UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS REITOR JOÃO DAVID FERREIRA LIMA
PROGRAMA DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Artur Ribeiro Alfa

APLICATIVO MULTIPLATAFORMA "PROGRESSIVETUTOR" UTILIZANDO PROGRESSIVE WEB APP PARA AUXÍLIO DE ESTUDOS NA ÁREA MOBILE, DESKTOP E MULTIPLATAFORMA

Florianópolis, Santa Catarina – Brasil
2023

Artur Ribeiro Alfa

APLICATIVO MULTIPLATAFORMA "PROGRESSIVETUTOR" UTILIZANDO PROGRESSIVE WEB APP PARA AUXÍLIO DE ESTUDOS NA ÁREA MOBILE, DESKTOP E MULTIPLATAFORMA

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador(a): Raul Sidnei Wazlawick, Dr.

Florianópolis, Santa Catarina – Brasil
2023

Notas legais:

Não há garantia para qualquer parte do software documentado. Os autores tomaram cuidado na preparação desta tese, mas não fazem nenhuma garantia expressa ou implícita de qualquer tipo e não assumem qualquer responsabilidade por erros ou omissões. Não se assume qualquer responsabilidade por danos incidentais ou consequentes em conexão ou decorrentes do uso das informações ou programas aqui contidos.

Catalogação na fonte pela Biblioteca Universitária da Universidade Federal de Santa Catarina.
Arquivo compilado às 07:48h do dia 12 de julho de 2023.

Artur Ribeiro Alfa
Aplicativo multiplataforma "ProgressiveTutor" utilizando Progressive Web App para auxílio de estudos na área mobile, desktop e multiplataforma / Artur Ribeiro Alfa; Orientador(a), Raul Sidnei Wazlawick, Dr. – Florianópolis, Santa Catarina – Brasil, 27 de junho de 2023.
90 p.

Trabalho de Conclusão de Curso – Universidade Federal de Santa Catarina, INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico, Programa de Graduação em Ciências da Computação.

Inclui referências

1. Tutorial, 2. Engine, 3. PWA, I. Raul Sidnei Wazlawick, Dr. II. Programa de Graduação em Ciências da Computação III. Aplicativo multiplataforma "ProgressiveTutor" utilizando Progressive Web App para auxílio de estudos na área mobile, desