobre os pontos que dão qualidade a aula	-	- Questionário pós-módulo (Pergunta 1)
PA3. A UI facilita a aprendizagem (usabilidade da unidade)?	M3.1 Grau de facilidade das aulas	-	- Questionário pós-módulo (Pergunta 2 e 12)
	M3.2 Grau de qualidade geral das aulas	-	- Questionário pós-módulo (Pergunta 5)
	M2.3 Pontos fortes das aulas	-	- Questionário pós-módulo (Pergunta 6)

	M2.4 Pontos fracos das aulas	-	- Questionário pós-módulo (Pergunta 7)
PA4. A UI proporciona o interesse dos professores para ensinar conteúdos de computação em suas disciplinas?	M4.1 Grau de satisfação em realizar atividades fechadas de programação	-	- Status de conclusão das lições no Code.org (Figura 32 na seção de 4.2.4) - Questionário pós-módulo (Pergunta 3, 4, 9 e 11)
	M4.2 Grau de satisfação em fazer jogos	-	- Questionário pós-módulo (Pergunta 3, 4, 9 e 11)
	M4.3 Grau de satisfação em fazer apps para smartphones	-	- Questionário pós-módulo (Pergunta 3, 4, 9 e 11)
	M4.4 Percepção da importância do ensino da computação na Educação Básica	-	- Questionário pós-módulo (Pergunta 13)
	M4.5 Intenção de ensinar conceitos de computação nas suas disciplinas na Educação Básica	-	- Questionário pós-módulo (Pergunta 9 e 10)

Fonte: elaborada pela autora.

Coleta de dados

Considerando que a unidade instrucional “Aprenda a ensinar algoritmos e programação” é dividida em módulos, é planejada a coleta de dados ao final de cada módulo. A Tabela 37 apresenta o planejamento da coleta de dados durante a aplicação, ressaltando que as medições, exceto as Folhas de atividades, ocorreram após a realização de todas as aulas de cada módulo. Os questionários estão apresentados na íntegra no Anexo II.

Tabela 37. Planejamento da coleta de dados para avaliação da UI.

Módulo	**Momento da coleta de dados**	**Instrumentos de medição**
Módulo 1 - Motivação e conceitos básicos com Code.org	Final do Módulo	Status de conclusão da lição no Code.org
		Questionário pós-módulo
Módulo 2.A - Desenvolver um jogo com SCRATCH:	Durante o Módulo (Durante as aulas 1, 2 e 3)	Rubrica para avaliar as Folhas de atividades

UNidade Instrucional Interdisciplinar de Computação e HistóriA	Final do Módulo	Dr. Scratch
		Rubrica para avaliar apresentação do jogo feito no Scratch
		Questionário pós-módulo
Módulo 2.B1 - Desenvolver um aplicativo com App Inventor: Programação do Ecopontos+	Final do Módulo	Questionário pós-módulo
Módulo 2.B2 - Desenvolver o seu aplicativo com App Inventor: Programação, *Design Thinking, Design* de Interface	Durante o Módulo (Durante as aulas 1, 2, 3 e 8)	Rubrica para avaliar as Folhas de atividades
	Final do Módulo	CodeMaster v2.0 (pensamento computacional e UI design)
		Questionário pós-módulo
Módulo 3 - Aspectos pedagógicos no ensino de computação	Final do Módulo	Questionário pós-módulo
Módulo 4 - Aspectos tecnológicos no ensino de computação	Final do Módulo	Questionário pós-módulo

Fonte: elaborada pela autora.

O projeto foi aprovado pela escola e pela CEPSH - Comitê de Ética em Pesquisa com Seres Humanos da UFSC com o número do parecer 3.338.481.

5.2 APLICAÇÃO DA UNIDADE INSTRUCIONAL

A unidade instrucional foi aplicada por pesquisadores da iniciativa Computação na Escola/INCoD/INE/UFSC no segundo semestre de 2019 e primeiro semestre de 2020 em um curso de professores da Educação Básica. O projeto envolveu sete professores de diferentes disciplinas e anos escolares de uma escola privada de Florianópolis/SC durante aproximadamente 3 meses, em encontros semanais de quatro horas na própria escola. Os professores foram selecionados e convidados pela escola, os quais participaram de maneira voluntária no período que não lecionam.

A sala utilizada não era informatizada, então os professores utilizaram notebooks

próprios ou da escola, com conexão à internet por meio de uma rede Wi-fi. A coleta de dados ocorreu nas primeiras quatro semanas da UI, cobrindo os Módulos 1 e 2.A. A Tabela 38 mostra a quantidade dos dados coletados durante a aplicação.

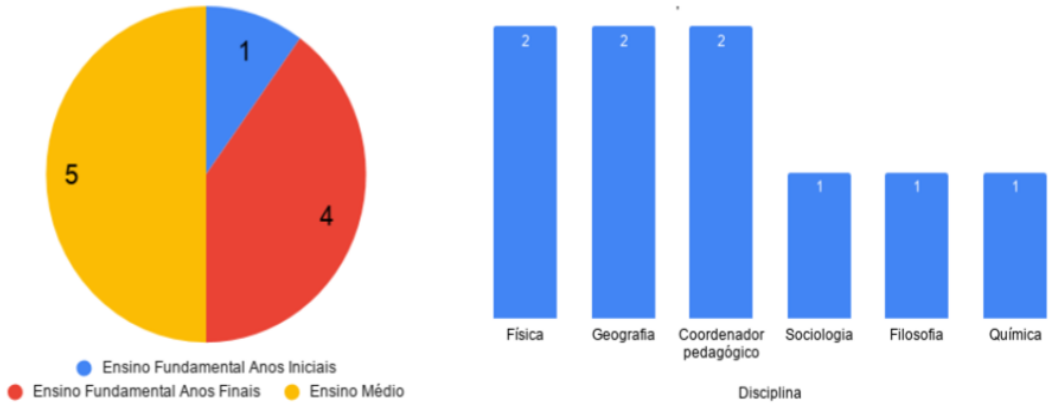
Tabela 38. Quantidade de respostas para cada questionário.

Questionário	**Quantidade**
Quantidade de questionário pós-módulo 1 respondidos	7
Quantidade de questionário pós-módulo 2.A respondidos	6

Fonte: elaborada pela autora.

Alguns dos professores lecionam em mais de uma disciplina ou em mais de um nível escolar (Figura 37). Por isso, a soma do número de professores no gráfico é maior que o número de questionários respondidos.

Figura 37. Nível escolar e disciplinas lecionadas pelos professores.



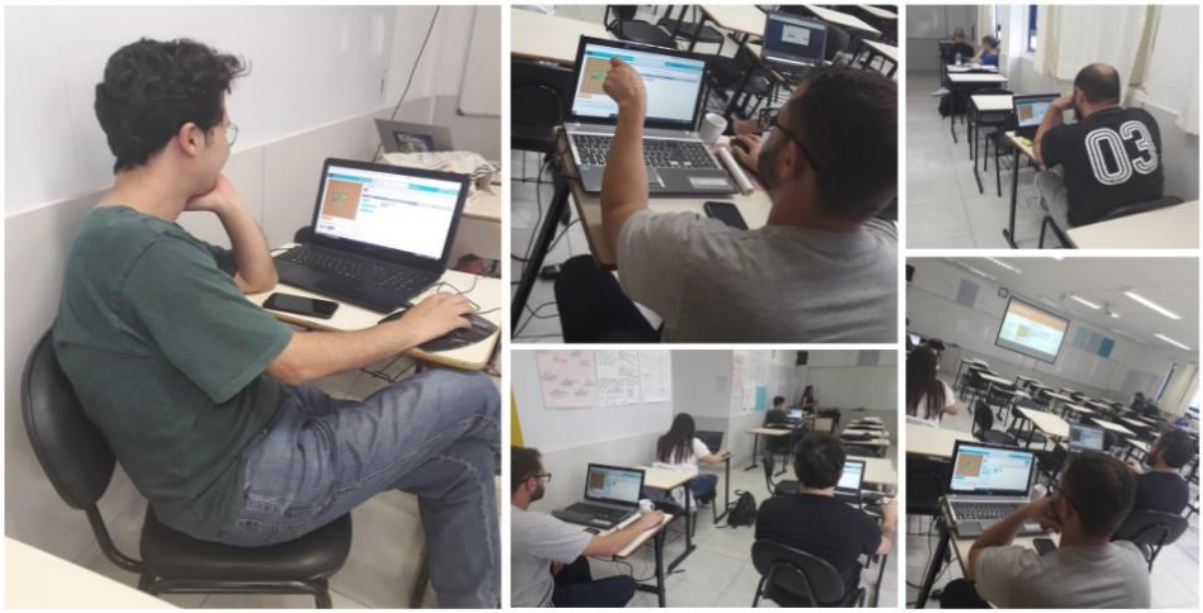
Fonte: elaborada pela autora.

No contexto deste trabalho foram aplicados e avaliados os Módulos 1 e 2.A. Isso ocorreu por restrições de tempo, abrangência da unidade instrucional e pelo grande número de encontros necessários para completar todos as aulas. Mas a unidade continuará mesmo após a entrega do presente trabalho.

O primeiro encontro foi realizada a apresentação de alguns slides, introduzindo conceitos básicos de computação, começando com o conhecimento de algoritmos. Também foram mostrados alguns vídeos para ensinar as regras do jogo *SplashCode*, assim com o intuito de fixar o conceito de algoritmo e os professores jogaram este jogo. Neste encontro os professores tiveram a oportunidade de conhecer o Code.org

(CODE.ORG, 2019) e realizar o Labirinto Clássico que faz parte da Hora do Código (*https://studio.code.org/hoc/1*). Ao final do encontro os professores preencheram o questionário pós-módulo.

Figura 38. Primeiro encontro da unidade instrucional.



Fonte: elaborada pela autora.

O segundo encontro proporcionou aos professores o primeiro contato com o ambiente de programação Scratch, no qual puderam conhecer as partes da interface de usuário do ambiente. Os professores programaram algumas ações utilizando os principais tipos de blocos do Scratch, juntamente com alguns conceitos básicos de algoritmos e programação (Figura 38). Neste encontro os professores também tiveram a oportunidade de desenvolver um jogo, utilizando um tutorial como base. O final da aula foi proposto aos professores pensarem em um tema para programar seu próprio jogo no terceiro encontro.

O terceiro encontro (Figura 39) consistiu no projeto e implementação de um jogo no Scratch. Nesse encontro os professores foram auxiliados na construção de seus jogos, com ajuda dos instrutores e do material didático produzido no âmbito deste projeto. No quarto encontro alguns professores apresentaram seus projetos, enquanto outros tiveram maiores dificuldades e aproveitaram o início da aula para fazer algumas perguntas e pedir ajuda nas partes do projeto que não conseguiram terminar. Esses professores que pediram ajuda apresentaram no início do quinto encontro. Na metade

final do quarto encontro foi começado as aulas do Módulo 2.B1, referentes ao App Inventor, a qual sua avaliação não está no escopo deste trabalho.

Figura 39. Segundo e terceiro encontros da unidade instrucional.

Fonte: elaborada pela autora.

5.3 ANÁLISE DOS DADOS

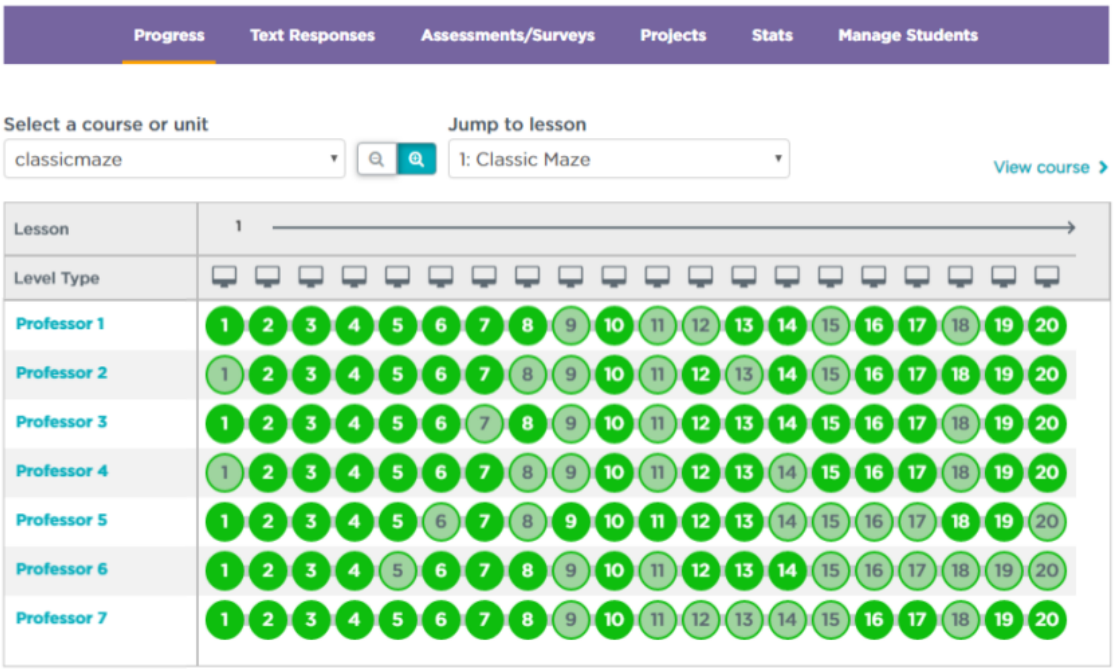
De acordo com as perguntas de pesquisa, os dados são analisados a fim de respondê-las. As seções a seguir contém a análise dos dados referente a cada um dos módulos avaliados (Módulo 1 e Módulo 2.A). Os demais módulo não foram avaliados por este trabalho.

5.3.1 A UI proporciona aprendizagem de competências em relação aos conteúdos de computação? (PA1)

Módulo 1 - Motivação e conceitos básicos com Code.org

O status de conclusão da lição em consequência da realização de atividades práticas no Code.org, demonstrou ótimos resultados em relação ao código produzido pelos professores (Figura 40). Os professores fizeram todas as atividades do Labirinto Clássico (CODE.ORG, 2019). Apesar de a maioria dos professores ter informado que não se sentem capazes de desenvolver um programa de computador vários dos níveis foram concluídos de forma perfeita (verde escuro) por todos os professores. Os níveis que estão em verde claro representam que o nível foi concluído, mas foram resolvidos com mais blocos que o necessário. Os professores resolveram de 4 a 9 níveis com o status verde claro.

Figura 40. *Status* de conclusão das lições no Code.org.

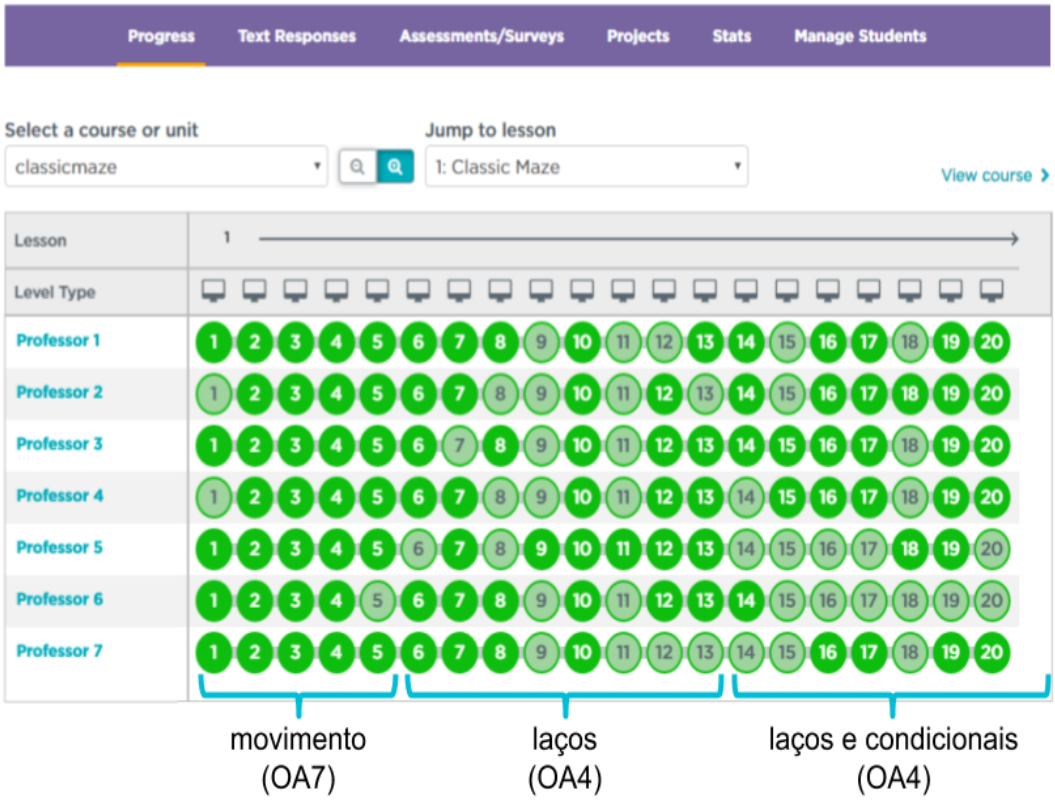


Fonte: CODE.ORG (2019).

Pode-se observar na Figura 40 os níveis 9 e 11 eram os mais difíceis, pois a maioria dos professores (6) não os completou “perfeitamente”. Um possível motivo para a dificuldade dos níveis 9 e 11, encontra-se na descoberta de um padrão no caminho correto no nível 9. Esse nível é o primeiro que o professor precisa descobrir um “padrão” e utilizar o conceito de laços, mostrando que toda vez que um conceito mais complexo é apresentado o professor sentiu dificuldade para compreendê-lo na primeira vez. O nível 11 apresenta o mesmo problema do nível 9, mas com localização diferente. Uma evidência de que os professores têm dificuldades na primeira vez que aprendem um

conceito complexo está nos níveis 15 e 18, os quais também não foram completados “perfeitamente” por muitos professores (5). O nível 15 é o primeiro que pede para o aluno usar o conceito de condicionais e o nível 18 é o primeiro que mostra o conceito de condicional composto. Mesmo com essa dificuldade, vários professores afirmaram que sabem explicar para os alunos como fazer programas de computador. Então, detectou-se a importância de trabalhar de diferentes maneiras o mesmo conceito para que o aprendiz tenha várias oportunidades para entendê-lo e não desista de aprender um novo conceito (Figura 41).

Figura 41. Objetivos de aprendizagem dos conceitos relacionados as atividades do Code.org.



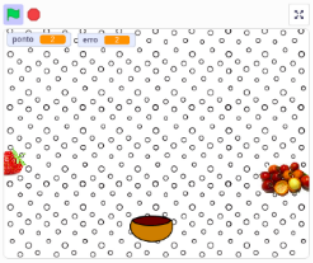
Fonte: CODE.ORG (2019).

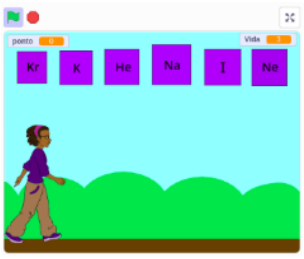
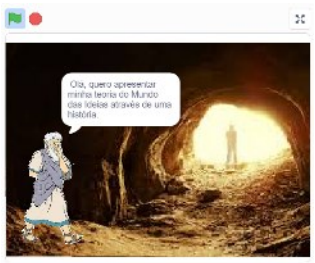
Módulo 2.A – Desenvolver um jogo Scratch

Os professores afirmaram no questionário pós-módulo que conseguem fazer um programa de computador e demonstraram, de forma geral, que realmente aprenderam os conceitos apresentados durante as aulas deste módulo. Os conceitos de computação apreendidos foram utilizados pelos professores para o desenvolvimento de jogos. Os temas dos jogos que os professores escolheram foram relacionadas com as disciplinas que lecionam. No total, os professores desenvolveram 7 jogos completos.

As características dos jogos desenvolvidos são encontradas na Tabela 39.

Tabela 39. Jogos desenvolvidos pelos professores.

Disciplina do professor	**Nome/ Características do jogo**	**Imagem**	***Link* para o jogo**
Física	**Quiz - Planeta Terra:** quiz para promover a reflexão sobre características do planeta Terra		https://scratch.mit.edu/projects/339967002
Física	**Ionizante:** quiz sobre os tipos de radiação ionizante e suas aplicações no nosso cotidiano		https://scratch.mit.edu/projects/339033318
História e coordenador pedagógico	**RevoluQuiz:** quiz para os usuários entrarem em contato com os aspectos mais relevantes sobre o fim do Império Russo e a ascensão do movimento revolucionário		https://scratch.mit.edu/projects/338049189/
Coordenador Pedagógico	**Mitos medievais:** quiz sobre mitos na Idade Média		https://scratch.mit.edu/projects/338095682/
Geografia	**Frute-se:** jogo para coletar as frutas da Região Sul do Brasil		https://scratch.mit.edu/projects/338057189

Química	**Keep Walking dos elementos Nobres:** controle de caminhada de um personagem que deve acertar os blocos de elementos gases nobres		https://scratch.mit.edu/projects/334947616
Filosofia e sociologia	**O Mito da Caverna do Platão:** história com a explicação do Mito da Caverna de Platão		https://scratch.mit.edu/projects/339968345

Fonte: elaborada pela autora.

Realizando uma avaliação de desempenho com base nos jogos criados utilizando Dr. Scratch é possível perceber que os conceitos apresentados nas aulas foram devidamente aplicados nos jogos desenvolvidos pelos professores (Tabela 40). Analisando o resultado, compreende-se que os itens os quais obtiveram pontuações maiores são os itens relacionados aos objetivos de aprendizagem, consequentemente ensinados durante a aplicação e por este motivo foram incorporados aos jogos. Com destaque para o item lógica, o qual dois alunos conseguiram obter a pontuação máxima. A avaliação desse item está relacionada com o aprendizado de conceitos definidos nos objetivos de aprendizagem, especialmente condicionais e operadores. Como já se esperava, o item paralelismo teve pontuações muito baixas, pois esse conceito não é um objetivo de aprendizagem deste módulo. Nos demais itens, os professores conseguiram boas pontuações. Esses itens estão relacionados aos objetivos de aprendizagem que envolve os conceitos de sensores e movimentos, avaliados no item interatividade com o usuário; variáveis (item representação de dados), laços (item controle de fluxo), eventos (item sincronização) e abstração (item abstração). Então, as pontuações apresentadas para cada item demonstram que os professores conseguiram compreender os conceitos que lhes foram ensinados e por este motivo foram capazes de aplicar em seus jogos.

Tabela 40. Avaliação do desempenho utilizando a ferramenta Dr. Scratch para avaliação de jogos do Scratch na escala de pontuação [0..3].

	Lógica	Paralelismo	Interatividade com o usuário	Representação de dados	Controle de fluxo	Sincronização	Abstração	**Total**
Quiz - Planeta Terra	2	1	2	2	2	2	2	**13**
Ionizante	2	0	2	2	1	2	2	**11**
Revolu Quiz	3	0	2	1	2	2	2	**12**
Mitos medievais	2	0	2	2	1	1	2	**10**
Frute-se	1	1	2	2	2	2	1	**11**
Keep Walking dos Elementos Nobres	3	1	2	2	2	1	3	**14**
O Mito das Cavernas do Platão	0	3	2	2	2	3	2	14

Fonte: elaborada pela autora.

A avaliação dos artefatos criados como resultado do processo de desenvolvimentos de jogos documentados em Folhas de Atividades, utilizando por meio da rubrica da Tabela 3, apresentou ótimos resultados em relação aos objetivos de aprendizagem deste módulo (Tabela 41). A média das pontuações da análise das Folhas de Atividade foi 10, mostrando que todos os professores souberam detalhar a ideia do jogo que quiseram desenvolver. Assim, essa avaliação demonstrou que os objetivos de aprendizagem do módulo foram atingidos, pois os professores identificaram e resolveram problemas criando sistemas de software e também conseguiram desenvolver artefatos computacionais, neste caso jogos.

Tabela 41. Avaliação do desempenho na produção das Folhas de Atividades baseada na rubrica dos artefatos textuais produzidos na implementação de um jogo Scratch.

Jogos tipo Quiz							
	Atores [0..1]	Objetivo [0..1]	Interatividade [0..1]	Regras [0..2]	Derrota [0..1]	Perguntas [0..4]	**Total**
Quiz - Planeta Terra	1	1	1	1,5	1	4	**9,5**

Ionizante	1	1	1	2	1	4	**10**
RevoluQuiz	1	1	1	2	1	4	**10**
Mitos medievais*	-	-	-	-	-	-	**-**
Jogos tipo Ação							
	Atores [0..1]	Objetivo [0..2]	Interatividade [0..2]	Regras [0..3]	Derrota [0..2]	**Total**	
Frute-se	1	2	2	3	2	**10**	
Keep Walking dos Elementos Nobres	1	2	2	3	2	**10**	
Jogos tipo Aventura							
	Atores [0..3]	Objetivo [0..1]	Interatividade [0..2]	Sequência da história [0..4]		**Total**	
O Mito das Cavernas do Platão	3	1	2	4		**10**	

*Não entregou a Folha de atividade
Fonte: elaborada pela autora.

Os professores também tiveram que elaborar uma apresentação explicando como utilizaram os blocos de comando para transformar as ideias em um jogo. As apresentações tinham a intenção de verificar se os professores aprenderam todos os objetivos de aprendizagem desse módulo. Os professores apresentaram os jogos foram realizadas sem dificuldades, possuindo uma média de 9,4 (Tabela 42). Um dos professores demonstrou dificuldade na compreensão dos blocos de programação do projeto e não conseguiu responder com clareza qual conceito teve mais dificuldade de compreender. Mas, em geral, os professores demonstraram compreensão dos blocos de comandos utilizados na programação e souberam explicar os aspectos fáceis e difíceis de programar. As maiores dificuldades apontadas pelos professores foi entender operadores lógicos e condicionais, entender a execução de vários blocos separados e desenvolver um código para automatizar uma rotina repetitiva, a qual foi citada por dois professores. Mesmo com as dificuldades apontadas pelos professores, todos afirmaram que conseguem explicar aos alunos como fazer um programa de computador.

Tabela 42. Avaliação do desempenho na apresentação do jogo baseada na rubrica para apresentação de jogos do Scratch na escala de pontuação [0..5].

	Bloco de comandos	Identificação de aspectos fáceis e difíceis	**Pontuação total**
Quiz - Planeta Terra	5	5	**10**
Ionizante	5	5	**10**
RevoluQuiz	5	5	**10**
Mitos Medievais	5	5	**10**
Frute-se	3,5	3,5	**7**
Keep Walking dos Elementos Nobres	5	5	**10**
O Mito das Cavernas do Platão	5	5	**10**

Fonte: elaborada pela autora.

5.3.2 A UI promove uma experiência de aprendizagem agradável e divertida? (PA2)

Módulo 1 - Motivação e conceitos básicos com Code.org

Os comentários dos professores indicam que suas experiências de aprendizagem nesse módulo foram positivas, com a maioria dos professores sabendo explicar a outras pessoas como fazer um programa de computador. As atividades de introdução a programação fizeram as aulas do Módulo 1 proporcionarem uma aprendizagem agradável, segundo três professores (Tabela 43). Um professor citou a abordagem dinâmica e interativa, a qual fez a aula prática e rápida, como informou outro professor.

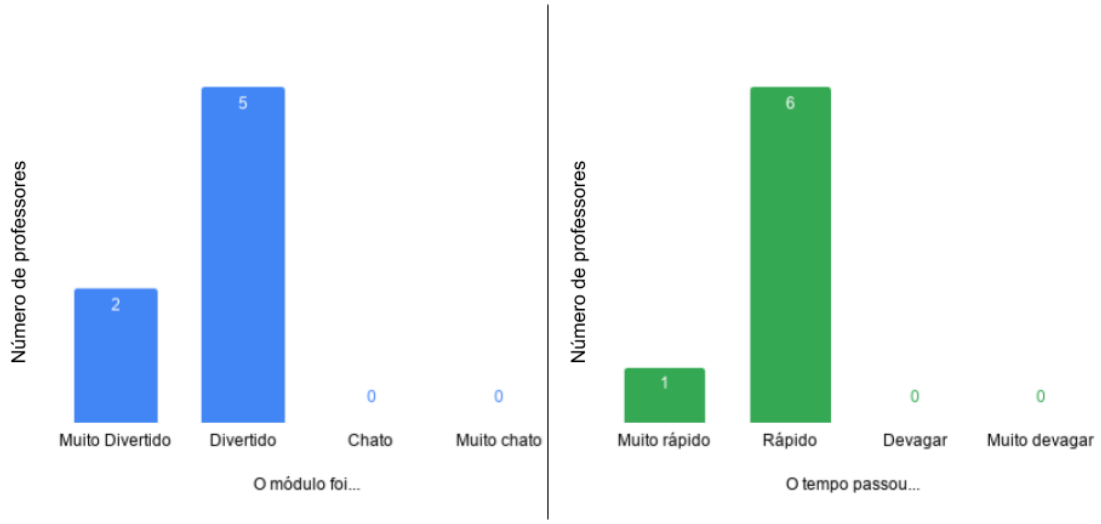
Tabela 43. Respostas da Pergunta “O que você achou desse módulo?”.

Prático e rápido.
Achei interessante essa introdução de programação.
Divertido e lúdico; as atividades introdutórias foram bem “concretas” para ilustrar o que é programação.
Excelente introdução.
Achei a forma de abordagem dinâmica e interativa. Chamou minha atenção a possibilidade de aplicação com diferentes faixas etárias.

Fonte: elaborada pela autora.

A percepção de praticidade e rapidez, citada por um professor, é reforçada nas respostas de múltipla escolha do questionário, quando todos os professores informam que acharam o módulo divertido ou muito divertido (Figura 42). E também quando as respostas indicam que o tempo das aulas do módulo passou rápido ou muito rápido, indicando que os professores ficaram focados no conteúdo e não notaram a passagem do tempo (Figura 42).

Figura 42. Resposta dos professores quanto a diversão e rapidez das aulas do Módulo 1.



Fonte: elaborada pela autora.

Módulo 2.A – Desenvolver um jogo Scratch

A experiência de aprendizagem dos conteúdos deste módulo foi considerada positiva pelos professores, conforme respostas obtidas no questionário pós-módulo (Tabela 44). De modo geral, os professores acharam os conteúdos agradáveis, com destaque para a ferramenta ensinada. Alguns professores citaram a programação e a oportunidades de experimentar novas possibilidades como aspectos importantes do módulo. Um professor também destacou a oportunidade de utilizar a lógica de programação, o que levou a sair do papel de um usuário para o papel de um programador de *software*.

Tabela 44. Respostas da Pergunta “O que você achou desse módulo?”.

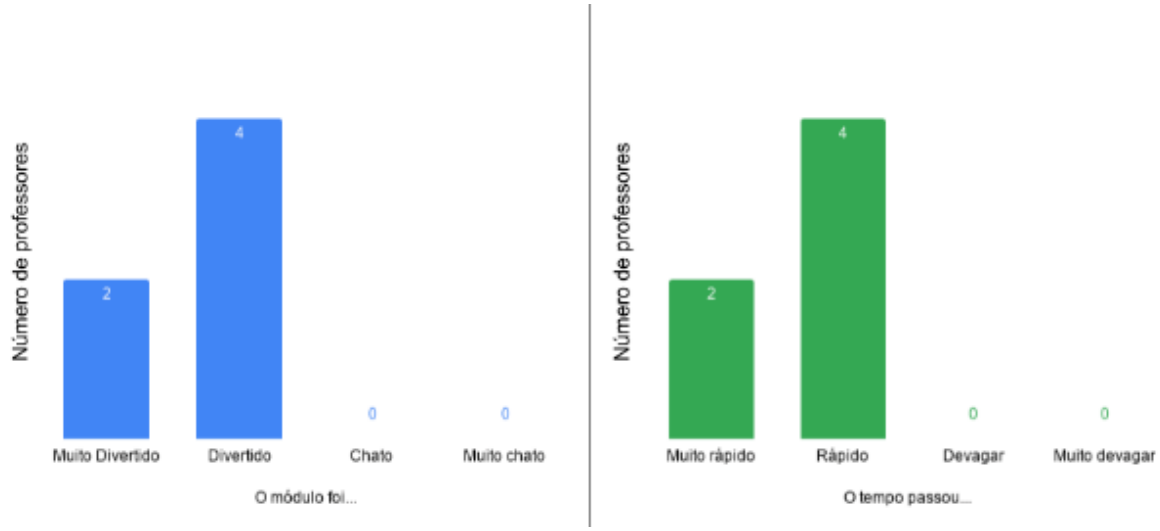
Opinião
Muito interessante. Várias ferramentas e possibilidades foram contempladas neste módulo.

Gostei com ressalvas; achei interessante a abordagem pós *Code.org* e o trabalho final do projeto do jogo. Mas senti falta de tarefas menores (intermediárias) para melhor familiarização com a plataforma.
Gostei bastante da proposta da plataforma.
Interessante e divertido na prática.
Achei muito interessante e, me deu uma ideia de como programar.
Foi um módulo mais “maduro”. Apesar de ter o *layout* colorido e semelhante aos anterior desta vez saímos do papel do “jogador” e adentramos na lógica do “produtor” de uma ferramenta. Gostaria de mais tempo para experimentar as possibilidades.

Fonte: elaborada pela autora.

Os professores também demonstraram satisfação significativa, os quais informam que acharam o módulo divertido ou muito divertido (Figura 43). Os professores demonstraram satisfação significativa, os quais, acharam divertido ou muito divertido aprender os conteúdos. As respostas dos questionários também mostram que os professores não sentiram o tempo passar, indicando que ficaram focados no conteúdo. Dessa forma, a experiência de aprendizagem pode ser considerada positiva.

Figura 43. Resposta dos professores quanto a diversão e rapidez das aulas do Módulo 2.A.



Fonte: elaborada pela autora.

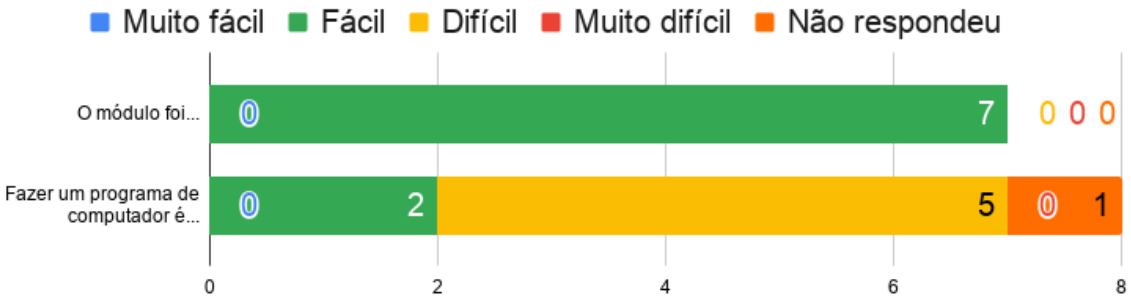
5.3.3 A UI facilita a aprendizagem (usabilidade da unidade instrucional)? (PA3)

Módulo 1 - Motivação e conceitos básicos com Code.org

Os professores acharam os tópicos ensinados no Módulo 1 fáceis de aprender (Figura

44). Porém a maioria achou difícil fazer um programa de computador. A pergunta “Fazer um programa de computador é” apresenta 8 respostas pois um dos professores informou duas opções (Fácil e Difícil) no questionário. Então pode ser que esse professor achou que fazer programa de computador possui algumas partes fáceis e outras difíceis ou um erro no preenchimento dessa questão no questionário.

Figura 44. Facilidade de aprendizagem referente ao módulo e a fazer programa de computador.



Fonte: elaborada pela autora.

A qualidade, no geral, desse módulo foi excelente, com apenas um professor informando a resposta bom. Os pontos positivos citados pelos professores reforçam que as atividades práticas de programação foram o ponto forte desse módulo. Além de praticar a programação, o ensino de algoritmos utilizando o jogo SplashCode também foi citado por um professor. Teve ainda quem citasse a interatividade da aula, mostrando que a instrutora conseguiu chamar a atenção da turma. Quanto aos pontos negativos, a maioria não respondeu ou disse que não tinha nada a reclamar. Porém foi citado o exercício da pizza e do lixo, além da parte introdutória, a qual um professor achou muito longa. Também se observou que os professores gostaram mais da parte prática do que da parte introdutória. Embora na pergunta de análise PA2 tenha-se concluído que a introdução a programação proporcionou experiência agradável e divertida.

Tabela 45. Respostas da Pergunta “O que mais gostei no módulo foi”.

As partes práticas.
A flexibilidade e paciência das “profs”. Conhecer sites e ferramentas de fácil manuseio.
A atividade do labirinto.
O que eu mais gostei foi praticar a programação.

A interatividade.
Questões práticas.
Jogo “manual” *SplashCode*.

Fonte: elaborada pela autora.

Tabela 46. Respostas da Pergunta “O que menos gostei no módulo foi”.

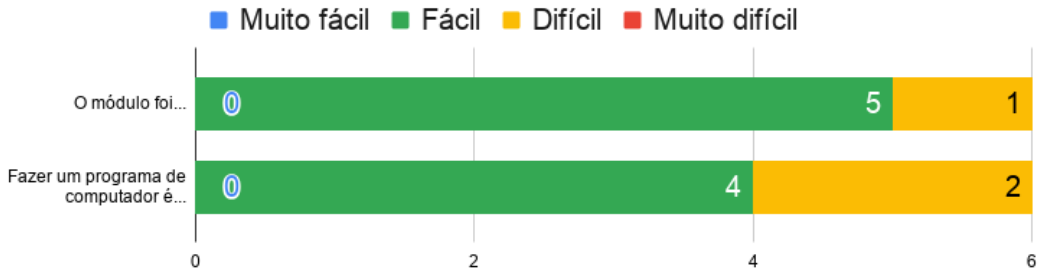
Parte introdutória (justificativa de programação muito longa porque ensinar programação muito longa).
Os jogos da pizza e do lixo.

Fonte: elaborada pela autora.

Módulo 2.A – Desenvolver um jogo com Scratch

Por meio dos questionários pós-módulo, os professores avaliaram o nível de facilidade deste módulo. De modo geral, a maioria dos professores, com exceção de um que achou o módulo difícil, considerou que os conteúdos do módulo foram fáceis de aprender. A dificuldade informada por um professor pode estar relacionada o pouco tempo para experimentação da ferramenta Scratch. Alguns professores (2) também acham que fazer um programa de computador é difícil, porém quatro professores afirmaram que é fácil fazer um programa (Figura 2). Essas respostas também podem estar associadas ao pouco tempo para experimentação da ferramenta, em conjunto com as dificuldades dos professores realizarem um projeto de jogo.

Figura 45. Frequência de respostas referente a facilidade de aprendizagem referente ao módulo e a fazer programa de computador.



Fonte: elaborada pela autora.

O modo como este módulo foi projetado facilita a aprendizagem, pois segundo a maioria dos professores (5) a qualidade foi excelente, somente um professor afirmando que a qualidade foi boa. Os pontos positivos citados, demonstram que os professores gostam de atividades práticas (Tabela 47). Com os professores respondendo que os pontos fortes deste módulo foram a possibilidade de testar os comandos no

Scratch, execução prática do jogo, o desenvolvimento do trabalho final, o aprendizado de programar uma rotina e a lógica dos movimentos programados. Outros pontos positivos citados foram a possibilidades de encontrar modelos de jogos e como a ferramenta foi apresentada. Quanto aos pontos negativos (Tabela 48), foi citada por dois professores foi o fato das dificuldades que os professores encontraram para criar um jogo. Um professor achou que houve pouco tempo para a experimentação necessitando de mais aulas. Outro achou que deveriam ter tarefas para fazer entre uma aula e outra, embora um professor tenha informado que o que menos gostou foram os exercícios de fixação.

Tabela 47. Respostas da Pergunta “O que mais gostei no módulo foi”.

Opinião
O *layout* do programa, a paciência das “profs”, a possibilidade de modelos.
A forma com que o professor apresentou as ferramentas.
O desenvolvimento do trabalho final.
A possibilidade de testar os comandos principais da plataforma.
A execução prática do jogo, pensar na lógica dos movimentos e interface.
Gostei de aprender a rotina para programar no programa.

Fonte: elaborada pela autora.

Tabela 48. Respostas da Pergunta “O que menos gostei no módulo foi”.

Opinião
Foi tudo muito interessante.
Pouco tempo para experimentação. Exercícios de fixação do “passo-a-passo”.
Nada a reclamar.
Falta de tarefas para semana.
Como estive ausente no módulo de criação, tive muitas dificuldades de criar um jogo sozinho.
Não achei nada maçante, porém algumas dificuldades ocorreram na lógica do projeto, mas rapidamente sanadas pelo “prof”.

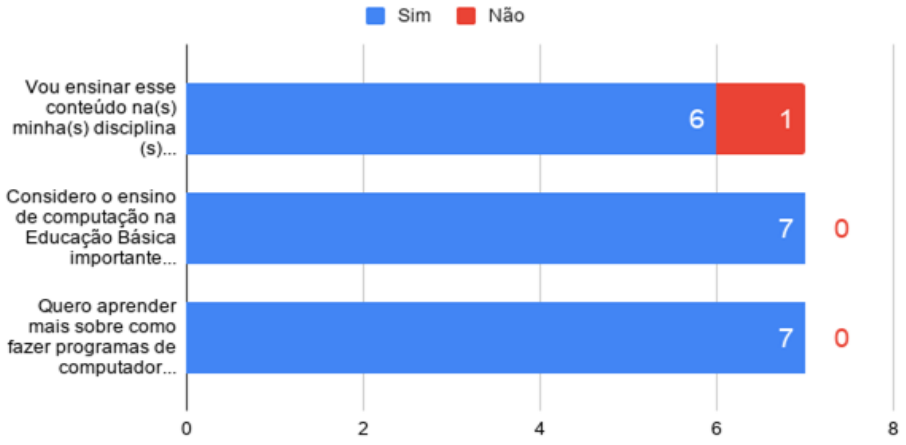
Fonte: elaborada pela autora.

5.3.4 A UI proporciona o interesse dos professores para ensinar conteúdos de computação em suas disciplinas? (PA4)

Módulo 1 - Motivação e conceitos básicos com Code.org

A maioria dos professores relataram que se divertiram durante as atividades do módulo e citaram que o tempo da aula passou com rapidez. Essa experiência positiva se tornou evidente na intenção dos professores em ensinar computação, na qual seis professores declararam que irão ensinar os conteúdos nas suas disciplinas (Figura 44). As aulas do Módulo 1 também tinham a intenção de mostrar a importância do ensino de computação na Educação Básica, e todos os professores entenderam a relevância da computação no dia-a-dia dos alunos (Figura 46).

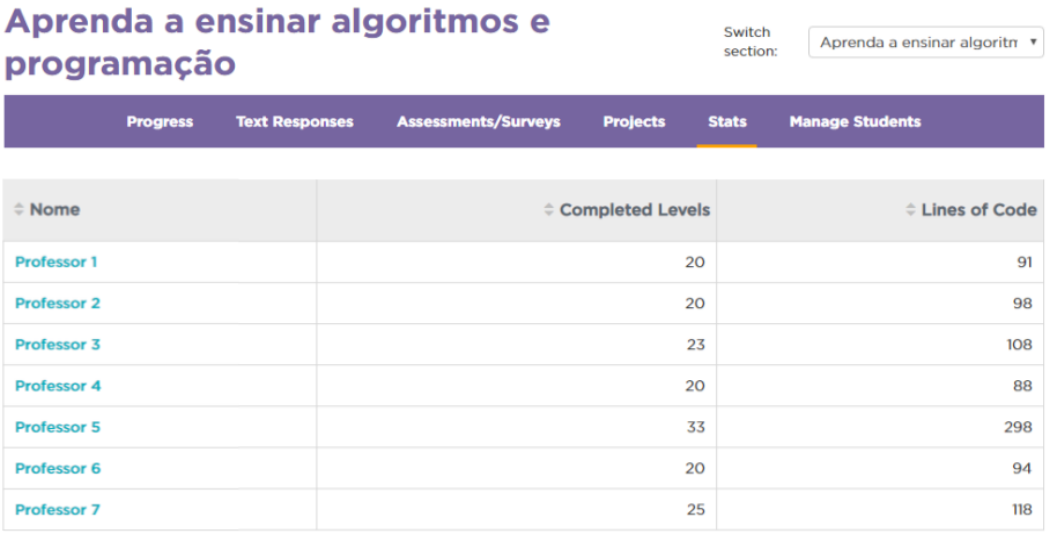
Figura 46. Distribuição da frequência de itens relacionados ao interesse na aprendizagem.



Fonte: elaborada pela autora.

A forma como o conteúdo desse módulo foi organizada também fez todos os professores mostrar interesse em aprender mais sobre programas de computador (Figura 47), sendo que um deles colocou como comentário adicional que gostaria de se aprofundar mais. É possível perceber esse fator na Figura 47, a qual é um *print* da tela do Code.org um dia após o encontro que foi ensinado todas as aulas desse módulo. A atividade do Labirinto Clássico realizada no primeiro encontro possui 20 níveis, e na Figura 45, três professores possuem números maiores que 20 na coluna de níveis completados, significando que após o término do encontro alguns professores procuraram fazer outras atividades propostas pelo Code.org.

Figura 47. Estatísticas de conclusão das lições no *Code.org.*



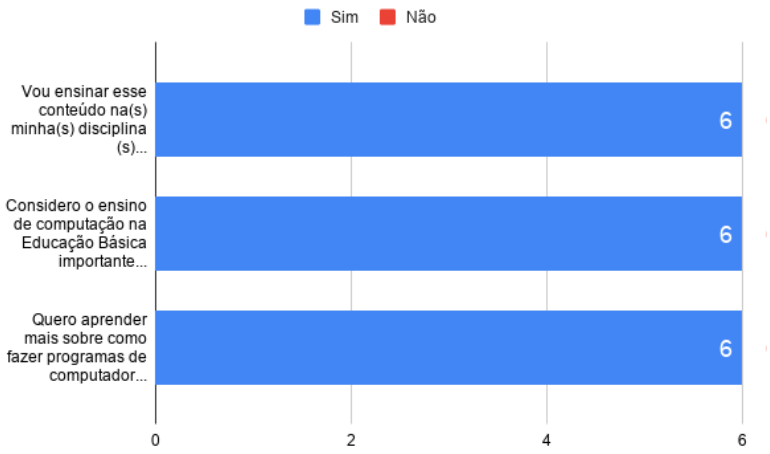
Nome	Completed Levels	Lines of Code
Professor 1	20	91
Professor 2	20	98
Professor 3	23	108
Professor 4	20	88
Professor 5	33	298
Professor 6	20	94
Professor 7	25	118

Fonte: CODE.ORG (2019).

Módulo 2.A – Desenvolver um jogo com o Scratch

Os professores afirmaram que sabem explicar aos alunos como fazer um programa de computador, e ao final deste módulo também demonstraram um interesse em aprender mais sobre como fazer um programa de computador (Figura 48). Além disso, os professores também afirmaram que é importante ensinar computação na Educação Básica, e todos querem, aplicar os conteúdos aprendidos nas disciplinas que lecionam.

Figura 48. Distribuição da frequência de itens relacionados ao interesse na aprendizagem.



Fonte: elaborada pela autora.

O modo como o módulo foi organizado permitiu que alguns professores entendessem que a computação pode ser utilizada para ensinar conteúdos disciplinares. Um professor comentou que a computação pode ajudar os alunos a entenderem a lógica utilizada nas ciências exatas. Enquanto outro indicou que a organização de pensamentos e a lógica utilizada na computação podem ser utilizados para estudos em geral. Dessa forma, pode-se concluir que os professores acham interessante dos professores para ensinar conteúdos de computação em suas disciplinas.

5.4 DISCUSSÃO

Os resultados da análise dos dados coletados durante a aplicação demonstram de forma inicial que a unidade instrucional proposta neste trabalho permite a professores da Educação Básica aprender conceitos de programação. Observou-se também especialmente em relação ao Módulo 2.A que a UI permite que aos professores a transferência de conhecimento aos seus respectivos domínios de conhecimento criando jogos para as suas disciplinas. Na apresentação dos jogos desenvolvidos, os professores demonstraram compreensão dos blocos de comandos utilizados e souberam explicar os conceitos fáceis e difíceis. Assim, percebeu-se que os professores aprenderam os conteúdos de computação e podem ensinar o conteúdo aprendido.

A utilização das ferramentas de programação baseada em blocos, o Code.org e Scratch, também contribuíram na experiência de aprendizagem e no aprendizado de programação. Mesmo com as dificuldades encontradas pelos professores no Módulo 2.A, a utilização da ferramenta Scratch permitiu aos professores o desenvolvimento de programas de maneira mais fácil. Os resultados evidenciam que houve uma aprendizagem expressiva em programação em ambos os módulos.

Os professores avaliaram que os conteúdos do módulo foram fáceis de aprender, esse fator pode ser influência da utilização das ferramentas Code.org e Scratch e também da estratégia instrucional. Por mais que a maioria dos conceitos ensinados tenham um grau de complexidade elevado, os participantes das aplicações sentiram um nível adequado de aprendizagem. Porém, no Módulo 1 os professores informaram que acham difícil fazer um programa de computador, apesar de todos ter realizados as

atividades do Labirinto Clássico no Code.org. Esta opinião pode ter ocorrido pelo fato dos professores não conhecerem conteúdos de computação e assim não se sentiram confiantes para realizar este tipo de atividade. Pois no decorrer da UI, no Módulo 2.A o número de professores que acham fácil fazer um programa de computador, aumentou consideravelmente.

Além da programação, a unidade instrucional contribuiu para ensinar outras competências relacionadas a computação. No Módulo 2.A os professores seguiram um processo de desenvolvimento jogos e adotaram práticas de Engenharia de Software. Os professores também utilizaram os conhecimentos que possuíam nos conteúdos de suas disciplinas e desenvolveram um jogo interdisciplinar. Esta abordagem incentivou a criatividade e a imaginação dos professores, para desenvolver soluções computacionais que possam ser usadas nas respectivas disciplinas. Embora alguns professores não tenham conseguido finalizar o desenvolvimento do seu jogo até o fechamento deste trabalho, o desenvolvimento dos jogos foi útil para que os professores conheceram novas alternativas de ensino integrando o ensino de computação de forma interdisciplinar.

Os professores consideraram a participação no curso uma experiência agradável e divertida, e se sentiram envolvidos nos conteúdos dos módulos 1 e 2.A ao perceber que as aulas passaram rápidas. Isto demonstra que a estratégia das aulas, as quais predominou o uso de atividades práticas, pode ter ajudado a manter os professores imersos no processo de aprendizagem. A estratégia instrucional utilizada também pode ser notada na motivação e interesse dos professores em aprender mais sobre os conceitos de computação. Embora alguns professores tenham afirmado nos questionários pós-módulo que as atividades as quais menos gostaram foram exercícios de algoritmos propostos.

A utilização de um processo sistemático abordando competências de computação estimulou os professores a se sentirem capazes de ajudar os seus alunos a desenvolverem diversas habilidades do século XXI, como por exemplo a criatividade, a resolução de problemas e o pensamento crítico. Assim, a unidade instrucional demonstrou resultados animadores em termos de aprendizagem de computação. A UI também impactou positivamente na motivação, na experiência e na aprendizagem, como também em empoderar os professores para ensinar computação de maneira

interdisciplinar nas disciplinas que lecionam.

5.4.1 Ameaças à validade

O estudo de caso desta pesquisa pode apresentar várias limitações em relação a validade dos resultados. Com o intuito de mitigar estas ameaças, foi definida e documentada uma metodologia sistemática para o estudo, usando a abordagem GQM (BASILI, CALDIEIRA & ROMBACH, 1994). Uma das ameaças se refere a dificuldade de medir aspectos diversão e satisfação, normalmente capturados por medidas subjetivas. Para contornar essa ameaça à validade, os itens dos questionários foram sistematicamente derivados com base no instrumento de medição padronizado dETECT (GRESSE VON WANGENHEIM *et al.*, 2019).

Outra ameaça é falta de um *benchmark*, testando o conhecimento dos alunos antes do curso, para fazer uma comparação e analisar as diferenças. Porém por se tratar de um curso de professores de outras diferentes disciplinas não foi feito um teste de conhecimento no início do curso.

As características do projeto representam outra ameaça, no qual só foi possível realizar um estudo de caso, em vez de um experimento. Além disso, o fato de a unidade instrucional ter sido aplicada apenas com um conjunto pequeno de participantes reduz tanto a representatividade quanto a possibilidade de generalização dos resultados. Entretanto, considerando a natureza exploratória da pesquisa neste momento, considera-se aceitável o rigor científico de um estudo de caso cuidadosamente definido.

6. CONCLUSÃO

O objetivo do presente trabalho foi o desenvolvimento de uma unidade instrucional para ensinar conceitos básicos de computação aos professores *in-service* da Educação Básica, e assim permitir a utilização desse conhecimento de forma interdisciplinar inserida em conteúdo programado do Ensino Fundamental. Com o intuito de atingir esse objetivo a unidade instrucional desenvolvida inclui prática de programação por meio de desenvolvimento de atividades fechadas de programação no Code.org, de jogos no Scratch e de apps no App Inventor. Também aborda aspectos pedagógicos referentes ao ensino da computação e também aspectos tecnológicos para possibilitar ao professor preparar a infraestrutura necessária para as aulas.

Como parte do presente trabalho foi realizada a análise da fundamentação teórica dos assuntos abordados, com a intenção de conhecer os principais conceitos envolvidos no trabalho (O1). A análise do estado da arte foi realizada por meio de um mapeamento sistemático buscando unidades instrucionais que tinham como objetivo ensinar computação a professores *in-service* utilizando linguagens de programação visual baseada em blocos (O3). O resultado dessa análise foi submetido como artigo científico:

KRETZER, F. K. GRESSE VON WANGENHEIM, C. HAUCK, J. PACHECO, F. Desenvolvimento Profissional de Professores para o Ensino de Computação na Educação Básica: Um Estudo de Mapeamento Sistemático Revista Brasileira de Informática na Educação, submetido (28 páginas).

Seguindo o método de design instrucional foi definido o sequenciamento do conteúdo e as estratégias de ensino da unidade instrucional (O4) com base na análise de contexto que identificou as características do público-alvo e infraestrutura (O2). Como consequência foram estabelecidos os objetivos de aprendizagem e o plano de ensino, para então os materiais didáticos serem produzidos (O5).

A unidade instrucional foi aplicada (O6) por meio de um estudo de caso exploratório envolvendo professores, com a maioria lecionando no Ensino Fundamental II e Ensino Médio. Como resultado do presente trabalho, é disponibilizada uma unidade instrucional para ensinar computação para professores por meio de um processo sistemático de desenvolvimento de jogos com Scratch e aplicativos utilizando o App Inventor, e integrando práticas pedagógicas e tecnológicas. Essa unidade pode ser

aplicada para professores da Educação Básica, com ou sem experiência em computação. Dessa forma, as escolas poderão utilizar o material didático para contribuir na formação de professores para ensinar computação de forma interdisciplinar. Com professores formados em computação, as escolas de Educação Básica poderão ensinar computação de forma mais abrangente e os professores também poderão utilizar o material didático referente ao ensino do Code.org, Scratch e App Inventor na unidade instrucional desenvolvida para aplicar em suas aulas.

Como trabalhos futuros, sugere-se a criação de unidades instrucionais com pequena duração, como oficinas de até três dias para que professores que não tem muito tempo disponível para fazer cursos longos, adaptando os materiais produzidos no presente trabalho. Visa-se também o desenvolvimento de um MOOC a partir da unidade instrucional desenvolvida. Pretende-se também replicar a aplicação e avaliação com mais professores da Educação Básica para ampliar os resultados referente a aplicação da unidade.

REFERÊNCIAS BIBLIOGRÁFICAS

ACM/IEEE. Computer Science Curricula – Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. New York, NY, EUA, ACM, 2013.

AIGA. Design thinking - An introduction to design thinking. 2008. Disponível em: <https://www.aiga.org/aiga/content/events-and-competitions/competitions/graphic-design-training-curriculum-for-high-school-teachers/>. Acesso em: abr. 2019.

ALIMISIS, D.; FRANGOU, S.; PAPANIKOLAOU K. A Constructivist Methodology for Teacher Training in Educational Robotics. In: the 9th IEEE International Conference on Advanced Learning Technologies, Riga, Letônia, 2009, pp. 25-28.

ALVES, N. da C. Desenvolvimento de uma unidade instrucional interdisciplinar para ensinar computação no Ensino Fundamental. Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) - Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2016.

ALVES, N. da C. CodeMaster: Um Modelo de Avaliação do Pensamento Computacional na Educação Básica através da Análise de Código de Linguagem de Programação Visual. Programa de Pós-Graduação em Ciência da Computação (PPGCC) – Universidade Federal de Santa Catarina, 2019.

ALVES, N. da C.; GRESSE VON WANGENHEIM, C.; HAUCK, J. C. R.; BORGATTO, A. F.; ANDRADE, D. F. Uma Análise das Diretrizes para Ensino de Pensamento Computacional propostas pela SBC na Educação Básica. Anais da X Reunião da ABAVE. São Paulo/SP, 2019.

ANDRADE, D. F.; TAVARES, H. R.; VALLE, R. da c. Teoria da Resposta ao Item: Conceitos e Aplicações. São Paulo: ABE — Associação Brasileira de Estatística, 4º SINAPE, 2000.

ANDRADE, L. A.; LIMA, M. G.; RODRIGUES, R. C.; FONTANA, V. S. Possibilidades da computação em nuvem como estratégia no processo de controle da produção - Estudo de caso com o software TINY ERP, 2018. Disponível em: <https://multivix.edu.br/wp-content/uploads/2018/08/possibilidades-da -computacao-em-nuvem-como-estrategia-no-processo-de-controle-da-producao-estudo-de-caso-com-o-software-tiny-erp.pdf>. Acesso em: Setembro de 2018.

BASILI, V.; CALDIERA, G.; ROMBACH, H. D. The Goal Question Metric Approach. Disponível em: <http://www.cs.umd.edu/~mvz/handouts/gqm.pdf>. Acesso em: maio de 2019.

BERGES, M.; HUBWIESER, P.; MAGENHEIM, J.; BENDER, E.; BROKER, K.; MARGARITIS-KOPECKI, M.; NEUGEBAUER, J.; SCHAPER, N.; SCHUBERT, S.; OHNDORF, L. Developing a competency model for teaching computer science in schools. In: Proceedings of the 18th ACM Conference on Innovation and technology in computer science education, Canterbury, Inglaterra, 2013, pp. 327-327.

BRANCH, R. M. Instructional Design: The ADDIE Approach. New York, New York, USA, Springer Science & Business Media, 2009.

BRASIL. Lei nº 9.394, de 20 de dezembro de 1996. Estabelece as diretrizes e bases da educação nacional. Diário Oficial da União, Brasília, 23 de dezembro de 1996. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/L9394.htm>. Acesso em: Setembro de 2018.

BRASIL. Decreto nº 8.752, de 10 de maio de 2016. Dispõe sobre a Política Nacional de Formação dos Profissionais da Educação Básica. Diário Oficial da União: Seção 1, Brasília, DF: Atos do Poder Executivo, n. 88, p. 5-6, 10 maio 2016.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canadá, 2012.

CAO, X.; HE, K. Web-Based Training Framework for China School Teachers Educational Technique Ability Training. In: 2009 First International Workshop on Education Technology and Computer Science, Wuhan, China, 2009, pp. 626-630.

CETIC. TIC EDUCAÇÃO - Pesquisa Sobre o Uso das Tecnologias de Informação e Comunicação nas Escolas Brasileiras, 2017. Disponível em: <https://cetic.br/media/docs/publicacoes/2/tic_edu_2017_livro_eletronico.pdf>. Acesso em: Agosto de 2018.

CETIC. TIC EDUCAÇÃO - Pesquisa Sobre o Uso das Tecnologias de Informação e Comunicação nas Escolas Brasileiras, 2016. Disponível em: <https://cetic.br/media/docs/publicacoes/2/TIC_EDU_2016_LivroEletronico.pdf>. Acesso em: Agosto de 2018.

CHO, S.; PAUCA, V. P.; JOHNSON, D.; JAMES Y. V. Computational Thinking for the Rest of Us: A Liberal Arts Approach to Engaging Middle and High School Teachers with Computer Science Students. In: Proceedings of the Society for Information Technology & Teacher Education International Conference, 2014, Jacksonville, EUA, pp. 79-86.

CODE.ORG. Disponível em: <https://code.org/>. Acesso em: 12 de junho de 2019.

COOPER, S.; DANN, W.; LEWIS, D.; LAWHEAD, P.; RODGER, S.; SCHEP, M.; STALVEY, R. H. A pre-college professional development program. In: Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, Darmstadt, Alemanha, 2011, pp. 188-192.

COOPER, S., RODGER, S. H.; SCHEP, M.; STALVEY, R. H.; DANN, W. Growing a K-12 Community of Practice. In: Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, EUA, 2015, pp. 290-295.

CSTA. ACM. CSTA K-12 Computer Science Standards Revised 2011. The CSTA Standards Task Force, ACM, New York/EUA.

CSTA. ACM. CSTA K –12 Computer Science Standards, 2016. Disponível em:

<http://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf>. Acesso em: outubro de 2018.

DE KEREKI, I. F.; MANATAKI, A. “Code Yourself” and “A Programar”: a bilingual MOOC for teaching Computer Science to teenagers. In: Proceedings of the Frontiers in Education Conference (FIE), Erie, EUA, 2016, pp.1-9.

DRISCOLL M. P. Psychology of Learning for Instruction. Allyn & Bacon, 2000.

FERRAZ A. P. C. M..; BELHOT R. V. Taxonomia de Bloom: revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais. Gest. Prod., vol. 17, n. 2, p. 421-431, 2010.

FERREIRA, N. F. Design de Interfaces para apps: proposição de uma unidade de ensino para estudantes do ensino fundamental II. Programa de Pós Graduação em Design (PósDESIGN) - Universidade Federal de Santa Catarina, 2019.

FREEDMAN, D.; PISANI, R.; PURVES.; R. Statistics. New York: W. W. Norton & Company, 2007.

FREIRE, P. Extensão ou comunicação? Rio de Janeiro: Paz e Terra, 1971.

FREITAS, S. R. P. C. Processo de Ensino e Aprendizagem: A Importância da Didática. In: Anais VIII Fórum Internacional de Pedagogia, 2016 pp. 1-6.

GAL-EZER, J.; STEPHENSON, C.. Computer Science Teacher Preparation is Critical. ACM Inroads, vol. 1, n. 1, pp. 61–66, 2010.

GAROFALAKIS, J. Enhancing teachers’ digital skills - Developing digital competencies in the school of tomorrow. In: Proceedings of the Classroom Conference, Atenas, Grécia, 2015.

GELDREICH, K.; TALBOT, M.; HUBWIESER, P. Off to new shores: preparing primary school teachers for teaching algorithmics and programming. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education, Potsdam, Alemanha, 2018, pp. 1-6.

GIANNAKOS, M. N.; DOUKAKIS, S.; CROMPTON, H.; CHRISOCHOIDES, N.; ADAMOPOULOS, N.; GIANNOPOULOU, P. Examining and mapping CS teachers' technological, pedagogical and content knowledge (TPACK) in K-12 schools. In: Proceedings of the 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, Madrid, Espanha, 2014.

GOODE, J. If you build teachers, will students come? the role of teachers in broadening computer science learning for urban youth. Journal of Educational Computing Research, vol. 36 n. 1, pp.65–88, 2007.

GOODE, J. Increasing Diversity in K-12 Computer Science: Strategies from the Field. In: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, EUA, vol. 40, n. 1, pp. 362-366, 2008.

GRANOR, N.; DELYSER, L. A.; WANG, K. TEALS: Teacher Professional Development Using Industry Volunteers. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, Memphis, EUA, 2016, pp. 60-65.

GRESSE VON WANGENHEIM, C. G.; SHULL, F. To game or not to game? IEEE Software, vol. 26, n. 2, pp. 92-94, 2009.

GRESSE VON WANGENHEIM, C.; ALVES, N. d. C.; RODRIGUES, P. E.; HAUCK, J. C. R. Teaching Computing in a Multidisciplinary Way in Social Studies Classes in School – A Case Study. International Journal of Computer Science Education in Schools, vol. 1, n. 2, 2017.

GRESSE VON WANGENHEIM, C.; HAUCK, J. C. R.; DEMETRIO, M. F.; PELLE, R. ALVES, N. d. C.; BARBOSA, H.; AZEVEDO, L. F. CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! Programs. Informatics in Education, vol. 17, n. 1, 2018, pp. 117-150.

GRESSE VON WANGENHEIM, C.; PETRI, G.; ZIBETTI, A. W.; BORGATTO, A. F.; HAUCK, J. C. R.; PACHECO, F. S.; MISSFELDT FILHO, R. dETECT: Um Modelo para a Avaliação de Unidades Instrucionais para o Ensino de Computação na Educação Básica. Disponível em: <http://www.incod.ufsc.br/wp-content/uploads/2017/04/Relatorio-Tecnico-dETECT-201701.pdf>. Acesso em: Setembro de 2019.

GRESSE VON WANGENHEIM, C.; ARAÚJO E SILVA DE MEDEIROS, G. A; MISSFELDT FILHO, R.; PETRI, G. ; DA CRUZ PINHEIRO, F.; FERREIRA, M. N. F.; HAUCK, J. C. R. SplashCode– A Board Game for Learning an Understanding of Algorithms in Middle School. Informatics in Education, 2019.

GROVER, S.; PEA, R. Computational thinking in K–12: A review of the state of the field. Educational Researcher, vol.42, n. 1, pp. 38–43, 2013.

HADEN, P.; GASSON, J.; WOOD, K.; PARSONS, D. Can you learn to teach programming in two days?. In: Proceedings of the Australasian Computer Science Week Multiconference, Canberra, Austrália, 2016.

HAMLEN, K.; SRIDHAR, N.; BIEVENUE, L.; JACKSON, D. K.; LALWANI, A. Effects of Teacher Training in a Computer Science Principles Curriculum on Teacher and Student Skills, Confidence, and Beliefs. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, EUA, 2018, pp. 741-746.

HAMNER, E.; CROSS, J.; ZITO, L.; BERNSTEIN, D.; MUTCH-JOBES, K. Training teachers to integrate engineering into non-technical middle school curriculum. In: Proceedings 2016 IEEE Frontiers in Education Conference (FIE), Erie, EUA, 2016, pp. 1-9.

HILL, H.; ROWAN, B.; BALL D. Effects of Teachers' Mathematical Knowledge for Teaching on Student Achievement. American Educational Research Journal, 42(2), 2005, pp. 371-406.

HODHOD, R.; KHAN, S.; KURT-PEKER, Y.; RAY, L. Training Teachers to Integrate Computational Thinking into K-12 Teaching. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, Memphis, EUA, 2016, pp. 156-157.

HOED, R. M. Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de Computação. Dissertação apresentada como requisito parcial para conclusão do Mestrado Profissional em Computação Aplicada - Departamento de Ciência da Computação, Universidade de Brasília, Brasília, 2016.

INEP. Censo da Educação Superior, 2015. Disponível em: <http://www.andifes.org.br/wp-content/uploads/2017/04/INEP-Censo-da-Educa%C3%A7%C3%A3o-Superior-Andifes-16042017.pdf>. Acesso em: Agosto de 2018.

INEP. Notas Estatísticas Censo Escolar da Educação Básica, 2016. Disponível em: <http://download.inep.gov.br/educacao_basica/censo_escolar/notas_estatisticas/2017/notas_estatisticas_censo_escolar_da_educacao_basica_2016.pdf>. Acesso em: Agosto de 2018.

INEP. Microdados da Prova Brasil, 2017. Disponível em: <https://www.qedu.org.br/brasil/pessoas/professor>. Acesso em: Outubro de 2019.

INEP. Censo Escolar, 2018. Disponível em: <https://www.qedu.org.br/brasil/censo-escolar?year=2018&dependence=0&localization=0&education_stage=0&item=r>. Acesso em: Outubro de 2019.

ISTE. ISTE Standards for Computer Science Educators. Dísponível em : <https://www.iste.org/standards/for-computer-science-educators>. Acesso em: 12 de junho de 2019.

JUSTEN, K. A. Desenvolvimento de um analisador de design de interface no contexto do ensino de computação com o App Inventor. Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) - Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2019.

KAMPF, C. A geração Z e o papel das tecnologias digitais na construção do pensamento. Jornal ComCiência, vol. 1, n. 131, 2011.

KAY, J. S.; MOSS, J. G. Using robots to teach programming to K-12 teachers. In: Proceedings of the 2012 Frontiers in Education Conference, Seattle, EUA, 2012, pp. 1-6.

KAY, J. S.; MOSS, J. G.; ENGELMAN, S.; MACKLIN, T. Sneaking in through the back door: introducing k-12 teachers to robot programming. In: Proceedings of the 45th ACM technical symposium on Computer science education, Atlanta, EUA, 2014, pp. 499-504.

KELLEHER K.; PAUSCH R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Comput. Surv., vol. 37, n. 2, pp. 83-137, 2005.

KIRKPATRICK, D. L.; KIRKPATRICK, J. D. Evaluating training programs: the four levels. Berrett-Koehler Publishers, 2006, p. 379.

KITCHENHAM, B. Procedures for Performing Systematic Reviews. Joint Technical Report, TR/SE-0401 and NICTA 0400011T.1: Keele University. Reino Unido, 2004.

KOELLNER, K.; JACOBS, J. Knowledge, Instruction, and Student Achievement Distinguishing Models of Professional Development: The Case of an Adaptive Model's Impact on Teachers' On behalf of. American Association of Colleges for Teacher Education. Journal of Teacher Education, 2015, pp. 51-67.

KUBO, O. M.; BOTOMÉ, S. P.. Ensino-aprendizagem: uma interação entre dois processos comportamentais. Interação em Psicologia, Curitiba, vol. 5, 2001.

LAMPROU, A.; REPENNING, A.; ESCHERLE, N. A. The Solothurn Project — Bringing Computer Science Education to Primary Schools in Switzerland In: Proceedings of the 22th Annual Conference on Innovation and Technology in Computer Science Education, Bologna, Itália, 2017.

LIBÂNEO, J. C. O processo de ensino na escola. São Paulo: Cortez, 1994. pp. 77-118.

LIU, J.; LIN, C. H.; HASSON, E. P.; BARNETT, Z.; XU, Y. Introducing computer science to K-12 through a summer computing workshop for teachers. In: Proceedings of the 42nd ACM technical symposium on Computer science education, Dallas, EUA, 2011, pp. 389-394.

LIU, J.; LIN, C. H.; POTTER, P.; HASSON, E. P.; BARNETT, Z.; XU, Y. Singleton M. Going mobile with app inventor for android: a one-week computing workshop for K-12 teachers. In: Proceedings of the 44th ACM technical symposium on Computer science education, Denver, EUA, 2013, pp. 433-438.

LIU, J.; LIN, C. H.; WILSON, J.; HEMMENWAY, D.; HASSON, E. P.; BARNETT, Z.; XU, Y Making games a "snap" with Stencyl: a summer computing workshop for K-12 teachers. In: Proceedings of the 45th ACM technical symposium on Computer science education, 2014, Atlanta, EUA pp. 169-174.

LIU, J.; WILSON, J.; HEMMENWAY, D.; XU, Y.; LIN, C. Oh SNAP! A one-week summer computing workshop for K-12 teachers. In: Proceedings of the 2015 10th International Conference on Computer Science & Education (ICCSE), Cambridge, Reino Unido, 2015, pp. 663-668.

LLOYD, M.; COCHRANE, J. Celtic knots: Interweaving the elements of effective teacher professional development in ICT. 2006.

LYE, S. Y.; KOH, J. H. L. Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, vol. 41, n. C, pp. 51-61, 2014.

MARTIN, L. S. N.; TOSCHI, M. S. Celular na escola: políticas, usos e desafios

pedagógicos. Revista Inter Ação, vol. 39, pp. 557-574, 2018.

MARTINEZ, M.; GOMEZ, M. J.; MORESI, M.; BENOTTI, L. Lessons Learned on Computer Science Teachers Professional Development. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, Arequipa, Peru, 2016, pp. 77-82.

MEC. Base Nacional Comum Curricular, 2019. Disponível em: <http://basenacionalcomum.mec.gov.br/abase/>. Acesso em: maio de 2019.

MICHAELIS. Dicionário prático da língua portuguesa. 1. ed. São Paulo: Editora Melhoramentos, 2009.

MISHRA, P.; KOEHLER, M. J. Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge, vol. 108, n. 6, pp. 1017-1054, 2006.

MISSFELDT FILHO, R. Desenvolvimento de uma unidade instrucional para ensinar o desenvolvimento de apps no Ensino Fundamental com o App Inventor. Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) - Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2019.

MITa. App Inventor - Explore. Disponível em: <http://appinventor.mit.edu/explore/>. Acesso em: 12 de junho de 2019.

MITb. Scratch. Disponível em: <https://Scratch.mit.edu/>. Acesso em: 12 de junho de 2019.

MORELLI, R.; UCHE, C.; LAKE, P.; BALDWIN, L. Analyzing Year One of a CS Principles PD Project. In: Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, EUA, 2015, pp. 368-373.

MORENO-LEÓN, J.; ROBLES, G. Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. Proceedings of the Workshop in Primary and Secondary Computing Education, Londres, Reino Unido, 2015.

NI, L.; GUZDIAL, M.; TEW, A. E.; MORRISON, B.; GALANOS, R. Building a community to support HS CS teachers: the disciplinary commons for computing educators. In: Proceedings of the 42nd ACM technical symposium on Computer science education, Dallas, EUA, 2011, pp. 553-558.

NI, L.; GUZDIAL, M. Who AM I?: understanding high school computer science teachers' professional identity. In: Proceedings of the 43rd ACM technical symposium on Computer Science Education, Raleigh, EUA, 2012, pp. 499-504.

PARTANEN, T.; MANNILA, L.; PORANEN, T. Learning programming online: a racket-course for elementary school teachers in Finland. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli, Finlândia, 2016, pp. 178-179.

PETERSEN, K.; VAKKALANKA, S.; KUNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, v. 64, 2015, p. 1-18.

PINHEIRO, F. C. Modelo instrucional para o ensino de Engenharia de Software e Usabilidade voltado à Educação Básica. Programa de Pós-Graduação em Ciência da Computação (PPGCC) - Universidade Federal de Santa Catarina, 2019.

POKRESS, S.; DOMINGUEZ VEIGA, J. J. MIT App Inventor: Enabling Personal Mobile Computing. In: PROMOTO ’13, Indianapolis, EUA, 2013.

PROINFO/MEC. Coleta de dados do projeto PROINFO/MEC de inclusão digital nas escolas públicas brasileiras, 2018. Disponível em: <http://proinfodata.c3sl.ufpr.br/attendance/os/proinfo/>. Acesso em: Agosto de 2018.

RAVITZ, J.; STEPHENSON, C.; PARKER, K.; BLAZEVSKI, J. Early Lessons from Evaluation of Computer Science Teacher Professional Development in Google’s CS4HS Program. ACM Trans. Comput. Educ., vol. 17, n. 4, 2017.

ROBINSON, À.; PÉREZ-QUINÕNES, M. A. Underrepresented middle school girls: on the path to computer science through paper prototyping. In: Proceedings of the 45th ACM technical symposium on Computer science education, Atlanta, EUA, 2014. pp. 97-102.

RODRIGUES F. S. O Uso do Celular na Sala de Aula e a Legislação Vigente no Brasil. In: Proceedings of the 3nd Congresso sobre Tecnologia na Educação, Fortaleza, Brasil, 2018, pp.111-122.

RUSU, A. Introducing gaming tools for computing education in STEM related curricula. In: Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE), El Paso, EUA, 2015, pp. 1-8.

SAMEJIMA, F. A. Estimation of latent ability using a response pattern of graded scores. Psychometric Monograph, v. 17, 1969.

SARDESSAI, S.; KAMAT, V. V. Using a Blended Approach to In-service Teacher Training: A Case Study. In: Proceedings of the 2011 IEEE International Conference on Technology for Education, Washington, EUA, 2011, pp. 266-269.

SASKATCHEWAN EDUCATION. Instructional Approaches: A Framework for Professional Practice. Saskatchewan Education, Canada, 1991.

SBC. Diretrizes para ensino de computação na Educação Básica, 2018. Disponível em: <http://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/1177-diretrizes-para-ensino-de-computacao-na-educacao-basica>. Acesso em: Maio de 2019.

SBC. Referenciais de Formação em Computação: Educação Básica, 2017. Disponível em: <http://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/1166-referenciais-de-formacao-em-computacao-educacao-basica-julho-

2017>. Acesso em: Maio de 2018.

SHADISH, W. R.; COOK, T. D.; CAMPBELL, D. T. Experimental and quasi-experimental designs for generalized causal inference. Houghton Mifflin Company, 2002.

SHERMAN, M.; MARTIN, F.; BALDWIN, L.; DEFILIPPO, J. App Inventor Project Rubric – Computational Thinking through Mobile Computing, 2014. Disponível em: <https://nsfmobilect.files.wordpress.com/2014/09/mobile-ct-rubric-for-app-inventor-2014-09-01.pdf>. Acesso em: Agosto de 2018.

SHERMAN, M.; MARTIN, F. The assessment of mobile computational thinking. Journal of Computing Sciences in Colleges, vol. 30, n. 6, pp. 53–59, 2015.

SHULMAN, L. S. Those who understand: Knowledge growth in teaching. Journal Educational Researcher, vol. 15, n. 2, pp. 4–14, 1986.

SMITH, D. C.; CYPHER, A.; TESLER, R. Novice programming comes of age. Communications of the ACM, vol. 43, n. 3, pp. 75-81, 2000.

SNAP!; BERKELEY. Snap!, 2013. Disponível em: <http://snap.berkeley.edu>. Acesso em: Outubro 2019.

SOLECKI, I.; PORTO, J. A.; JUSTEN, K. A.; ALVES, N. d. C., GRESSE VON WANGENHEIM, C., BORGATTO, A. F.; HAUCK, J. C. R. CodeMaster UI Design – App Inventor: Uma Rubrica de Avaliação do Design de Interface de Aplicativos Android desenvolvidos com App Inventor. In: Proc. of the XVII Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais, Vitória, Brasil, 2019.

TISSENBAUM M.; SHELDON, J.; ABELSON, H. From computational thinking to computational action. Magazine Communications of the ACM, vol. 62, n. 3, pp. 34-36, 2019.

TODOS PELA EDUCAÇÃO. Pesquisa sobre Perfil dos Professores da Educação Básica, 2018. Disponível em: <https://www.todospelaeducacao.org.br//_uploads//_posts/23.pdf?750034822>. Acesso em: Agosto de 2018.

TOLEDO, P. B. F. O Comportamento da Geração Z e a Influência nas Atitudes dos Professores. In: Proceedings of the IX Simpósio de excelência em Gestão e Tecnologia, Rio de Janeiro, Brasil, 2012.

VAN DRIEL, J. H.; BEIJAARD, D.; VERLOOP, N. Professional development and reform in science education: The role of teachers' practice and knowledge. Journal of Research in Science Teaching, vol. 38, n. 2, pp. 137-158, 2001.

VERLOOP, N.; VAN DRIEL, J. H.; MEIJER, P. C. Teacher knowledge and the knowledge base of teaching. International Journal of Educational Research, vol. 35, n. 5, pp. 441-461, 2001.

VIVIAN, R.; FALKNER, K.; FALKNER, N. Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. Research in Learning Technology, 2014.

VON WANGENHEIM, A.; GRESSE VON WANGENHEIM, C.; PACHECO, F. S.; HAUCK, J. C. R.; FERREIRA, M. N. F. Motivating Teachers to Teach Computing in Middle School – A Case Study of a Physical Computing Taster Workshop for K-12 Teachers. International Journal of Computer Science Education in Schools, vol. 35, n. 1, pp. 368-373, 2017.

WENGER, E. R.; MCDERMOTT, R.; SNYDER, W. M. Cultivating communities of practice: a guide to managing knowledge. Boston, Mass, Harvard Business School Press, 2002.

WOHLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. Experimentation in Software Engineering. Berlin: Springer-Verlag Berlin Heidelberg, 2012, p. 235.

WONGSOPAWIRO, D. S. Examining science teachers' pedagogical content knowledge in the context of a professional development program. 2012. Tese de doutorado, Leiden University, Países Baixos.

YIN, R. K. Case Study Research: Design and Methods. SAGE, 4 ed., 2009, p. 219.

APÊNDICE I

Tabela I.1. Visão geral das competências de computação e pedagogia das UI

ID	Objetivo geral de aprendizagem da UI	Estágio alvo de competência	Descrição geral	Conceitos e práticas de computação	Competências pedagógicas	Competências tecnológicas
(PARTANEN, MANNILA & PORANEN, 2016)	Ensinar programação para professores, promovendo criatividade e compartilhamento de artefatos programados e ideias pedagógicas durante o curso, e uso de exercícios. Outros objetivos: dar aos participantes do curso conhecimento suficiente sobre o assunto para que eles sejam capazes não apenas de avaliar e usar materiais de programação prontos, mas também para criar seus próprios exercícios de programação, e permitir o apoio de pares, pedindo participantes para ajudar uns aos outros através de áreas de discussão.	applying	A UI ocorre na forma de MOOC e foi desenvolvida buscando se adequar ao novo currículo finlandês, que abrange alguns conceitos de computação. Os principais assuntos ensinados são algoritmos, programação e pensamento computacional, além de aspectos pedagógicos e tecnológicos. O diferencial dessa UI é o modo online, no qual os participantes podem estudar quando tiverem tempo livre.	Desenvolvimento e uso de abstrações, algoritmos e programação (lógica e repetição)	Conhecimento pedagógico (CK) e requisitos curriculares da Finlândia	Conhecimento tecnológico (TK)
(MARTINEZ *et al.*, 2016)	Ensinar aos professores práticas de programação e de ensino para que professores possam aplicar computação em suas salas de aula.	applying	A UI foi aplicada no modelo presencial e busca proporcionar aos participantes experiências práticas de programação, ensinando também conceitos fundamentais, pedagógicos e tecnológicos. O diferencial é a utilização de múltiplas linguagens e plataformas.	Algoritmos e programação (condicionais, loops, variáveis etc.) e colaboração na computação (através do uso de múltiplas linguagens e plataformas)	Conhecimento, pedagógico (PK), através da prática de sala de aula para professores	Conhecimento tecnológico (TK)
(KAY & MOSS, 2012) (KAY *et al.*, 2014)	Fornecer aos professores conhecimentos de conteúdos de programação de robôs suficientes para poderem resolver problemas e elaborar atividades. Dar aos professores a confiança e as habilidades para iniciar programas extra-curriculares de robótica com seus alunos.	applying	A UI é realizada no formato de workshop (presencial) e tem o objetivo de fornecer conhecimento de programação de robôs.	Algoritmos e programação (saídas simples, sensores, loops, manipulação de eventos, condicionais, funções e variáveis)	Conhecimento, pedagógico (PK), através do uso de formas de integrar a ciência da computação na sala de aula	NI
(COOPER *et al.*, 2011) (COOPER *et*	Ensinar programação para professores para que possam desenvolver planos de aula	applying	A UI acontece em workshop (presencial) e tem o intuito de ensinar programação	Algoritmos e programação	NI	Conhecimento tecnológico (TK)

al., 2015)	específicos de disciplina para integrar computação nas escolas de ensino fundamental e médio. Desenvolver a capacidade de adaptar os tutoriais presentes no site para aplicar em suas aulas.		aos participantes e fazê-los obter a capacidade de adaptar tutoriais para poderem aplicar em sala de aula. Os assuntos ensinados neste workshop são algoritmos, programação e conhecimento tecnológico (TK). O diferencial é o apoio após o término da UI.			
(LIU *et al.*, 2015)	Introduzir conceitos fundamentais de computação para professores do ensino fundamental e médio usando o SNAP! para que eles possam desenvolver currículos para os assuntos que irão ensinar nos semestres seguintes.	applying	Workshop presencial, no qual os participantes desenvolvem seus próprios currículos que podem ser usados em sala de aula. Possui a intenção de ensinar conceitos fundamentais de computação e não abrange aspectos pedagógicos e tecnológicos.	Algoritmos e programação (entrada de dados, condicionais, booleanos, variáveis, laços de repetição, métodos e listas)	NI	NI
(LIU *et al.*, 2013)	Introduzir conceitos de computação usando o App Inventor para professores do ensino fundamental e médio de diversas áreas. Os participantes devem desenvolver projetos para os assuntos que irão ensinar nos semestres seguintes.	applying	Uma oficina presencial com a meta de introduzir conceitos fundamentais de computação utilizando o AppInventor. Também busca ensinar conhecimento pedagógico através da elaboração de material curricular. O diferencial é o aprendizado em usar o emulador para simular a execução dos programas no Android.	Algoritmos, programação (entrada e saída de um programa, estruturas condicionais, “boolean”, variáveis, “loops”, métodos e funções), dados e análise	Conhecimento, pedagógico (PK), através de material curricular para suas áreas de conhecimento	Conhecimento tecnológico (TK), através da simulação da execução dos programas no emulador do Android
(VIVIAN, FALKNER & FALKNER, 2014)	Os objetivos estão relacionados aos impactos culturais da tecnologia, o pensamento computacional, o uso de sistemas e dados digitais, bem como conceitos e práticas de programação visual.	applying	Um MOOC (online) para entregar gratuitamente conteúdo de computação e pedagogia para professores com a integração de mídias sociais para apoiar a troca de conhecimento e a construção de recursos. Também abrange o ensino de conhecimento tecnológico. O diferencial dessa UI é o modo online, no qual os participantes podem estudar quando tiverem tempo livre.	Sistemas de computador, algoritmos e programação (programação visual, padrões e representação de dados)	Conhecimento, pedagógico (PK), através da criação de um recurso de ensino e um plano de aula	Conhecimento tecnológico (TK)
(HAMNER *et*	O objetivo promover a	applying	Uma UI presencial que	Algoritmos e	Conhecimento,	Não é uma meta

al., 2016)	habilidade, confiança e autoeficácia do professor no projeto e implementação de projetos de robótica em sala de aula.		busca ensinar pensamento computacional, algoritmos e também aspectos pedagógicos através da elaboração de um plano de ensino. Embora o conhecimento tecnológico não seja uma meta de aprendizado, a UI tem a finalidade de integrar de projetos tecnológicos em atividades multidisciplinares.	programação	pedagógico (PK), através da avaliação do perfil dos estudantes e elaboração de um plano de ensino	de aprendizado
(HADEN *et al.*, 2016)	Ensinar a habilidade de programação para professores e explicar técnicas pedagógicas em programação.	applying	Oficina presencial que desenvolve os conhecimentos do conteúdo e pedagógicos (PCK) e enfatizar os princípios fundamentais da computação programação, não apenas o uso de uma ferramenta específica ou língua. O diferencial é a acessibilidade aos participantes curta duração, aulas nos finais de semana e / ou feriados e de baixo custo.	Algoritmos, programação, reconhecimento e definição de problemas computacionais	Conhecimento pedagógico (PK), através da demonstração e exploração de práticas educativas específicas que os professores podem usar imediatamente em suas próprias salas de aula; fornecimento de amostras de materiais de sala de aula para fornecer conteúdo do curso e estruturar um ciclo de vida de desenvolvimento de software; discussão de questões puramente pedagógicas, como o valor da analogia e metáfora ao explicar conceitos abstratos de computação como fluxo de controle.	NI
(LIU *et al.*, 2014)	Aprender conceitos fundamentais de computação, programação usando Stencyl e desenvolver projetos para os assuntos que irão ensinar nos semestres seguintes.	applying	A UI ocorre na forma de workshop (presencial) e focou na introdução de computação fundamental conceitos para professores K-12 usando Stencyl, e ensinou também aspectos pedagógicos. O diferencial desse workshop é o uso de exemplos com jogos.	Algoritmos, programação (entrada e saída de um programa, estruturas condicionais, “boolean”, variáveis, “loops”, métodos, classes e listas) e criação de artefatos computacionais	Conhecimento pedagógico (PK), através do desenvolvimento de projetos para a sua disciplina	NI
(RUSU, 2015)	Ensinar aos educadores como usar a ciência da computação como um meio para fazer conexões entre diferentes áreas	applying	Esse programa oferece as aulas no formato de oficina presencial, com o ensina muitos conceitos de	Algoritmos, programação (incluindo reconhecimento de padrões), desenvolvimento	Conhecimento pedagógico (PK), através da identificação do conceito para educar os alunos,	NI

	curriculares e ensinar habilidades de resolução de problemas de ordem superior. Programas de desenvolvimento profissional ofereceram as habilidades necessárias para integrar a educação do Google Apps e a metáfora interativa, ferramentas de jogos de computador baseados em meio e ensino médio currículos, para ajudar os alunos a aprender ciência da computação e conceito de engenharia.		computação e o seu diferencial é o uso de jogos educacionais para transmitir conhecimento.	e uso de abstrações	desenvolvimento de metáforas adequadas para o conceito, identificação de ferramentas adequadas e utilização de jogos para educar seus alunos.	
(GELDREICH, TALBOT & HUBWIESER, 2018)	Oficina de desenvolvimento profissional in-service em algoritmos e programação. Eles então serão acompanhados e supervisionado por um ano letivo na implementação dos novos tópicos.	applying	A aplicação acontece no modo presencial e busca ensinar pensamento computacional, algoritmos e programação, além de aspectos pedagógicos que tem o intuito de fazer os participantes imaginarem como serão implementados esses conteúdos no currículo.	Algoritmos e programação	Conhecimento pedagógico (PK), através do foco em como os professores poderiam implementar esses conceitos no currículo da escola	NI
(VON WANGENHEIM *et al.*, 2015)	Formação de professores in-service de ensino médio, nós projetamos um workshop para motivar e despertar o entusiasmo dos professores por esta área. O objetivo deste workshop é ensinar conceitos básicos de computação o ensino médio professores de qualquer área do conhecimento (e / ou professores que fornecem TI apoio nas escolas), visando uma integração multidisciplinar da computação no currículo existente.	emerging	O workshop abrange o conhecimento com foco no pensamento computacional e programação por meio da programação de robôs interativos (Arduino e *Scratch*/Snap!), a aplicação do ciclo de engenharia de software e práticas colaborativas também como a compreensão de quais algoritmos são e como eles funcionam. O diferencial é a abordagem multidisciplinar que a UI aplica e o ensino de como instalar o Linux..	Algoritmos e programação (loops, condicionais, manipulação de eventos, etc.)	Conhecimento pedagógico (PK), incluindo possibilidades de integrar a educação em computação em forma multidisciplinar em diferentes disciplinas e estratégias instrucionais	Conhecimento tecnológico (TK), incluindo instalação de Linux
(ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009)	O objetivo do curso foi capacitar os formandos a compreender as perspectivas pedagógicas da robótica educacional e desenvolver atividades robóticas dentro da abordagem construtivista do ensino.	applying	Workshop presencial que apresenta uma metodologia construtivista para a formação de professores em tecnologia robótica e sua implementação no âmbito de um curso de formação de professores. A metodologia em	NI	Pedagogia construtiva	Conhecimento tecnológico (TK), através da montagem dos robôs LEGO

			questão visa manter em sintonia com o uso proposto da robótica como uma ferramenta de ensino e aprendizagem construtivista e se destina a treinar os professores da maneira que eles serão convidados a treinar seus alunos nas aulas da escola.			
(CHO *et al.*,2014)	Os objetivos do workshop apresentados aos professores foram: 1) melhorar a percepção social da pesquisa e oportunidades de trabalho em ciência da computação; 2) preparar os professores para se tornarem modelos eficazes para o pensamento computacional; e 3) ajudar a integrar o pensamento computacional seus currículos.	applying	A UI ocorre no formato workshop presencial como ensino de pensamento computacional, e não é ensinado aspectos pedagógicos e tecnológicos. O diferencial foi que a maior parte do workshop envolveu atividades práticas usando o *Scratch* e o App Inventor.	Algoritmos, programação (representação de dados, pesquisa binária e o problema da mochila), desenvolvimento e uso de abstrações, testar e refinar artefatos computacionais	NI	NI
(LIU *et al.*, 2011)	O objetivo é introduzir conceitos de computação para professores em computação, tecnologia, matemática e ciências em todos os níveis do ensino para expor os alunos à computação em idade precoce.	applying	Workshop presencial com aulas durante uma semana os participantes aprenderam conceitos de computação e pensamento computacional e desenvolveram materiais curriculares para os assuntos que irão ensinar nos semestres seguintes. O diferencial foi o ensino de instruções de instalação do software.	Algoritmos e programação (estrutura do programa e fluxo de execução, objetos, métodos / funções, ramos condicionais, expressão booleana, variáveis, loops, entrada do usuário lógica booleana e processamento paralelo)	Conhecimento pedagógico (PK), através do desenvolvimento de materiais curriculares para os assuntos que irão ensinar nos semestres seguintes, com a ajuda dos tutores.	Conhecimento tecnológico (TK), através de instruções de instalação do software.

Fonte: elaborada pela autora.

Tabela I.2. Visão geral das características das UI

ID	**Modo de educação**	**Duração da UI**	**Ambiente de programação**	**Método instrucional**	**Organização das Seções**
(PARTANEN, MANNILA & PORANEN, 2016)	MOOC (online)	42 horas / aula	ScratchJr, Scratch, Racket e Python, com DrRacket e WeScheme	Cada tópico começa com um vídeo motivacional seguido de tutorial, com vídeos introduzindo novos conceitos durante a programação. Os exercícios de programação e suas soluções são utilizados para fins de auto-avaliação. Cada semana um pequeno artefato de programação é entregue, 2/3 deles são exibidos em uma	1- Introdução à programação com Racket usando imagens, problemas e como usar variáveis como constantes globais. 2- Como usar funções e parâmetros para resolver problemas (abstração), como usar booleanos, operadores

				parede de curso estilo Padlet, o resto é revisado por pares. O trabalho final consiste em uma redação sobre aspectos pedagógicos.	de comparação, predicados e a estrutura condicional (if) controlar a execução de código, como testar funções a partir da janela de interação e como gravar testes de unidade (check-expect). 3- Uso de receitas de design em programação funções (escrevendo primeiro casos de teste), como usar operadores booleanos e expressões condicionais para formar expressões com lógica mais complexa, uso do mecanismo de animação, uso de WeScheme para compartilhar código 4- Como dividir o código em funções auxiliares, como escrever funções recursivas (incluindo uma função que chama a si mesma para formar um loop), como obter entrada do usuário, como usar variáveis locais para armazenar entradas do usuário, como usar blocos de código com código com efeitos colaterais (interação do usuário). 5- Como usar listas para armazenar um conjunto de valores, como pegar uma lista recursivamente e produzir uma nova lista ou um valor de resultado, como usar arquivos de imagem em aplicativos DrRacket e WeScheme. 6- Looping usando funções de ordem superior (mapa) e listas, uso da biblioteca Racket Turtle para desenhar formas geométricas 7- Quais são os requisitos reais em relação à programação no currículo finlandês, o que se entende por pensamento algorítmico / pensamento computacional, como ensiná-lo, como integrá-lo com outros assuntos.
(MARTINEZ *et*	Presencial	60 horas /	Alice, Code.org,	Cada um dos encontros	Três módulos: Robot,

al., 2016)		aula	Chatbot e UNC++Duino	segue uma abordagem baseada em investigação. Em um curto segmento de 5 minutos do encontro são apresentadas particularidades da plataforma. Antes de introduzir qualquer conceito, é dado aos professores um desafio de programação. Os professores então resolvem o desafio em grupos de 4 a 5 pessoas que promovem a colaboração entre professores. Os treinadores andam pela sala para ajudar os professores e prevenir ansiedade e frustração.	Chatbot e Animations; abordando conceitos fundamentais de programação, como seqüência, evento, condicional, ciclo, variável, método, parâmetro, entre outros. Prática de aplicação em sala de aula, onde os professores aplicam as lições com seus alunos. Sessões pessoas de coaching com estudantes de CCO.
(KAY & MOSS, 2012) (KAY *et al.*, 2014)	Workshop presencial	25 horas / aula	LEGO NXT-G, Scratch e Alice	Lições Interativas de Programação Robótica: começam com uma breve introdução a um tópico, seguido por uma série de exemplos de programação passo-a-passo que o instrutor passa aos professores (os professores programam seus próprios robôs em paralelo com o instrutor); Aulas expositivas: Os tópicos foram apresentados como um conjunto de curtas aulas com perguntas; Hands-on Exercises: exercícios com dificuldades crescentes com assistência dos instrutores; Discussões e práticas sobre outras formas de integrar a Ciência da Computação em sala de aula. Demonstrações: No final do workshop, os participantes demonstraram seus programas favoritos uns aos outros.	1) Lições Interativas de Programação Robótica 2) Sessões expositivas curtas 3) Sessões práticas 4) Demonstrações
(COOPER *et al.*, 2011) (COOPER *et al.*, 2015)	Workshop presencial	120 horas / aula	Alice	Aulas através de tutorias, problemas para resolver e desenvolvimento de um plano de ensino. Um ano depois retornam para mais uma semana de prática. O projeto também oferece apoio para os professores durante o ano escolar.	NI
(LIU *et al.*, 2015)	Workshop presencial	25 horas / aula	SNAP!	O período da manhã foi para introduzir conceitos fundamentais de computação, juntamente com projetos de exemplo e projetos de designação usados para demonstrar especificamente o tópico de programação da sessão. As tardes eram usadas pelos participantes para desenvolver seus próprios currículos que	0- Snap! Visão Global 1- Snap! Introdução 2- Entrada e saída 3- Desvio Condicional e boolean 4- Variáveis 5- Loops 6- Métodos 7- Listas

				poderiam ser retomados e usados em suas próprias salas de aula. Os participantes apresentaram seu SNAP! projetos curriculares nos últimos dois dias do workshop.	
(LIU *et al.*, 2013)	Oficina presencial	25 horas / aula	App Inventor	O período da manhã foi geralmente dividido em duas sessões, cada introduzindo conceitos fundamentais de computação usando exercícios e tarefas usadas para mostrar especificamente o tópico de programação da sessão. Conceitos de programação foram apresentados passo a passo. Durante as tardes os participantes desenvolvem seus próprios projetos curriculares que poderiam ser usados em sua disciplina. Tutores ajudavam os participantes conforme necessário. Os participantes apresentaram seus aplicativos de currículo do App Inventor no último dia da oficina.	0- App Inventor Visão Geral 1a- Variáveis 1b- Uso de entradas 2- Desvio condicional e Expressões booleanas 3- Loops 4- Procedimentos (Métodos e Funções) 5- Booleana Avançada e Listas 6- Tópicos Avançados
(VIVIAN, FALKNER & FALKNER, 2014)	MOOC (online)	NI	Scratch	Aprendizagem online, estudo auto-dirigido. Utilização de espaço que os que permite participantes se conectarem, compartilharem ideias e atividades e construir coletivamente recursos online correspondentes às áreas de tópicos. Podem visualizar o conteúdo e localizar recursos adicionais. Udo de hangouts permitindo os instrutores construírem uma conexão com os participantes assistindo, discutindo questões comuns sobre os cursos e respondendo às perguntas postadas. Acesso à vídeos pelo youtube.	1- Introdução 2- Dados (padrões e jogo) 3- Dados (Representação) 4- Sistemas Digitais 5- Sistemas de informação 6- Algoritmos e programação 7- Programação Visual
(HAMNER *et al.*, 2016)	Presencial	4 horas / aula	Kits de robótica, CREATE Lab Visual Programmer	Introdução ao kit de hardware Hummingbird através de uma aula ativa detalhada, seguida de uma introdução ao ambiente CREATE Lab Visual programmer, seguida de uma prática direcionada com o software de programação.	NI
(HADEN *et al.*, 2016)	Oficina presencial	20 horas / aula	Scratch	Em 2013 cada sessão era realizada em um dia, resultando em três dia. Já em 2014 a oficina foi compactada para dois dias.	Dividido em 3 secções: 1- Conceitos essenciais de programação e como usar o Scratch para ensiná-los. 2-Projetos completos de programação Scratch adequados para a sala de aula do

					ensino médio. 3- Discussão sobre como apoiar um projeto de desenvolvimento completo.
(LIU *et al.*, 2014)	Workshop presencial	25 horas / aula	Stencyl	O período da manhã foi dividido em duas sessões, cada uma introduzindo conceitos fundamentais de computação junto com amostra jogos e atribuição games usado para demonstrar especificamente o tópico de programação da sessão. As tardes foram usadas pelos participantes para desenvolverem seus próprios currículos que poderiam ser levados de volta e usados em suas próprias salas de aula. Durante o período de desenvolvimento curricular da tarde, os tutores ajudariam o participante, conforme necessário. Os participantes apresentou seu currículo Stencyl jogos nos últimos dois dias de oficina.	1- Introdução à Stencyl 2- Classes e Variáveis 3- Métodos 4- Desvio Condicional e Boolean 5- Loops 6- Blocos Personalizados 7- Listas
(RUSU, 2015)	Oficinas presencial	15 horas / aula	Scratch, Game Maker e Alice	Diferentes atividades de aprendizagem alternando entre aulas expositivas, discussão, atividades práticas, atividades unplugged, apresentação e brainstorming	NI
(GELDREICH, TALBOT & HUBWIESER, 2018)	Presencial	14 horas / aula	Scratch	Utiliza atividades unplugged, exercícios de programação e discussão.	Sessão 1: Introdução aos algoritmos e programação no contexto da escola primária Sessão 2: Envolvendo-se com estruturas algorítmicas básicas e Programação em Scratch Sessão 3: Próprias implementações nas escolas dos professores
(VON WANGENHEIM *et al.*, 2015)	Workshop presencial	6 horas / aula	Scratch, Snap!	Intercalação entre aulas expositivas, discussões e atividades práticas por pares	Parte 1: Conhecimento de computação (ensinando uma oficina para crianças) Parte 2: Conhecimento pedagógico e de conteúdo Parte 3: Conhecimento tecnológico
(ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009)	Presencial	30 horas / aula	Sistema LEGO Mindstorms NXT (LEGO NXT-G)	As tarefas de aprendizagem do curso foram organizadas para incentivar os professores a projetar e desenvolver seus próprios produtos. Participar de todas as atividades do	1º encontro: trabalho em grupos estudando uma seção específica de um trabalho sobre aprendizagem construtivista,

				curso por meio de discussões em pequenos grupos, apresentações em sessões plenárias e publicações em sua aula.	apresentaram resumidamente o resumo a toda a turma. 2º encontro: introdução dos materiais incluídos no kit Lego Mindstorms Education NXT e na montagem dos robôs. 3º encontro: trabalharam em grupos em diversas atividades destinadas a introduzir instruções de programação específicas e funcionalidades do ambiente de programação. 4º encontro: trabalharam em um projeto real projetado de acordo com essa metodologia, discutiram suas experiências e concluíram os critérios de avaliação para projetos aprimorados em robótica 5º encontro: Os alunos tiveram um kit à sua disposição por duas semanas para desenvolver seu próprio projeto com base na metodologia para projetar atividades aprimoradas de robótica propostas no curso. Então, durante esta sessão final, os trainees apresentaram seus próprios projetos e receberam feedback.
(CHO *et al.*,2014)	Workshop presencial	16 horas / aula	Scratch e App Inventor	Os alunos fizeram as apresentações, conduziram as atividades práticas e ajudaram ativamente os professores a desenvolver suas lições interativas. A maior parte do workshop envolveu atividades práticas usando o Scratch e o App Inventor.	Em primeiro lugar, o desenvolvimento de atividades práticas foi uma experiência desafiadora, mas recompensadora para os estudantes voluntários. Eles prepararam os “planos de aula” do Scratch e do App Inventor com pouca participação dos organizadores do corpo docente. Eles foram instruídos a discutir e desenvolver um programa Scratch e um App Inventor que eles achavam que teria valor educacional para os professores. Além disso, as lições tiveram que se relacionar com os atuais padrões de ensino médio e médio e ser factíveis em cerca de 30 a 60 minutos por

					nossa ampla audiência de professores. Simplicidade e facilidade de uso de seus programas, conceitos fundamentais em engenharia de software, eram questões desafiadoras, mas práticas, que eles tiveram que enfrentar na preparação de suas lições. Em segundo lugar, as lições concluídas listar tópicos de ciência da computação e os Padrões Essenciais da Carolina do Norte que foram abordados, criando um mapeamento para os professores entre tópicos de pensamento computacional e objetivos educacionais. Os planos de aula foram estruturados de modo que um professor pudesse iniciar a atividade relevante do Scratch ou do App Inventor sob nossa supervisão e terminar / estender a lição em seu próprio tempo. Os programas finais concluídos foram disponibilizados no site para os professores para obter um vislumbre do produto final. Essa abordagem incentivou os professores a serem criativos em suas próprias implementações dos planos de aula.
(LIU *et al.*, 2011)	Workshop presencial	25 horas / aula	Scratch e Alice	Através de aula práticas e uso de slides os tutores ensinaram os professores aspectos da computação e desenvolvimento de seus próprios currículos que poderiam ser retomados e usados em suas próprias salas de aula.	Scratch Sessão 1: Apresentação da ferramenta Scratch Sessão 2: Os participantes aprenderam como coletar informações do usuário e como essa entrada pode ser usada para afetar um programa em execução. Eles também foram apresentados aos problemas que surgem quando uma entrada inválida é aceita por um programa. Sessão 3: Enfoque na ramificação condicional

					como uma maneira de alterar o fluxo normal de execução em um programa baseado em uma expressão booleana. Sessão 4: Introdução do conceito de loops no Scratch. Sessão 5: Os participantes aprenderam como criar, alterar, usar e excluir variáveis por meio do uso de vários exemplos. Sessão 6: Os participantes foram introduzidos em expressões booleanas avançadas, passagem de mensagens e matrizes. Alice Após as sessões do Scratch, todos os conceitos centrais da computação foram introduzidos e discutidos; isso facilitou para os participantes aprenderem Alice durante o restante do workshop.

Fonte: elaborada pela autora.

ID	**Material instrucional**	**Métodos de avaliação**	**Inclusão de suporte**	**Língua**	**Licença**
(PARTANEN, MANNILA & PORANEN, 2016)	Vìdeos motivacionais, vídeos tutoriais com conceitos e exemplos de programação, exercícios e slides	Auto-avaliação por meio de exercícios de programação, revisão por pares de artefatos de programação e desempenho em lições sobre aspectos pedagógicos	NI	finlandês	Koodiaapinen MOOC
(MARTINEZ *et al.*, 2016)	Lições (exercícios) baseadas em questionários e exemplos de planos de aulas	Auto-avaliação por meio de pré e pós questionários e observações em sala de aula	NI	espanhol	NI
(KAY & MOSS, 2012) (KAY *et al.*, 2014)	Material sobre o programa de Robótica da FIRST LEGO League (FLL) para crianças de 9 a 14 anos de idade	Questionários de avaliação do conhecimento dos conteúdos antes e depois do workshop	NI	inglês	NI
(COOPER *et al.*, 2011) (COOPER *et al.*, 2015)	Tutoriais sobre Alice, exemplos de planos de aula, vídeos e desafios disponibilizados via o site do projeto	Apresentação do plano de ensino e feedback dados pelos outros professores em relação ao plano	Visitas durante o ano seguinte para ajudar na aplicação	inglês	grátis

(LIU *et al.*, 2015)	Exemplos no SNAP!	Testes antes e depois de cada dia	NI	inglês	NI
(LIU *et al.*, 2013)	Programas exemplo e tarefas (exercícios)	Quiz pré e pós sessão todos os dias	NI	inglês	NI
(VIVIAN, FALKNER & FALKNER, 2014)	Conteúdo na web, vídeos conceituais e exemplos trabalhados ligados aos objetivos de aprendizagem do currículo australiano	Criação de recursos de ensino, um plano de aula e tarefas para cada seção	Comunidade Google+ para troca de experiências	inglês	NI
(HAMNER *et al.*, 2016)	NI	Artefato criado	NI	inglês	NI
(HADEN *et al.*, 2016)	Slides, exercícios e projeto de programação completo (exemplo)	NI	NI	inglês	NI
(LIU *et al.*, 2014)	Exemplos de jogos	Questionários pré e pós-sessão no início	NI	inglês	NI
(RUSU, 2015)	NI	NI	NI	inglês	NI
(GELDREICH, TALBOT & HUBWIESER, 2018)	Handouts, exercícios e material de ensino	NI	Recebem material online e possibilidade de procurar apoio	alemão	NI
(VON WANGENHEIM *et al.*, 2015)	Slides, vídeos, exemplos, tutorial online e robô interativo.	Não há avaliação dos participantes	NI	português do Brasil	Creative Commons
(ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009)	NI	Feedback em intervalos regulares através das observações	Suporte online durante o curso	inglês	NI
(CHO *et al.*,2014)	Slides e atividades práticas	NI	NI	inglês	NI
(LIU *et al.*, 2011)	Slides	NI	Suporte nos semestres seguintes	inglês	NI

Fonte: elaborada pela autora.

Tabela I.3. Visão geral das características de contexto das UI

ID	**Área de conhecimento dos professores que participam na UI**	**Nível de ensino dos professores que participam na UI**	**Pré-requisitos**
(PARTANEN, MANNILA & PORANEN, 2016)	Matemática	Ensino Fundamental	NI
(MARTINEZ *et al.*, 2016)	Qualquer disciplina	Ensino Fundamental e Médio	Professores que estão dispostos a ensinar

			computação.
(KAY & MOSS, 2012) (KAY *et al.*, 2014)	Qualquer disciplina	Ensino Fundamental e Médio	professores k-12
(COOPER *et al.*, 2011) (COOPER *et al.*, 2015)	Qualquer disciplina	Ensino Fundamental e Médio	Pouco ou nenhum conhecimento de computação.
(LIU *et al.*, 2015)	Qualquer disciplina	Ensino Fundamental e Médio	professores k-12
(LIU *et al.*, 2013)	Qualquer disciplina	Ensino Fundamental e Médio	NI
(VIVIAN, FALKNER & FALKNER, 2014)	Qualquer disciplina	Ensino Fundamental e Médio	NI
(HAMNER *et al.*, 2016)	Disciplinas tradicionais (como Inglês, História ou Ciência)	Ensino Fundamental	professores k-12
(HADEN *et al.*, 2016)	Qualquer disciplina	Ensino Fundamental e Médio	professores com pouco ou nenhum conhecimento de computação
(LIU *et al.*, 2014)	Qualquer disciplina	Ensino Fundamental e Médio	professores com pouco ou nenhum conhecimento de computação
(RUSU, 2015)	Qualquer disciplina	Ensino Fundamental Anos Finais e Médio	NI
(GELDREICH, TALBOT & HUBWIESER, 2018)	Qualquer área de conhecimento	Ensino Fundamental	sem nenhum conhecimento prévio
(VON WANGENHEIM *et al.*, 2015)	Qualquer área de conhecimento e professores que fornecem TI apoio nas escolas	Ensino Fundamental	saibam usar computadores, mas não necessariamente saber programar.
(ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009)	2 Matemáticos, 1 Físico, 5 Engenheiros, 8 Professores de Informática e 4	Ensino Fundamental e em formação	com ou sem conhecimento prévio

	Professores do Ensino Primário		
(CHO *et al.*,2014)	(1) Artes da Linguagem (4), Estudos Sociais (3) e Espanhol (1) se inscreveram, e os professores de disciplinas não relacionadas ao tema representaram 2/3 de todos os participantes do curso (2) estudos sociais, artes da linguagem, língua estrangeira e outras que não são tipicamente associadas a assuntos STEM	Ensino Fundamental e Médio	NI
(LIU *et al.*, 2011)	matemática, laboratório de informática, leitura, ciências, biologia e até mesmo educação especial	Ensino Fundamental e Médio	NI

Fonte: elaborada pela autora.

Tabela I.4. Métodos de desenvolvimento das UI

ID	**Base teórica / método de desenvolvimento**
(PARTANEN, MANNILA & PORANEN, 2016)	Um primeiro curso foi organizado como uma versão beta para testar a adequação dos materiais e tecnologias, e também para obter feedback contínuo dos participantes para cada seção. O segundo curso foi organizado com os arranjos e conteúdos modificados com base no feedback e nas experiências. A implementação da trilha de Racket foi inspirada pelo Projeto do Programa Sistemático curso online oferecido pela edx.org. O material do curso continha elementos semelhantes para cada semana embora o programa do curso e os exercícios fossem diferentes.
(MARTINEZ *et al.*, 2016)	NI
(KAY & MOSS, 2012) (KAY *et al.*, 2014)	NI
(COOPER *et al.*, 2011) (COOPER *et al.*, 2015)	NI
(LIU *et al.*, 2015)	NI

(LIU *et al.*, 2013)	Os programas de extensão de professores do ensino fundamental e médio têm se mostrado um meio eficaz de tornar a ciência da computação acessível a esses professores. Uma pesquisa foi conduzida pela Universidade Lamar em programas de treinamento de professores de informática K-12 em 2011 [12]. A Carnegie Mellon University ofereceu os programas CS4HS e CSbots, que visavam professores de ciências da computação do ensino médio [2] [10]. Em 2010, a Marquette University liderou o workshop “Pensamento Computacional para as Ciências”, um workshop de verão de três dias para professores de ciências e matemática do ensino médio [1]. A Lamar University ofereceu oficinas de verão para professores do ensino fundamental e médio em 2010 [11]. Durante o verão de 2008, a Duke University realizou um curso de treinamento de Alice de três semanas para professores do ensino fundamental [13]. A Saint Joseph's University tornou-se o local no qual o curso de Enriquecimento de Professores em Ciência da Computação foi baseado [7]. O Instituto de Tecnologia da Geórgia também se empenhou em melhorar o envelhecimento do oleoduto da educação em informática atualmente empregado no estado, oferecendo uma série de oficinas de professores, começando em 2006, para influenciar estilos de ensino de RS de maneira positiva [3]. No entanto, com base no conhecimento adquirido, o App Inventor não foi usado em nenhum programa de educação básica para professores antes de 2012. Em 2009, o Google lançou um programa piloto com dez universidades onde o App Inventor foi ministrado como um curso básico [13]. No verão de 2011, a Universidade Estadual de Valdosta ofereceu um acampamento de verão do App Inventor para estudantes do ensino médio [9].
(VIVIAN, FALKNER & FALKNER, 2014)	Desenvolvimento em 3 etapas: 1. Revisão CS & MOOCs, 2. Desenvolvimento & Design, 3. Implementação & Revisão. Para determinar o que literatura pedagógica empírica estava disponível para a educação CS, começamos por realizar uma revisão semi-sistemática de trabalhos de pesquisa sobre CS educação, implementado para crianças entre as idades de 5 e 18, usando Simon sistema (2007) para determinar (1) o assunto ensinado; (2) a faixa etária estudada; e (3) métodos de coleta de dados. O desenvolvimento de conteúdo foi alcançado através de sessões de brainstorming entre a equipe, de diversas áreas, experiência e conhecimento. Foram adaptadas ideias de lições de organizações e iniciativas existentes, como CS Unplugged e Code.org, e ideias de lições e abordagens da educação textos em outras áreas de aprendizagem, como Matemática, Ciências e Alfabetização, desenvolvido por uma equipe de pesquisadores em educação e palestrantes, usando também Google Course Builder. O trabalho foi realizado em contato com desenvolvedores de currículo e elaboração de currículo para alinhar conteúdo e idéias aos objetivos australianos de aprendizado.
(HAMNER *et al.*, 2016)	Modelos de treinamento já existentes na literatura e experiências em edições anteriores do projeto.
(HADEN *et al.*, 2016)	Nossas oficinas são cuidadosamente projetadas para garantir que qualquer conteúdo conhecimento ou técnicas que os professores aprendem podem ser transferidos para suas salas de aula após o término do workshop. Os projetos de programação são baseados em exemplos encontrados em livro Starting from Scratch, por Jeremy Scott [29].
(LIU *et al.*, 2014)	Baseado em diversos trabalhos realizados por universidades americanas.
(RUSU, 2015)	NI

(GELDREICH, TALBOT & HUBWIESER, 2018)	Seguindo a metodologia da pesquisa baseada em design [34], levou-se em conta princípios teóricos e pesquisas anteriores de várias disciplinas, incluindo não apenas a educação em ciência da computação, mas também psicologia do desenvolvimento, ciência cognitiva e formação de professores. Isso resultou nos seguintes princípios básicos aos quais nos alinhamos o desenho da oficina de formação de professores.
(VON WANGENHEIM *et al.*, 2015)	O estudo de caso é realizado de acordo com o procedimento proposto por Yin (13) e Wohlin et al. (12): O estudo é definido em termos de objetivos, questões de pesquisa e desenho de pesquisa. A partir do objetivo, as questões e medidas de análise são sistematicamente derivadas usando a abordagem Meta / Pergunta / Métrica (GQM) (Basili et al., 94). Os instrumentos de coleta de dados são definidos com relação às medidas. A execução do estudo é realizada adotando o ADDIE (Branch, 09) como abordagem de design instrucional. Primeiro, a unidade de instrução é desenvolvida. Portanto, os alunos e o ambiente instrucional são caracterizados. As necessidades de aprendizagem são eliciadas e os objetivos de aprendizagem são definidos. De acordo com o contexto, a instrutiva estratégia é delineada, definindo seu conteúdo, sequência e métodos instrucionais a serem adotados. Instrucional o material é desenvolvido de acordo com a estratégia instrucional. Então, a unidade de instrução é aplicada no treinamento sessões e avaliadas, coletando dados conforme definido pela definição do estudo. Os dados coletados são analisados em relação às questões de pesquisa, utilizando métodos quantitativos e qualitativos. Então, os resultados são interpretados e discutidos.
(ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009)	A ideia introduzida por Resnick de “learning by design”, que é central na pedagogia construtivista
(CHO *et al.*,2014)	Desde os trabalhos seminais de Wing sobre o pensamento computacional (Wing, 2006, 2008), muitas abordagens aplicaram essa estrutura a novos estudos e workshops. As oportunidades mais óbvias estão nos cursos de matemática e ciências, mas Lu e Fletcher argumentam que os estudiosos das ciências sociais e humanas estão descobrindo que os processos de computação também podem avançar em suas disciplinas (Lu & Fletcher, 2009). Settle et al. sugerem que modificar o currículo K-12 para incluir uma ênfase mais forte no pensamento computacional causará um impacto maior na competência computacional (Settle et al., 2012). Um programa que obteve grande sucesso na introdução de conceitos de ciência da computação para professores de nível pré-universitário é o programa Google CS4HS (http://cs4hs.com) que começou na Universidade Carnegie Mellon em 2006.
(LIU *et al.*, 2011)	Em 2004, a UCLA iniciou um programa anual chamado de Instituto de Verão para Professores de Ciências da Computação em Colocação Avançada (APCS). Em 2010, a Marquette University liderou o workshop Pensamento Computacional para as Ciências, um workshop de verão de três dias para professores de ciências e matemática do ensino médio. A Saint Joseph's University tornou-se o local no qual o curso de Enriquecimento de Professores em Ciência da Computação foi baseado no Error! Fonte de referência não encontrada. O Georgia Institute of Technology (Instituto de Tecnologia da Geórgia) esforçou-se para melhorar o antigo pipeline educacional de informática atualmente empregado no estado da Geórgia, oferecendo uma série de oficinas de professores para influenciar estilos de

	ensino de CS de maneira positiva. A Carnegie Mellon University ofereceu os programas CS4HS e CSbots que visavam professores de ciências da computação do ensino médio.

Fonte: elaborada pela autora.

Tabela I.5. Visão geral da avaliação da qualidade das UI

ID	Design de pesquisa	Fator(es) avaliado (s)	Método(s) de coleta de dados	Tamanho da amostra	Replicação do estudo	Método(s) de análise de dados	Resultados
(PARTANEN, MANNILA & PORANEN, 2016)	Estudo de caso	Nível de dificuldade, utilidade do conteúdo, entusiasmo dos participantes	Artefatos criados e autoavaliação	540 professores	Replicado	NI	O feedback do curso da primavera de 2016 foi positivo e indicou que o nível de dificuldade e a carga de trabalho estavam se tornando razoáveis. O conteúdo do curso também foi percebido como adequado e útil. Além disso, o curso parecia criar uma quantidade razoável de entusiasmo.
(MARTINEZ *et al.*, 2016)	Estudo de caso	Aprendizagem e participação dos professores	Questionários e observação	106 professores	Replicado	Análise qualitativa	No geral os professores têm ideias errôneas sobre ciências da computação, e muitos acreditam que não precisam ensinar. Aprender a pedagogia é tão importante quanto aprender conceitos de computação porque simplesmente ensinar algoritmos e definições não pode promover uma aprendizagem significativa da disciplina. No entanto, esse treinamento foi insuficiente para preparar professores sem experiência anterior. Não ficou claro o melhor cenário para ensinar ciência da computação. Os professores foram mais propensos a replicar as mesmas atividades que experimentaram durante as oficinas em suas salas de aula do que a produzir suas próprias.
(KAY & MOSS, 2012) (KAY *et al.*, 2014)	Estudo de caso	Atitude, capacidade de ensinar e satisfação dos professores	Questionários	44 professores	Replicado	Análise quantitativa (teste de hipótese)	Sua confiança aumentou dramaticamente e eles tinham uma forte expectativa de que usariam o material com seus alunos. Uma pesquisa de acompanhamento nove meses depois indicou que centenas de estudantes e muitos colegas foram afetados apenas no primeiro ano. Os participantes mostraram um aumento estatisticamente significativo ($p < 0,01$) em sua confiança relatada. A eficácia de ensino estatisticamente significativa ($p < 0,05$) aumentou de antes para depois do seminário. Estatisticamente significativo ($p < .01$) aumento em conhecimento e habilidades em programação de robôs LEGO. O conhecimento de programação foi aprimorado como resultado do workshop por meio de pré e pós testes.

(COOPER *et al.*, 2011) (COOPER *et al.*, 2015)	Ad-hoc	Conhecimento em programação, satisfação dos professores e aprendizagem	Questionários e artefatos criados	> 100	Não replicado	Análise quantitativa (teste de hipótese)	Aumento significativo em conhecimento de programação e de Alice, demonstrado por um exame de conteúdo administrado como um pré e pós-teste no início e na conclusão do programa. Os professores sentiram que os tópicos e métodos instrucionais abordados nas oficinas adequado para eles ensinarem seus alunos, e que a interação com professores universitários e estudantes universitários parceiros satisfizesse suas necessidades. Os professores sentiram que Alice tornou a programação divertida e mais fácil de ensinar aos alunos. Os professores identificaram uma miríade de benefícios para uso em sala de aula de Alice. Mais de 90% dos professores indicaram planos para continuar usando Alice em suas salas de aula. 80% dos professores relataram que usaram o que aprenderam durante as oficinas de verão em suas salas de aula durante os anos subsequentes. O apoio do diretor é muito importante a formação dos professores em em computação. Discussão entre grupos de professores também é importante.
(LIU *et al.*, 2015)	Estudo de caso	Conhecimento de computação, satisfação dos professores dificuldade do material instrucional	Questionários e entrevistas	13 professores	Não replicado	NI	A pontuação média do conhecimento de computação na pesquisa pré-workshop foi de 2,52 de 5, o que aumentou para 3,76 na pesquisa pós-workshop, e a pontuação média de conhecimento de computação melhorou em 25%. 92,31% dos participantes classificaram o workshop como bom ou excelente. 30,77% dos participantes avaliaram o material como difícil, 61,54% deles consideraram os materiais moderados e cerca de 7,69% dos participantes acharam um pouco fácil.
(LIU *et al.*, 2013)	Estudo de caso com pré e pós-teste	Conhecimento de computação, sugestões de melhoria	Questionários, auto-avaliação do grau de conhecimento e observações	14 participantes	Não replicado	Análise quantitativa (estatística descritiva)	Os resultados da avaliação do workshop mostram um aumento de 29,5% nível de confiança no ensino de informática e de 109% aumento do nível de conhecimento de computação.
(VIVIAN, FALKNER & FALKNER, 2014)	Ad-hoc	Participação e experiência dos participantes	Questionários e artefatos criados	1378 professores	Não replicado	NI	Dos 1378 inscritos no curso, 99 participantes completaram o curso e 438 apenas se inscreveram. Como resultado, temos um 7,2% taxa de conclusão, ou 10,5% taxa de conclusão para aqueles que foram em frente e começou a curso. 56,39% dos participantes completando menos da metade do curso. Em termos de componentes principais apenas, 8,13% dos participantes exploraram metade ou mais dos componentes

							principais (sem conclusão) e 52,3% dos professores (grupo de participantes do MOOC) explorou menos de 50% dos componentes principais. Nossa taxa de conclusão para exploradores foi de 55,7% e 46,9% ao considerar apenas componentes principais. 26 de 50 que responderam o survey online disseram que o curso influenciou muito suas percepções, já que eles não estavam cientes das oportunidades de carreira e 16 disseram que aprenderam mais do que eles sabiam anteriormente, sendo que vários participantes relataram que os tópicos posteriores foram mais desafiadores.
(HAMNER *et al.*, 2016)	Estudos de caso	NI	Questionários, entrevistas e observações de aula.	222 professores	Replicado	NI	Os resultados sugeriram três áreas nas quais a DP poderia ser aprimorada: construção e programação, integração de robótica em conteúdo de classe e identificação de talentos. Professores avaliaram os workshops como benéficos para o ensino, com 100% indicando que isso melhorou seu ensino. A escala PMTE (escala Personal Mathematics Teaching Efficacy) mostrou um efeito significativo do nos escores pré e pós, na avaliação da eficácia (t (14) = 2,234, p <0,042)
(HADEN *et al.*, 2016)	Estudo de caso	Capacidade de ensinar	Questionário	14 professores	Replicado	NI	Os resultados preliminares são positivos, com a maioria dos professores desenvolvendo a capacidade de transferir o treinamento para suas próprias salas de aula. Depois de oficinas, os professores continuam pedindo apoio, especialmente materiais adicionais prontos para sala de aula. Nós mantemos que o treinamento em serviço deve incluir esse suporte contínuo.
(LIU *et al.*, 2014)	Estudo de caso	Conhecimento de computação	Questionários e entrevista	16 professores	Não replicado	NI	Os resultados da avaliação mostram que a pontuação média de Stencyl conhecimento no inquérito pré-workshop foi de 1,26 em 5, o que foi aumentada para 3,76 na pesquisa pós-seminário. A pontuação média do conhecimento de computação foi melhorada em 61%.
(RUSU, 2015)	Estudo de caso	Qualidade da oficina, sugestões de melhoria, capacidade de ensinar, atitude e conhecimento de	Questionários	37 professores	Replicado	Análise qualitativa	Os participantes gostaram da “disponibilidade imediata de materiais ”apresentados, que os tópicos eram“ oportunos e informação atual, muitas ideias diferentes para o que fazer em sala de aula ”foram dadas, que“ tempo de computador para testar e utilizar a informação ”foi

		computação					alocada durante o workshop, e “compartilhar informações e comunicar idéias” com outras professores tem sido uma prioridade. A maioria dos comentários da pesquisa pós-atitudinal, quando em comparação com as respostas da pesquisa pré-atitudinal, mostraram muita mudança na forma como os professores participantes viram Ciência da Computação e sua relação com outros campos. Comparando a compreensão dos professores sobre o pensamento computacional antes e depois do workshop indica que os professores deixaram a oficina melhor preparada para infundi-lo em suas aulas.
(GELDREICH, TALBOT & HUBWIESER, 2018)	Estudo de caso	Capacidade de ensinar computação e satisfação	Questionário, entrevistas e diários de ensino	38 professores	Não replicado	Análise quantitativa e qualitativa usando estatística descritiva	33 professores de 38 afirmaram que estavam satisfeitos com o workshop em geral, 5 declararam que estavam bastante satisfatórios. Quando perguntados se suas expectativas em relação aos benefícios para a implementação na escola foram cumpridas, 17 afirmaram inteiramente, 18 declarou em grande medida e 3 afirmou em partes. No questionário, os professores também responderam várias questões relativas à auto eficácia em relação à implementação dos tópicos do workshop. Aqui pode-se ver que os professores são mais confiantes sobre tarefas desconectadas do que sobre a programação no computador. Os resultados também mostram que as pessoas não se sentem totalmente à vontade para responder às perguntas dos alunos, ensinando os novos tópicos a alunos problemáticos e ajudar os alunos na programação individual.
(VON WANGENHEIM *et al.*, 2015)	Estudo de caso	Dificuldade, conhecimento adquirido, entusiasmo de ensinar computação, sugestões de melhoria e aprendizagem	Questionários	16 professores	Não replicado	Análise quantitativa e qualitativa usando estatística descritiva	Resultados preliminares com professores são positivos, motivando a maioria dos participantes a introduzir computação em suas classes. No entanto, os resultados também destacam que, para permitir que o professor aplique oficinas de forma eficaz, cursos de treinamento mais longos e suporte contínuo são necessários.
(ALIMISIS, FRANGOU & PAPANIKOLAOU, 2009)	Estudo de caso	Qualidade do curso, aprendizagem e capacidade de ensinar	Questionário e entrevista coletiva.	23 professores	NI	NI	Os professores gostaram das atividades práticas, a criação de suas próprias estruturas de engenharia e sua programação. Parece, de fato, que eles gostaram de seu trabalho. Já a partir das atividades iniciais algumas delas

							começaram a pensar em cenários para inclusão de atividades similares em sua escola.
(CHO *et al.*,2014)	Estudo de caso	Utilidade de adquirir habilidades em computação	Questionários	47 professores	Replicado	Análise quantitativa	As maiores diferenças foram vistas em as declarações sobre os requisitos de fundo de matemática (Ano 1, Ano 2) (1.50, -0.70), o grau de interações sociais de cientistas da computação (-1,31, -0,79). As menores diferenças foram observadas em declarações sobre se a ciência da computação era importante na sociedade em relação a outras disciplinas (-0,28, -0,40) e o (im) equilíbrio de gênero de homens e mulheres no campo da ciência da computação (-0,45, +0,17).
(LIU *et al.*, 2011)	Estudo de caso	Competência e conhecimento em computação	Questionário e entrevista por avaliador externo	7 professores	NI	NI	Na avaliação pré-workshop, os resultados indicam uma média do nível de confiança de 19 de 50. No entanto, a pontuação média de avaliação pós-workshop foi de 42 de 50, indicando um aumento de 121% no nível de confiança. Uma mudança semelhante foi observada na análise dos pré- / pós-testes (classificados por instrutores de oficina) para cada sessão. Os participantes obtiveram uma média de 2,77 de 5 nos testes pré-sessão, mas essa pontuação subiu para 4,24 no teste pós-sessão, o que resultou em um aumento de nível de conhecimento de 62,76%.

Fonte: elaborada pela autora.

APÊNDICE II

Tabela II.1. Questionário utilizado para autoavaliação dos professores do Curso “Aprenda a ensinar algoritmos e programação”

Número	Pergunta	Possíveis respostas			
1	O que você achou desse módulo?				
2	O módulo foi	□ Muito fácil	□ Fácil	□ Difícil	□ Muito difícil
3	O módulo foi	□ Muito divertido	□ Divertido	□ Chato	□ Muito chato
4	O tempo de cada aula desse módulo passou	□ Muito rápido	□ Rápido	□ Devagar	□ Muito devagar
5	O módulo foi	Excelente 	Bom 	Regular 	Ruim 
6	O que mais gostei no módulo foi				
7	O que menos gostei no módulo foi				
8	Consigo fazer um programa de computador	□ Sim	□ Não		
9	Vou ensinar esse conteúdo na(s) minha(s) disciplina(s)	□ Sim	□ Não		
10	Consigo explicar para os alunos como fazer um programa de computador	□ Sim	□ Não		
11	Quero aprender mais sobre como fazer programas de computador	□ Sim	□ Não		

12	Fazer um programa de software é	□ Muito fácil	□ Fácil	□ Difícil	□ Muito difícil
13	Considero o ensino de computação na Educação Básica importante	□ Sim	□ Não		
14	Quer fazer mais algum comentário?				

Fonte: elaborada pela autora.

APÊNDICE III

Artigo no formato SBC.

Desenvolvimento de uma Unidade Intrucional para Formação de Professores da Educação Básica para o Ensino de Computação

Fabíola Maria Kretzer[1], Nathalia da Cruz Alves[1], Mirian Nathalie F. Ferreira[1], Christiane Gresse von Wangenheim[1], Jean Carlo R. Hauck[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina - Florianópolis, SC - Brazil

fabiola.kretzer@grad.ufsc.br, nathalia.alves@posgrad.ufsc.br, nathalie.fortuna@posgrad.ufsc.br, c.wangenheim@ufsc.br, jean.hauck@ufsc.br

Abstract. *Today, computing has become increasingly influential in our day-to-day lives. Due to this importance, computing needs to be popularized in K-12 giving students the opportunity to learn basic computing competencies. Therefore, this implementation of computer teaching at school requires teachers with computational, pedagogical, and technological skills in order to teach computing in an effective and motivating way. However, few teachers graduate from undergraduate computer courses. An alternative is to provide these skills to teachers in other areas who are working in the classroom through continuing education. In this context, an article presents the systematic development of a course to teach concepts of algorithms and programming to teachers in basic education. Results from a first application indicate that the course can contribute to the continuing education of teachers effectively and at the same time provide a fun and enjoyable experience. It is hoped that the results of this research may contribute to the expansion of computer education in Brazilian schools, supporting the popularization of this knowledge.*

Resumo. *Atualmente, a computação está se tornando cada vez mais influente no dia-a-dia dos cidadãos. Devido a essa importância, a computação precisa ser popularizada na Educação Básica, dando aos alunos a oportunidade de aprender competências básicas em computação. Porém, essa implementação de ensino de computação na escola exige professores com competências computacionais, pedagógicas e tecnológicas, a fim de ensinar computação de uma forma eficaz e motivadora. No entanto, poucos professores se formam em cursos de licenciatura em computação. Uma alternativa é proporcionar essas competências a professores de outras áreas que estejam atuando em sala de aula por meio de formação continuada. Neste contexto, artigo apresenta o desenvolvimento sistemático de um curso para ensinar conceitos de algoritmos e programação aos professores na Educação Básica. Resultados de uma primeira aplicação indicam que o curso pode contribuir a formação continuada dos professores de forma eficaz e ao mesmo tempo proporcionar uma experiência agradável e divertida. Espera-se que os resultados da presente pesquisa possam contribuir para a ampliação do ensino de computação nas escolas brasileiras apoiando a popularização desse conhecimento.*

1. Introdução

A influência da computação é sentida pelas pessoas diariamente e experimentada em um nível pessoal, social e global [CSTA 2016]. Num mundo rodeado de tecnologia, os cidadãos bem preparados, precisam ter uma compreensão clara dos princípios e práticas da computação, o que vai além de simples uso de TI (Tecnologia da Informação) [CSTA 2016]. Consequentemente, é importante que a pessoas aprendam sobre computação desde a infância. Atualmente as escolas ensinam apenas alfabetização digital, a qual possui o foco de ensinar os alunos a utilizar recursos tecnológicos, tornando um consumidor de TI [Tissenbaum, Sheldon & Abelson 2019]. Porém, o que deve ser ensinado nas escolas é a proficiência digital [Tissenbaum, Sheldon & Abelson 2019] que faz o aluno desenvolver várias habilidades do século XXI, incluindo criatividade, resolução de problemas e colaboração [Gal-Ezer & Stephenson 2010]. Essas competências estimulam os alunos a criar, inovar e prosperar em um ambiente de mudanças rápidas e contínuas [Naughton 2012]. Assim, se torna importante o desenvolvimento de habilidades computacionais já na Educação Básica.

Existem diversas iniciativas que buscam ensinar computação a crianças e adolescentes na Educação Básica, com foco em programação [Grover & Pea 2013] explorando diversas maneiras de ensino, como MOOC [De Kereki & ManatakiI 2016], oficinas [Robinson & Pérez-Quinõnes 2014; Ferreira et al. 2019], cursos [Ferreira et al. 2019; Ferreira et al. 2019], entre outros. Essas iniciativas geralmente utilizam linguagens de programação baseada em blocos, como App Inventor [MITa 2019] ou Scratch [MITb 2019], as quais não exigem aprendizagem de sintaxe e semântica complexas, como nas linguagens textuais. Porém, a implementação de ensino de computação nas escolas exige professores motivados e dedicados com competências computacionais, pedagógicas e tecnológicas, a fim de ensinar computação de uma forma que realmente envolva os alunos [Gal-Ezer & Stephenson 2010].

A implementação de ensino de computação na escola exige professores motivados, dedicados e com competências computacionais, pedagógicas e tecnológicas, a fim de ensinar computação de uma forma que realmente envolva os alunos [Gal-Ezer & Stephenson 2010; Goode 2008; Bower et al. 2017]. Idealmente, os professores de computação seriam preparados por meio de um rigoroso programa de formação de professores, normalmente por meio de programas de graduação ou pós-graduação [Berges et al., 2013] como cursos de licenciatura em computação [INEP 2018]. Porém, as matrículas nesses programas são cada vez mais reduzidas pelo crescimento contínuo do emprego na indústria, já que o incentivo para seguir carreiras lucrativas na indústria de TI é muito maior do que o incentivo para a licenciatura [Goode 2007]. Por exemplo, em 2018 no Brasil o número de concluintes em cursos para formação de professores de computação foi de 0,1% do total de concluintes de curso de licenciatura, muito menor do que de outras áreas, como por exemplo matemática, que possui 10% do total [INEP 2018]. Como resultado, professores formados em computação no nível da Educação Básica são escassos [Ni & Guzdial 2012; INEP 2018). Dessa forma, é importante recrutar e ensinar computação a professores *in-service* que possuem credenciais em outras áreas de estudo [Goode 2008; Cooper et al. 2015]. Consequentemente, muitos esforços atuais estão focados na formação continuada de professores *in-service* [Lamprou et al. 2017] para a computação ser integrada de maneira interdisciplinar [Lye & Koh 2014; Gresse von Wangenheim et al. 2017].

Esses cursos de formação continuada focam principalmente no ensino de conceitos de algoritmos e programação, adoram abordagens ativas de aprendizado, levando os participantes a criar artefatos e resolver problemas relacionados a computação [Liu et al. 2013]. No entanto poucos cursos abordando conhecimento pedagógico e uma quantidade menor que abordam conhecimento tecnológico, com alguns casos que pedem para o professor desenvolver o seu próprio material educacional para o uso posterior nas suas disciplinas [Liu et al. 2013]. Assim, esse artigo apresenta o desenvolvimento de um curso para proporcionar aos professores *in-service* (atuantes em outras disciplinas como, artes, história, geografia e outras) a oportunidade de aprender competências básicas de computação, bem como aspectos pedagógicos e tecnológicos, os quais permitam a integração do ensino de computação as disciplinas nessas respectivas áreas. O curso segue o *framework* TPACK (Conhecimento Tecnológico, Pedagógico e do Conteúdo) [Mishra & Koehler 2006] para formação de professores e o conteúdo é alinhada a diretrizes de currículo para o ensino de computação na Educação Básica [CSTA 2016; SBC 2018; Alves 2019].

2. Metodologia de pesquisa

O objetivo desta pesquisa é o desenvolvimento, a aplicação e a avaliação de um curso de formação continuada de professores para o ensino de computação de forma multidisciplinar no Ensino Fundamental. Para atingir este objetivo é realizado um estudo de caso exploratório para compreender os fenômenos observados durante a aplicação dos dois primeiros módulos do curso em um contexto particular e identificar direcionamentos para trabalhos futuros.

Definição do estudo: O estudo é definido em termos do objetivo e perguntas de pesquisa e o design de pesquisa. A partir do objetivo e das perguntas de análise são sistematicamente derivadas as medidas para a coleta de dados utilizando o método GQM [Basili, Caldiera & Rombach 1994]. Para a operacionalização da coleta de dados são definidos instrumentos de coleta de dados para todas as medidas definidas.

Execução do estudo: A execução do estudo é realizada adotando o modelo ADDIE [Branch 2009] como abordagem para o design instrucional. Em uma primeira etapa o curso é desenvolvido. Para isto, são primeiramente caracterizados os aprendizes e o ambiente em que curso acontecerá. São então levantadas as necessidades de aprendizagem e com base nessas informações são definidos os objetivos de aprendizagem. De acordo com a análise de contexto, é projetado o curso, definindo o seu conteúdo, a sequência e os métodos instrucionais a serem adotados. Em seguida, o material instrucional é desenvolvido. Durante a segunda etapa da execução do estudo, o curso é aplicado na prática e avaliado, coletando-se os dados conforme a definição do estudo.

Análise e Interpretação do estudo: Nesta etapa são analisados os dados em relação as perguntas de pesquisa, usando métodos quantitativos e qualitativos. Ao final, os resultados são interpretados e discutidos.

3. Desenvolvimento do curso “Aprenda a ensinar algoritmos e programação”

O curso para o ensino multidisciplinar de computação é desenvolvido seguindo uma abordagem de design instrucional [Branch 2009], incluindo as principais etapas de: Análise do contexto do

curso, Design do curso e o Desenvolvimento do material instrucional. As etapas de elaboração são descritas a seguir.

3.1 Análise do contexto

Seguindo o design instrucional são inicialmente estudados os perfis dos aprendizes e o ambiente e assim, a partir desses, definidos os objetivos de aprendizagem.

Aprendizes: são professores da Educação Básica das escolas brasileiras, os quais, majoritariamente, possuem idade entre 35 e 54, com aproximadamente 11 à 30 anos de carreiras. Esses professores possuem nível superior completo em áreas como matemática e história, mas normalmente não possuem formação em computação.

Ambiente: são escolas brasileiras que possuem sala informatizada com computadores (tipicamente com computadores e equipamentos/*softwares* não atualizados) e acesso à internet (geralmente com baixa velocidade). Os alunos da Educação Básica também são indiretamente afetados. Estes possuem capacidade de utilizar e controlar dispositivos eletrônicos, bem como o acesso à internet, porém tipicamente não possuem conhecimentos em computação.

Objetivos de aprendizagem: O objetivo geral do curso é formar professores *in-service* da Educação Básica para ensinar computação principalmente nos Anos Finais do Ensino Fundamental em alinhamento aos principais diretrizes de currículo CSTA (2016) e SBC (2018). Os objetivos de aprendizagem também focam na importância dos conceitos de algoritmos e programação na área de computação o curso foca nesses conceitos por meio de atividades de programação com Code.org [Code.org 2019], desenvolvimento de jogos com Scratch [MITb, 2019] e desenvolvimento de aplicativos com App Inventor [MITa, 2019]. Adicionalmente, também são incluídos objetivos de aprendizagem relacionados ao conhecimento pedagógico e tecnológico, conforme sugerido por Mishra e Koehler (2006).

3.2 Design do curso

De acordo com os objetivos de aprendizagem definidos, foi definido e sequenciado o conteúdo da unidade instrucional, o qual foi dividido em 5 módulos (Tabela 1), com uma duração de aproximadamente 40 h/a. O curso é voltado para o ensino de conceitos básicos da computação principalmente voltado a algoritmos e programação (como eventos, condicionais e laços) por meio de desenvolvimento de atividades de programação com Code.org [Code.org 2019], desenvolvimento de jogos com Scratch [MITb 2019] e desenvolvimento de aplicativos com App Inventor [MITa 2019]. Durante o desenvolvimento dos jogos e aplicativos são ensinadas práticas de engenharia de *software*, *design thinking* e *User eXperience*. E, também aborda conhecimento tecnológico, pedagógico e do conteúdo, conforme sugerido por Mishra e Koehler (2006).

Tabela 1. Conteúdos ensinados em cada módulos do curso.

Módulo	Duração (h/a)	Ambiente de programação	Conteúdo
1	4	Code.org	Atividades fechadas de programação para ensinar algoritmos e programação (condicionais, laços e movimento)
2.A	10	Scratch	Desenvolvimento de um jogo para *desktop* utilizando um

			processo sistemático da engenharia de *software*, algoritmos e programação (eventos, variáveis, condicionais, operadores, sensores, laços, movimento, aparência e som)
2.B1	10	App Inventor	Desenvolvimento de um aplicativo completo para *Android* utilizando tutoriais que abordam conceitos de algoritmos e programação (eventos, nomeação, *strings*, mapa, variáveis, condicionais, operadores abstração, sensores, bibliotecas externas, laços, listas e persistência)
2.B2	20	App Inventor	Desenvolvimento de um aplicativo para *Android* utilizando os conceitos aprendidos no módulo 2.B1 e um processo sistemático da engenharia de *software*, juntamente com conceitos de *design thinking e User eXperience*
3	4	Code.org, Scratch e App Inventor	Guias de currículo para o ensino de computação na Educação Básica, métodos de ensino (como ensinar computação de forma interdisciplinar) e estratégias para avaliar a compreensão dos alunos
4	2	Code.org, Scratch e App Inventor	Habilidades para revisar/criar e utilizar a infraestrutura técnica necessária para realizar as aulas com os ambientes de programação, além de questões práticas de tecnologia no ensino de computação

O curso adota várias estratégias instrucionais, incluindo aulas expositivas, exercícios e atividades práticas. As atividades práticas adotam a ação computacional [Tissenbaum, Sheldon & Abelson 2019], visando uma aprendizagem baseada em problemas, abrangendo desde a parte do levantamento do problema, definição de solução até o desenvolvimento de um jogo/aplicativo funcional. A ação computacional propõe que ao mesmo tempo em que os estudantes aprendem sobre computação, também têm a oportunidade de criar soluções computacionais (como apps e jogos) que tenham impacto direto em suas vidas e em suas comunidades [Tissenbaum, Sheldon & Abelson 2019]. Assim, o curso prevê que os professores desenvolvam seus próprios projetos de jogos e criem soluções computacionais que possam ser utilizadas em suas aulas, motivando a interdisciplinaridade da aplicação.

Os materiais didáticos foram desenvolvidos com base no plano de ensino. A Figura 1 apresenta alguns exemplos de materiais criados slides, com conteúdo e tutoriais de funcionalidades, folhas de atividades para a documentação dos artefatos criados, etc. Esses materiais didáticos foram organizados no *moodle*, como sistema de gestão de aprendizagem.

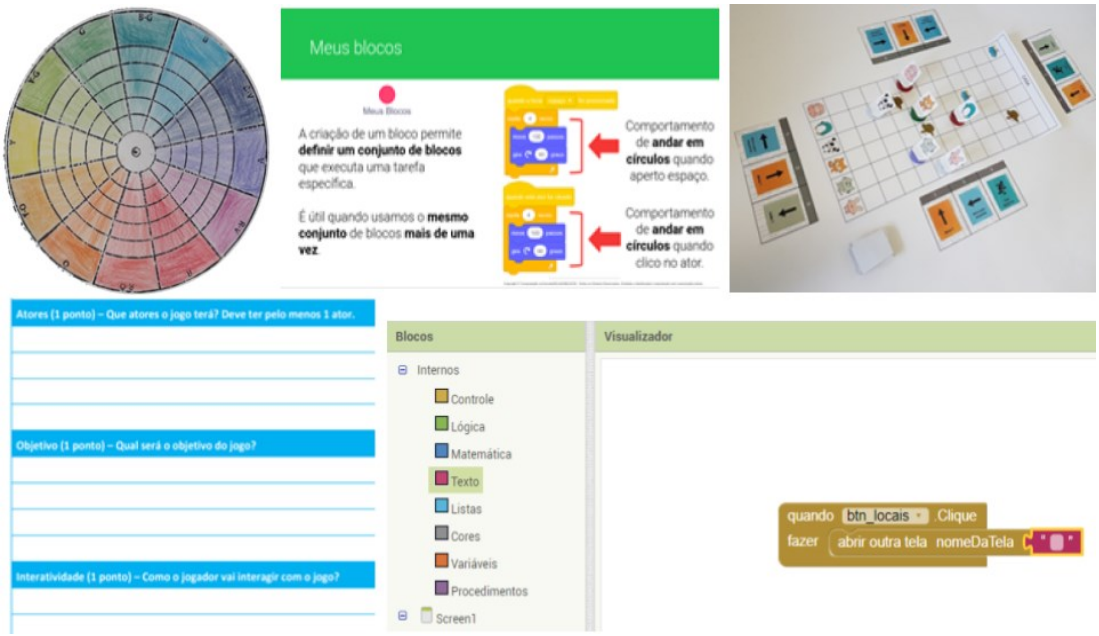
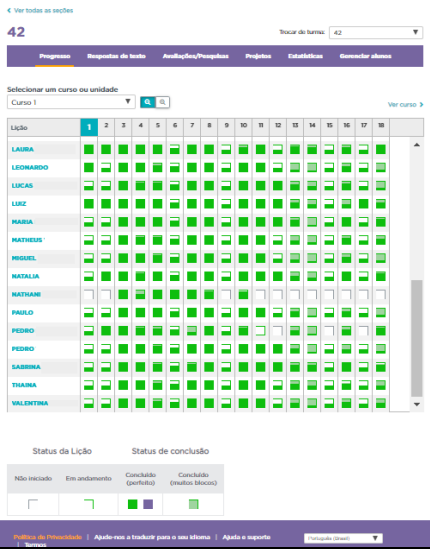
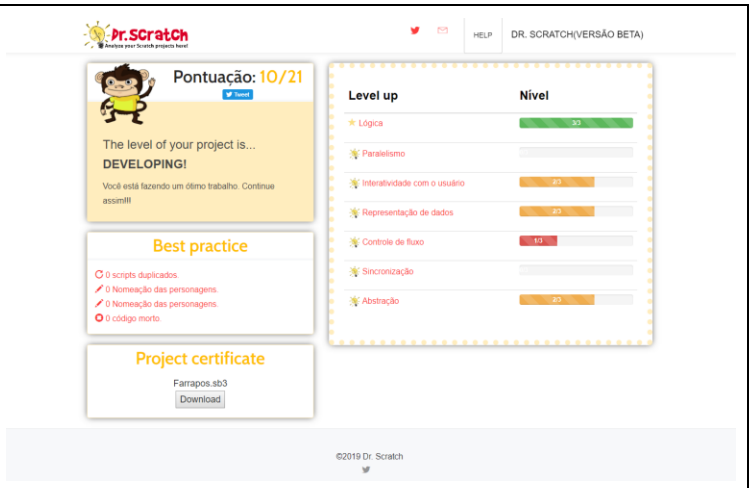


Figura 1. Exemplos de materiais instrucionais.

A avaliação da aprendizagem é baseada no desempenho nos artefatos produzidos pelos aprendizes. As avaliações por desempenho serão realizadas por meio de rubricas específicas para cada tipo de artefato e ferramentas de análise de progresso dos exercícios no Code.org [Code.org 2019], projetos Scratch [MITb 2019] e App Inventor [MITa 2019] (Tabela 2).

Tabela 2. Conteúdos ensinados em cada módulos do curso.

Módulo	Artefato	Avaliação por desempenho (rubrica ou automatizada)
1	Exercícios da Hora do Código	*Dashboard* do Code.org 
2.A	Código-fonte do jogo	Dr. Scratch

<table>
<tr><td></td><td></td><td> 
</td></tr>
<tr><td></td><td>Folhas de atividade para artefatos textuais envolvidos na produção de jogos</td><td>Extrato da rubrica
<table>
<tr><th>Tipo de jogo</th><th>Item</th><th>0 pontos</th><th>1 ponto</th><th>2 pontos</th><th>3 pontos</th><th>4 pontos</th></tr>
<tr><td rowspan="6">Ação</td><td>Atores - Que atores o jogo terá? Deve ter pelo menos 1 ator.</td><td>Não apresentou nada</td><td>Apresentou 1 ator</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Objetivo - Qual será o objetivo do jogo?</td><td>Não apresentou nada</td><td>Apresentou o objetivo de forma incompleta</td><td>Apresentou o objetivo de forma clara</td><td>-</td><td>-</td></tr>
<tr><td>Interatividade - Como o jogador vai interagir com o jogo?</td><td>Não apresentou nada</td><td>Apresentou a interatividade de forma incompleta</td><td>Apresentou a interatividade de forma clara</td><td>-</td><td>-</td></tr>
<tr><td>Regras - Quais serão as regras do jogo? Deve ter pelo menos 3 regras.</td><td>Não apresentou nada</td><td>Apresentou 1 regra</td><td>Apresentou 2 regras</td><td>Apresentou 3 ou mais regras</td><td>-</td></tr>
<tr><td>Derrota - O que faz o jogador perder o jogo?</td><td>Não apresentou nada</td><td>Apresentou a condição de derrota de forma incompleta</td><td>Apresentou a condição de derrota de forma clara</td><td>-</td><td>-</td></tr>
<tr><td colspan="6">Máximo de pontos atingível: 10 pontos</td></tr>
</table></td></tr>
<tr><td></td><td>Apresentação</td><td>Rubrica
<table>
<tr><th>Tópico</th><th>Insuficiente (0 pontos)</th><th>Regular (2 pontos)</th><th>Bom (3,5 pontos)</th><th>Excelente (5 pontos)</th></tr>
<tr><td>Bloco de comandos</td><td>O aluno não soube explicar nenhum comando de forma correta.</td><td>O aluno soube explicar corretamente, com alguma dificuldade, poucos comandos.</td><td>O aluno soube explicar corretamente, sem dificuldades, alguns comandos passo-a-passo.</td><td>O aluno soube explicar corretamente, sem dificuldades, todos os comandos passo-a-passo, isto é, de forma sistemática.</td></tr>
<tr><td>Identificação de aspectos fáceis e difíceis</td><td>O aluno não soube identificar os aspectos fáceis e difíceis de fazer durante a programação do jogo.</td><td>O aluno apresentou pouco conhecimento dos aspectos fáceis e difíceis de fazer durante a programação do jogo.</td><td>O aluno apresentou alguns argumentos sobre os aspectos fáceis e difíceis de fazer durante a programação do jogo.</td><td>O aluno apresentou uma explicação completa com argumentos convincentes dos aspectos fáceis e difíceis de fazer durante a programação do jogo.</td></tr>
<tr><td colspan="5">Máximo de pontos atingível: 10 pontos</td></tr>
</table></td></tr>
<tr><td>2.B1</td><td>-</td><td>-</td></tr>
</table>

<table>
<tr><td rowspan="2">2.B2</td><td>Código-fonte do aplicativo</td><td>CodeMaster v2.0
 
</td></tr>
<tr><td>Folhas de atividade para artefatos textuais envolvidos na produção de aplicativos</td><td>Rubrica
<table>
<tr><th>Folha de atividades</th><th>Item</th><th>0 pontos</th><th>1 ponto</th><th>2 pontos</th><th>3 pontos</th></tr>
<tr><td>Aula 2.B2.1</td><td>Descrição da necessidade/ problema</td><td>Não soube descrever</td><td>Descreveu incompleto</td><td>Descreveu com pouca clareza</td><td>Descreveu com clareza</td></tr>
<tr><td></td><td>Caracterização dos usuários</td><td>Não soube caracterizar a persona</td><td>Caracterizou incompleto</td><td>Caracterizou com pouca clareza</td><td>Caracterizou com clareza</td></tr>
<tr><td></td><td>Características dos celulares dos usuários</td><td>Não soube caracterizar a persona</td><td>Caracterizou incompleto</td><td>Caracterizou com pouca clareza</td><td>Caracterizou com clareza</td></tr>
<tr><td></td><td>Características do ambiente que o usuário utiliza o app</td><td>Não soube caracterizar a persona</td><td>Caracterizou incompleto</td><td>Caracterizou com pouca clareza</td><td>Caracterizou com clareza</td></tr>
<tr><td>Aula 2.B2.2</td><td>Descrição da solução</td><td>Não soube descrever</td><td>Descreveu incompleto</td><td>Descreveu com pouca clareza</td><td>Descreveu com clareza</td></tr>
<tr><td></td><td>Histórias de usuário</td><td>Não soube contar</td><td>Descreveu incompleto</td><td>Descreveu com pouca clareza</td><td>Descreveu com clareza</td></tr>
<tr><td></td><td>Resultados de usabilidade</td><td>Não soube descrever</td><td>Descreveu incompleto</td><td>Descreveu com pouca clareza</td><td>Descreveu com clareza</td></tr>
<tr><td>Aula 2.B2.3</td><td>Design de interface inicial</td><td>Não soube realizar</td><td>Realizou incompleto</td><td>Realizou com pouca clareza</td><td>Realizou com clareza</td></tr>
<tr><td></td><td>Resultados do teste do design da interface</td><td>Não soube descrever</td><td>Descreveu incompleto</td><td>Descreveu com pouca clareza</td><td>Descreveu com clareza</td></tr>
<tr><td></td><td>Design de interface final</td><td>Não soube realizar</td><td>Realizou incompleto</td><td>Realizou com pouca clareza</td><td>Realizou com clareza</td></tr>
<tr><td>Aula 2.B2.6</td><td>Resultados do teste funcional</td><td>Não soube descrever</td><td>Descreveu incompleto</td><td>Descreveu com pouca clareza</td><td>Descreveu com clareza</td></tr>
<tr><td></td><td>Resultados do teste de usabilidade</td><td>Não soube descrever</td><td>Descreveu incompleto</td><td>Descreveu com pouca clareza</td><td>Descreveu com clareza</td></tr>
<tr><td colspan="6">Total de pontos atingível: 48</td></tr>
</table></td></tr>
<tr><td>3</td><td>-</td><td>-</td></tr>
<tr><td>4</td><td>-</td><td>-</td></tr>
</table>

Algumas avaliações foram automatizadas utilizando ferramentas de análise de exercícios Code.org, e de análise de projetos Scratch (Dr. Scratch [Moreno-Léon & Robles 2015) e de projetos App Inventor (CodeMaster v2.0 [Alves 2019; Solecki et al. 2019]), as quais avaliam a aprendizagem de conceitos de algoritmos e programação e/ou do design de interface.

4. Aplicação do curso “Aprenda a ensinar algoritmos e programação”

O curso desenvolvido foi aplicado de forma experimental com uma turma de professores do Ensino Fundamental e Médio de uma escola particular em Florianópolis/SC de outubro de 2019 até março de 2020. O curso envolveu sete professores de diferentes disciplinas (física, geografia, sociologia filosofia, química e coordenador pedagógico). Os professores foram selecionados e convidados pela escola.

As aulas foram ministradas por pesquisadores e alunos de computação do Departamento de Informática e Estatística (INE) da UFSC, no âmbito da iniciativa Computação na Escola [Iniciativa Computação na Escola 2019]. As aulas foram inseridas na carga horária dos professores, os quais participaram de maneira voluntária no período que não lecionam.

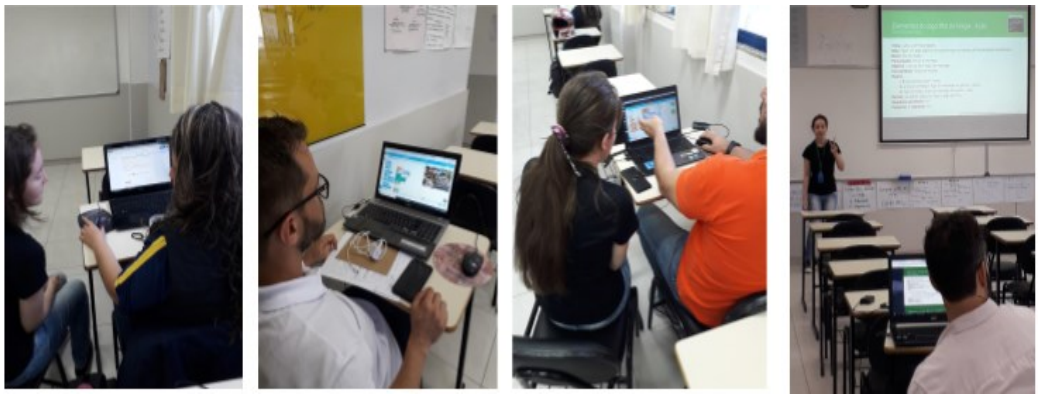


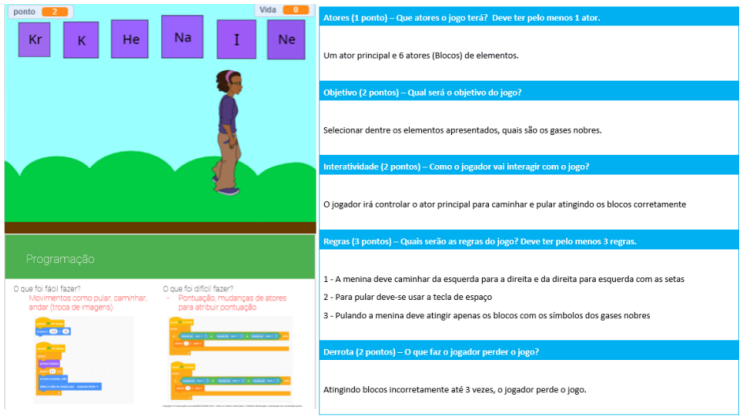
Figura 2. Cenas das aplicações da unidade instrucional.

Como a sala informatizada da escola estava ocupada, as aulas foram realizadas utilizando *notebooks* na sala de aula. Os *notebooks* possuem, em geral, tela de 15 polegadas e mouses ópticos, sistema operacional *Microsoft Windows* ou *Ubuntu*, e o acesso as páginas da *web* é realizado utilizando os navegadores *web Firefox* e *Chrome*. O acesso a rede de internet é realizado através de uma rede sem fio. A sala de aula também dispõe de projetor multimídia e quadro branco.

As aulas foram realizadas conforme o plano de ensino. Os pesquisadores e alunos forneceram assistência durante a realização dos artefatos, enquanto os professores desenvolviam os próprios programas de computador. Como resultados resolveram todas as lições do Labirinto Clássico da Hora do Código no Code.org e criaram 7 jogos completos no Scratch (Tabela 3).

Tabela 3. Artefatos produzidos pelos professores durante os dois primeiros módulos.

Módulo 1	Status de conclusão das lições no Code.org	

Módulo 2.A	Artefatos produzidos no desenvolvimento de um jogo no Scratch	

5. Avaliação do curso “Aprenda a ensinar algoritmos e programação”

5.1 Definição do estudo

O objetivo do estudo consiste em avaliar a percepção da qualidade em termos de qualidade da unidade instrucional, experiência de computação e percepção de aprendizagem do ponto de vista dos participantes (professores) no contexto do ensino de computação para professores da Educação Básica.

Para verificar se os objetivos de aprendizagem da unidade instrucional são atingidos, é definido um plano de medição seguindo a abordagem GQM [Basili, Caldiera & Rombach 1994], decompondo os objetivos em perguntas de análise e medidas baseada no modelo dETECT [Gresse von Wangenheim et al. 2017]. Adotou-se um design de pesquisa de one-shot time series post-test (X O X O X O X O X O) com os dados coletados por meio de questionários pós-módulo e avaliações de desempenho.

Tabela 4. Definição do plano de medição.

Pergunta de análise	**Medida**	**Instrumentos de coleta de dados**
PA1. O curso proporciona aprendizagem de competências em relação aos conteúdos de computação?	M1.1 Grau de aprendizagem referente a capacidade realizar atividades fechadas de programação	- Status de conclusão da lição no Code.org - Questionário pós-módulo
	M1.2 Grau de aprendizagem referente a capacidade de fazer jogos	- Dr. Scratch (pensamento computacional) - Rubrica para avaliar apresentação do jogo feito no Scratch - Rubrica para avaliar as Folhas de atividades - Questionário pós-módulo

	M1.3 Grau de aprendizagem referente a capacidade de fazer aplicativos para smartphones	- CodeMaster v2.0 (pensamento computacional e UI design) - Rubrica para avaliar as Folhas de atividades - Questionário pós-módulo
	M1.4 Grau de aprendizagem referente à capacidade de descrever um algoritmo em uma sequência de instruções	- Status de conclusão da lição no Code.org - Questionário pós-módulo
	M1.5 Grau de habilidade para ensinar os conteúdos aprendidos a outras pessoas	- Questionário pós-módulo
PA2. O curso promove uma experiência de aprendizagem agradável e divertida?	M2.1 Grau de diversão das aulas	- Questionário pós-módulo
	M2.2 Grau de percepção da passagem do tempo durante as aulas	- Questionário pós-módulo
	M2.3 Grau da interação social (compartilhar a experiência com outras pessoas)	- Questionário pós-módulo
	M2.4 Opinião subjetiva sobre os pontos que dão qualidade a aula	- Questionário pós-módulo
PA3. O curso facilita a aprendizagem (usabilidade da unidade)?	M3.1 Grau de facilidade das aulas	- Questionário pós-módulo
	M3.2 Grau de qualidade geral das aulas	- Questionário pós-módulo
	M2.3 Pontos fortes das aulas	- Questionário pós-módulo
	M2.4 Pontos fracos das aulas	- Questionário pós-módulo
PA4. O curso proporciona o interesse dos professores para ensinar conteúdos de computação em suas disciplinas?	M4.1 Grau de satisfação em realizar atividades fechadas de programação	- Status de conclusão das lições no Code.org - Questionário pós-módulo
	M4.2 Grau de satisfação em fazer jogos	- Questionário pós-módulo
	M4.3 Grau de satisfação em fazer apps para smartphones	- Questionário pós-módulo
	M4.4 Percepção da importância do ensino da computação na Educação Básica	- Questionário pós-módulo

	M4.5 Intenção de ensinar conceitos de computação nas suas disciplinas na Educação Básica	- Questionário pós-módulo

O estudo foi aprovado pela escola e pela CEPSH - Comitê de Ética em Pesquisa com Seres Humanos da UFSC com o número do parecer 3.338.481.

5.2 Coleta de dados

A avaliação foi realizada conforme previamente definida (vide Seção 3) com dados coletados via questionários pós-módulo do professor, além da análise dos artefatos produzidos pelos aprendizes. A Tabela 5 apresenta um resumo dos dados coletados durante o estudo.

Tabela 5. Quantidade de respostas para cada questionário.

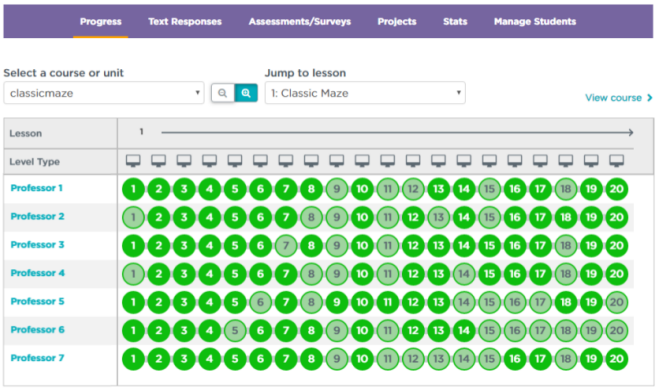
Questionário	**Quantidade**
Quantidade de questionário pós-Módulo 1 respondidos	7
Quantidade de questionário pós-Módulo 2.A respondidos	6

5.3 Análise e interpretação do estudo

Os dados coletados via questionários e avaliação de desempenho foram analisados por meio de análises qualitativas e quantitativas utilizando estatística descritiva. Os dados coletados em ambos os módulos da unidade instrucional são analisados em relação as perguntas de análise.

5.3.1 Avaliação da aprendizagem. Os resultados da análise demonstraram ótimos resultados em relação ao código e artefatos produzidos pelos professores (Tabela 6). Embora a maioria dos professores tenham informado no questionário pós-Módulo 1 que não são capazes de fazer um programa de computador, todos os professores fizeram todas as lições do Labirinto Clássico, com a maioria das lições sendo completadas de forma perfeita (verde escuro). No questionário pós-Módulo 2.A os professores afirmaram que sabem fazer um programa de computador e utilizaram os conhecimentos de computação que aprenderam para desenvolver 7 jogos educacionais.

Tabela 6. Artefatos produzidos pelos professores durante os dois primeiros módulos

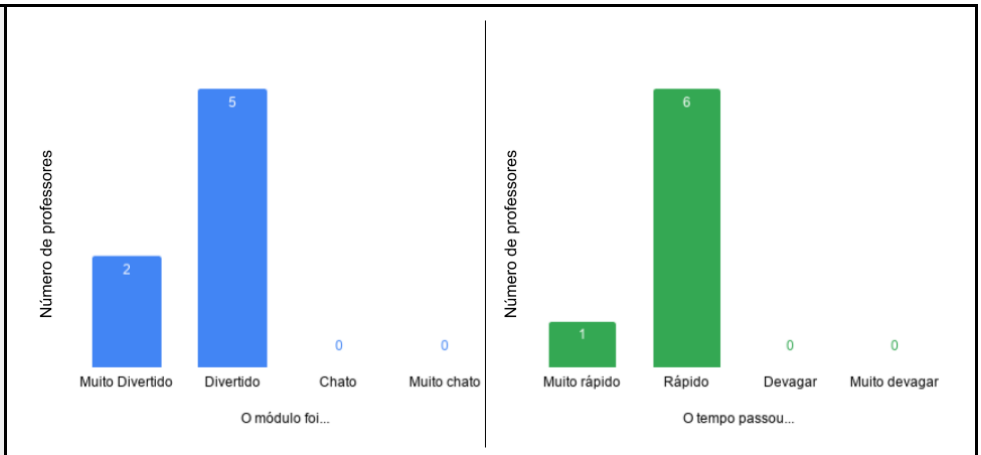
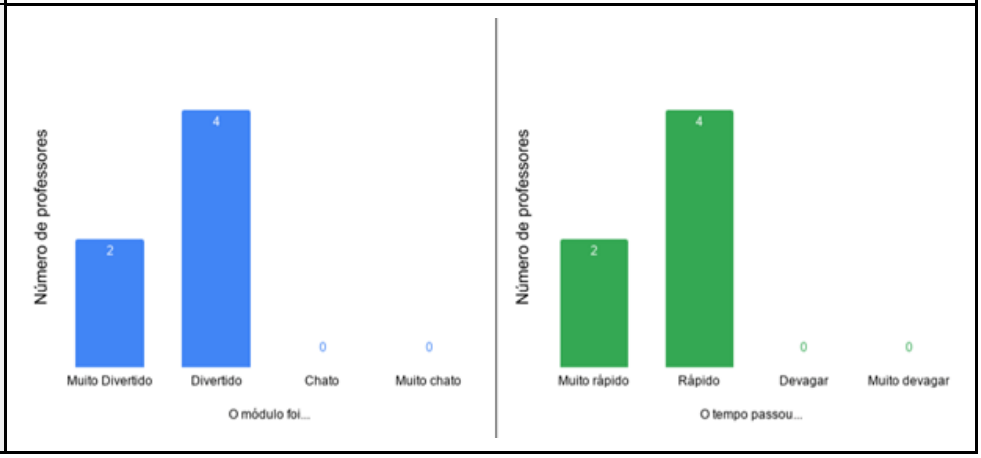
Módulo 1	Status de conclusão das lições no Code.org	
Módulo 2.A	Avaliação de um dos jogos Scratch	

O desempenho dos professores no Módulo 2.A foi avaliado com base nos jogos criados utilizando Dr. Scratch [Moreno-Léon & Robles 2019], sendo possível perceber que os conceitos apresentados nas aulas foram devidamente aplicados nos jogos desenvolvidos pelos professores. Analisando o resultado, compreende-se que os itens avaliados que os professores obtiveram pontuações maiores são os itens relacionados aos objetivos de aprendizagem, consequentemente ensinados durante a aplicação e por este motivo foram incorporados aos jogos. Com destaque para o item lógica, o qual dois alunos conseguiram obter a pontuação máxima. A avaliação desse item está relacionada com o aprendizado de conceitos definidos nos objetivos de aprendizagem, especialmente condicionais e operadores. As pontuações atribuídas para cada item avaliado demonstram que os professores conseguiram compreender os conceitos que lhes foram ensinados e por este motivo foram capazes de aplicar em seus jogos. Os artefatos textuais produzidos durante o processo de desenvolvimento dos jogos demonstram que os objetivos de aprendizagem do módulo foram atingidos, pois a maioria dos professores identificou e resolveu corretamente os problemas criando sistemas de *software* e também conseguindo desenvolver artefatos computacionais, neste caso jogos. Nas apresentações elaboradas pelos professores, estes demonstraram compreensão dos blocos de comandos utilizados na programação e souberam explicar os aspectos fáceis e difíceis de programar. as quais explicavam conceitos. Desta forma, pode-se afirmar que os professores aprenderam os conteúdos relativos aos objetivos de aprendizagem.

5.3.2 Experiência de aprendizagem. Os professores também demonstraram satisfação significativa em ambos os módulos, os quais informaram nos questionários pós-módulo que acharam o módulo divertido ou muito divertido (Tabela 7). As respostas dos questionários também mostram que os professores não sentiram o tempo passar, indicando que ficaram focados no conteúdo. Dessa forma, a experiência de aprendizagem pode ser considerada

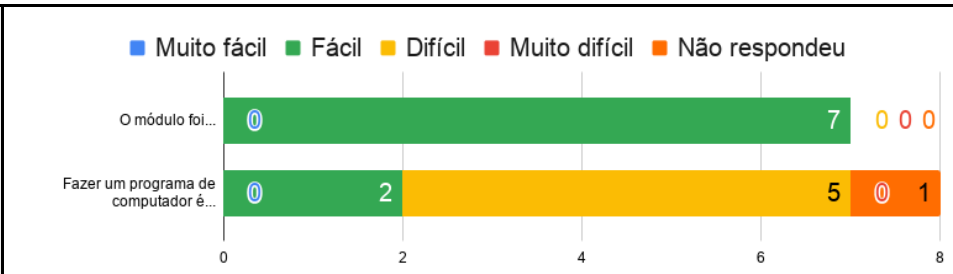
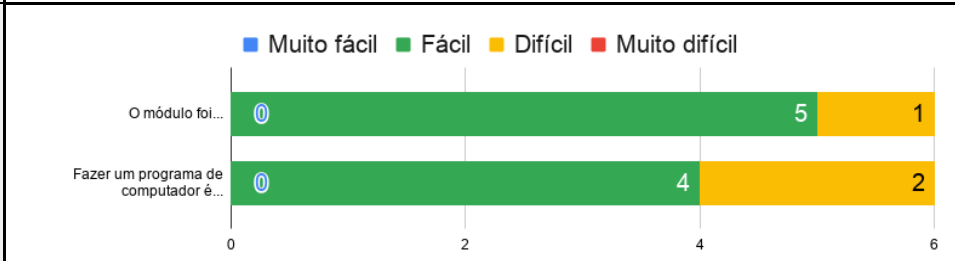
positiva. Um dos motivos para essa avaliação positiva pode estar relacionado com as atividades práticas, já que os professores afirmaram que gostaram muito desse tipo de atividade.

Tabela 7. Experiência de aprendizado nas duas aplicações

Módulo 1	
Módulo 2.A	

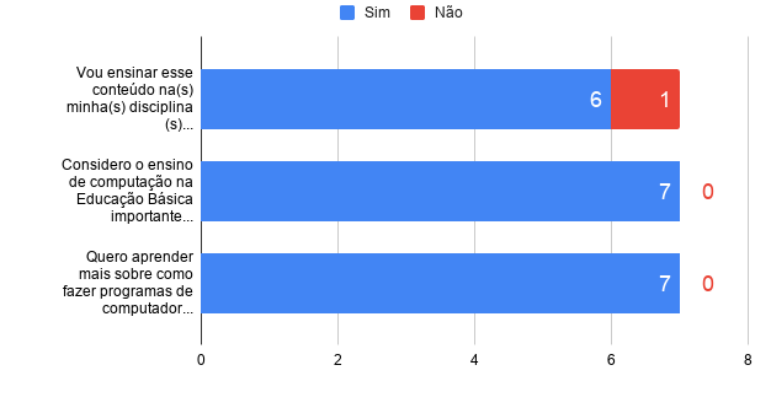
Facilidade na aprendizagem. O modo como aulas foram projetadas auxiliou para que os alunos avaliassem, em geral, que ambos os módulos foram fáceis (Tabela 8). A dificuldade informada por um professor no Módulo 2.A pode estar relacionada o pouco tempo para experimentação da ferramenta Scratch. Em ambos os módulos, alguns professores acham que fazer um programa de computador é difícil. Essas respostas podem estar associadas ao fato de ser o primeiro contato dos professores com uma linguagem de programação em blocos (no caso do Módulo 1) e ao pouco tempo para experimentação da ferramenta, em conjunto com as dificuldades dos professores realizarem um projeto de jogo (no caso do Módulo 2.A).

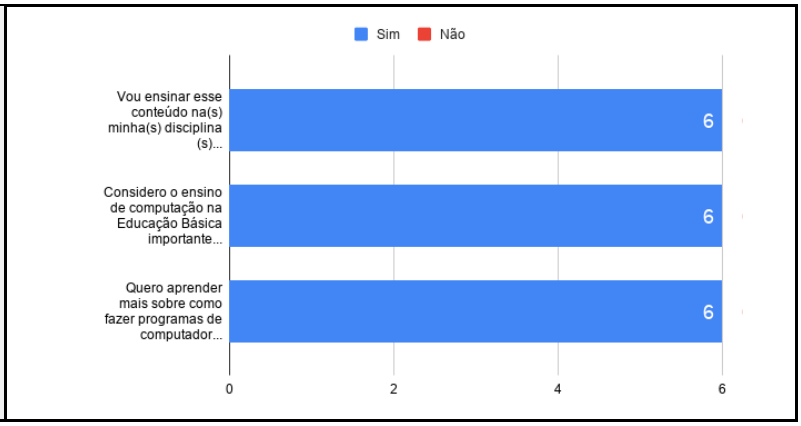
Tabela 8. Facilidade de aprendizado nas duas aplicações

Módulo 1	
Módulo 2.A	

5.3.3 Interesse dos professores referente a aprendizagem de conteúdos da computação. Os professores afirmaram que sabem explicar aos alunos como fazer um programa de computador, e ao final deste módulo também demonstraram um interesse em aprender mais sobre como fazer um programa de computador (Tabela 9). Além disso, os professores também afirmaram que é importante ensinar computação na Educação Básica, e todos querem aplicar os conteúdos aprendidos nas disciplinas que lecionam, exceto um professor que informou que não aplicará os conceitos do Módulo 1 em suas aulas. O modo como o Módulo 2.A foi organizado permitiu que alguns professores entendessem que a computação pode ser utilizada para ensinar conteúdos disciplinares. Um professor comentou que a computação pode ajudar os alunos a entenderem a lógica utilizada nas ciências exatas. Enquanto outro indicou que a organização de pensamentos e a lógica utilizada na computação podem ser utilizados para estudos em geral. Dessa forma, pode-se concluir que os professores acham interessante dos professores para ensinar conteúdos de computação em suas disciplinas.

Tabela 9. Interesse em aprender mais sobre os conteúdos nas duas aplicações

Módulo 1	

Módulo 2.A	

6. Discussão

Os resultados da análise dos dados coletados durante a aplicação demonstram de forma inicial que a unidade instrucional proposta neste trabalho permite a professores da Educação Básica aprender conceitos de programação. Observou-se também especialmente em relação ao Módulo 2.A que a UI permite que aos professores a transferência de conhecimento aos seus respectivos domínios de conhecimento criando jogos para as suas disciplinas. Na apresentação dos jogos desenvolvidos, os professores demonstraram compreensão dos blocos de comandos utilizados e souberam explicar os conceitos fáceis e difíceis. Assim, percebeu-se que os professores aprenderam os conteúdos de computação e podem ensinar o conteúdo aprendido.

A utilização das ferramentas de programação baseada em blocos, o Code.org e Scratch, também contribuíram na experiência de aprendizagem e no aprendizado de programação. Mesmo com as dificuldades encontradas pelos professores no Módulo 2.A, a utilização da ferramenta Scratch permitiu aos professores o desenvolvimento de programas de maneira mais fácil. Os resultados evidenciam que houve uma aprendizagem expressiva em programação em ambos os módulos.

Os professores avaliaram que os conteúdos do módulo foram fáceis de aprender, esse fator pode ser influência da utilização das ferramentas Code.org e Scratch e também da estratégia instrucional. Por mais que a maioria dos conceitos ensinados tenham um grau de complexidade elevado, os participantes das aplicações sentiram um nível adequado de aprendizagem. Porém, no Módulo 1 os professores informaram que acham difícil fazer um programa de computador, apesar de todos ter realizados as atividades do Labirinto Clássico no Code.org. Esta opinião pode ter ocorrido pelo fato dos professores não conhecerem conteúdos de computação e assim não se sentiram confiantes para realizar este tipo de atividade. Pois no decorrer da UI, no Módulo 2.A o número de professores que acham fácil fazer um programa de computador, aumentou consideravelmente.

Além da programação, a unidade instrucional contribuiu para ensinar outras competências relacionadas a computação. No Módulo 2.A os professores seguiram um processo de desenvolvimento jogos e adotaram práticas de Engenharia de Software. Os professores também utilizaram os conhecimentos que possuíam nos conteúdos de suas disciplinas e desenvolveram um jogo interdisciplinar. Esta abordagem incentivou a criatividade e a imaginação dos professores, para desenvolver soluções computacionais que possam ser usadas nas respectivas disciplinas. Embora alguns professores não tenham conseguido finalizar o

desenvolvimento do seu jogo até o fechamento deste trabalho, o desenvolvimento dos jogos foi útil para que os professores conheceram novas alternativas de ensino integrando o ensino de computação de forma interdisciplinar.

Os professores consideraram a participação no curso uma experiência agradável e divertida, e se sentiram envolvidos nos conteúdos dos módulos 1 e 2.A ao perceber que as aulas passaram rápidas. Isto demonstra que a estratégia das aulas, as quais predominou o uso de atividades práticas, pode ter ajudado a manter os professores imersos no processo de aprendizagem. A estratégia instrucional utilizada também pode ser notada na motivação e interesse dos professores em aprender mais sobre os conceitos de computação. Embora alguns professores tenham afirmado nos questionários pós-módulo que as atividades as quais menos gostaram foram exercícios de algoritmos propostos.

A utilização de um processo sistemático abordando competências de computação estimulou os professores a se sentirem capazes de ajudar os seus alunos a desenvolverem diversas habilidades do século XXI, como por exemplo a criatividade, a resolução de problemas e o pensamento crítico. Assim, a unidade instrucional demonstrou resultados animadores em termos de aprendizagem de computação. A UI também impactou positivamente na motivação, na experiência e na aprendizagem, como também em empoderar os professores para ensinar computação de maneira interdisciplinar nas disciplinas que lecionam.

6.1 Ameaças à validade

O estudo de caso dessa pesquisa pode apresentar vários problemas em relação à validade dos resultados. Com o intuito de mitigar estas ameaças, foi definida e documentada uma metodologia sistemática para o estudo, usando a abordagem GQM [Basili et al. 1994]. Uma das ameaças se refere a dificuldade de medir aspectos diversão e satisfação, normalmente capturados por medidas subjetivas. Para contornar essa ameaça à validade, os itens dos questionários foram sistematicamente derivados em base no instrumento de medição padronizado dETECT [Gresse von Wangenheim et al. 2019]. Outra ameaça é falta de um *benchmark*, testando o conhecimento dos alunos antes do curso, para fazer uma comparação e analisar as diferenças. Porém por se tratar de um curso de professores de outras diferentes disciplinas não foi feito um teste de conhecimento no início do curso. As características do projeto representam outra ameaça, a qual só foi possível realizar um estudo de caso, em vez de um experimento. Além disso, o fato de a unidade instrucional ter sido aplicada apenas com um número muito pequeno conjunto de participantes reduz tanto a representatividade quanto a possibilidade de generalização dos resultados. Entretanto, considerando a natureza exploratória de nossa pesquisa neste momento, consideramos aceitável o rigor científico de um estudo de caso cuidadosamente definido.

7. Conclusão

Como resultado do presente trabalho foi desenvolvida, aplicada e avaliada uma unidade instrucional para ensinar conceitos básicos de computação, bem como aspectos pedagógicos e tecnológicos aos professores *in-service* da Educação Básica, e assim permitir as utilizações esse conhecimento de forma interdisciplinar inserida em conteúdo programado do Ensino Fundamental. Os resultados obtidos nos dois primeiros módulos (Módulo 1 e Módulo 2.A) demonstram a qualidade da unidade instrucional, indicando que os professores aprenderam as

competências de computação definidas nos objetivos de aprendizagem e se divertiram aprendendo sobre os conceitos abordados.

Assim, como resultado do presente trabalho de conclusão de curso, foi criada uma unidade instrucional para ensinar computação para professores por meio de um processo sistemático de desenvolvimento de jogos com Scratch e aplicativos utilizando o App Inventor, e integrando práticas pedagógicas e tecnológicas. Essa unidade pode ser aplicada para professores da Educação Básica, com ou sem experiência em computação. Espera-se que as escolas utilizem o material didático desenvolvido para contribuir na formação continuada dos professores e, assim possam ensinar computação de forma interdisciplinar.

Referências

Alves, N. da C. (2019). CodeMaster: Um Modelo de Avaliação do Pensamento Computacional na Educação Básica através da Análise de Código de Linguagem de Programação Visual. Programa de Pós-Graduação em Ciência da Computação (PPGCC) – Universidade Federal de Santa Catarina.

Alves, N. da C.; Gresse von Wangenheim, C.; Hauck, J. C. R.; Borgatto, A. F.; Andrade, D. F. (2019). Uma Análise do Sequenciamento Pedagógico no Ensino de Computação na Educação Básica. Anais do Simpósio Brasileiro de Informática na Educação no VIII Congresso Brasileiro de Informática na Educação, Brasília/DF.

Basili, V. R.; Caldiera, G; Rombach, H. D (1994). The Goal Question Metric Approach, Encyclopedia of Software Engineering, John Wiley & Sons, New York, v. 1, 528-532.

Berges, M.; Hubwieser, P.; Magenheim, J.;Bender, E.; Margaritis-Kopecki, M.;Neugebauer, J.; Schaper, N.; Schubert, S.; Ohrndorf, L. (2013). Developing a competency model for teaching computer science in schools. In: Proc. of the 18th ACM Conference on Innovation and Technology in Computer Science Education, Cambridge, Reino Unido , 327-327.

Bower, M.; Wood, L. N.; Lai, J. W.; Howe, C.; Lister, R.; Mason, R.; Highfield, K.; Veal, J. (2017). Improving the Computational Thinking Pedagogical Capabilities of School Teachers. Australian Journal of Teacher Education, 42(3).

Branch, R. M. (2009) Instructional Design: The ADDIE Approach. New York: Springer Science & Business Media.

Code.org (2019). Disponível em: <https://code.org/>. Acesso em: 12 de junho de 2019.

Cooper, S.; Rodger, S. H.; Schep, M.; Stalvey, R. H.; Dann, W. (2015). Growing a K-12 Community of Practice. In: Proc. of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, MO, EUA, 290-295.

CSTA. (2016). K-12 Computer Science Framework. Disponível em: <http://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf>. Acesso em: 12 de junho de 2019.

De Kereki, I. F.; Manataki, A. (2016). "Code Yourself" and "A Programar": a bilingual MOOC for teaching Computer Science to teenagers. In: Proc. of the Frontiers in Education Conference, Erie, PA, EUA.

Ferreira, M. N. F., Gonçalves, B. S.; Damiani, J. C.; Gresse von Wangenheim, C. (2019). Elementos de interface em aplicativos para smartphone: uma oficina para estudantes do ensino fundamental II. Anais do Congresso Internacional de Design da Informação, Belo Horizonte.

Ferreira, M. N. F.; Gresse von Wangenheim, C.; Missfeldt Filho, R.; Da Cruz Pinheiro, F.; Hauck, J. C. R. (2019). Learning user interface design and the development of mobile applications in middle school. ACM Interactions, 26 (4).

Ferreira, M. N. F.; Pinheiro, F.; Missfeld Filho, R.; Gresse von Wangenheim, C. (2019). Ensinando Design de Interface de Usuário na Educação Básica: Um Mapeamento Sistemático do Estado da Arte e Prática. Anais do Workshop de Informática na Escola no Congresso Brasileiro de Informática da Educação, Brasília/DF.

Gal-Ezer, J.; Stephenson, C. (2010). Computer science teacher preparation is critical. ACM Inroads, 1(1), 61-66.

Goode, J. (2007). If you build teachers, will students come? the role of teachers in broadening computer science learning for urban youth. Journal of Educational Computing Research 36(1), 65–88.

Goode, J. (2008). Increasing Diversity in K-12 Computer Science: Strategies from the Field. In: Proc. of the 39th SIGCSE Technical Symposium on Computer Science Education, Portland, OR, EUA.

Gresse von Wangenheim, C.; Alves, N. d. C.; Rodrigues, P. E.; Hauck, J. C. R. (2017). Teaching Computing in a Multidisciplinary Way in Social Studies Classes in School – A Case Study. International Journal of Computer Science Education in Schools, 1(2).

Gresse von Wangenheim, C.; Petri, G.; Zibetti, A. W.; Borgatto, A. F.; Hauck, J. C. R.; Pacheco, F. S.; Missfeldt Filho, R. (2017). dETECT: A Model for the Evaluation of Instructional Units for Teaching Computing in Middle School. Informatics in Education, 16(2), 301-318.

Grover, S.; Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43.

INEP (2018). Sinopse estatística da Educação Superior 2018. Brasília: Inep, 2019. Dísponível em: <http://inep.gov.br/sinopses-estatisticas-da-educacao-superior>. Acesso em: 27 de novembro de 2019.

Iniciativa Computação na Escola (2019). Disponível em: . Acesso em: 24 de novembro de 2019.

Lamprou, A.; Repenning, A.; Escherle, N. A. (2017). The Solothurn Project — Bringing Computer Science Education to Primary Schools in Switzerland. In: Proc. of the 22th Annual Conference on Innovation and Technology in Computer Science Education. Bologna, Itália.

Liu, J.; Lin, C. H.; Potter P.; Hasson, E. P.; Barnett, Z.; Xu Y.; Singleton M. (2013). Going mobile with app inventor for android: a one-week computing workshop for K-12 teachers. In: Proc. of the 44th ACM Technical Symposium on Computer Science Education, Denver, CO, EUA, 433-438.

Lye, S. Y.; Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, 41(C), 51-61.

Mishra, P.; Koehler, M. J. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge, 108(6), 1017-1054.

Missfeldt Filho, R (2019). Desenvolvimento de uma unidade instrucional para ensinar o desenvolvimento de apps no Ensino Fundamental com o App Inventor. Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) - Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis.

MITa (2019). App Inventor - Explore. Disponível em: <http://appinventor.mit.edu/explore/>. Acesso em: 12 de junho de 2019.

MITb (2019). Scratch. Disponível em: <https://Scratch.mit.edu/>. Acesso em: 12 de junho de 2019.

Moreno-Léon, J.; Robles, G. (2015). Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. Proceedings of the Workshop in Primary and Secondary Computing Education, Londres, Reino Unido, 2015.

Naughton, J. (2012). Why All Our Kids Should Be Taught How to Code. The Guardian. Guardian News and Media.

Ni, L.; Guzdial, M. (2012). Who AM I?: understanding high school computer science teachers' professional identity. In: Proc. of the 43rd Technical Symposium on Computer Science Education, Raleigh, NC, EUA, 499-504.

Robinson, A.; Pérez-Quinõnes, M. A. (2014). Underrepresented middle school girls: on the path to computer science through paper prototyping. In: Proc. of the 45th ACM Technical Symposium on Computer science education, Atlanta, Georgia, EUA, 97-102.

SBC (2018). Diretrizes para ensino de computação na Educação Básica. Disponível em: <http://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/1177-diretrizes-para-ensino-de-computacao-na-educacao-basica>. Acesso em: 12 de maio de 2019.

Solecki, I.; Porto, J. A.; Justen, K. A.; Alves, N. d. C.; Gresse von Wangenheim, C.; Borgatto, A. F.; Hauck, J. C. R. (2019) CodeMaster UI Design – App Inventor: Uma Rubrica de Avaliação do Design de Interface de Aplicativos Android desenvolvidos com App Inventor. In: Proc. of the XVII Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais, Vitória, Brasil.

Tissenbaum, M.; Sheldon, J. e Abelson, H. (2019) From Computational Thinking to Computational Action. ACM Interactions, 62(3), p. 34-36.

Wohlin, C. et al. (2012). Experimentation in Software Engineering. Berlin: Springer-Verlag Berlin Heidelberg, p. 235.

Yin, R. K. (2009). Case Study Research: Design and Methods. SAGE, 4 ed., p. 219.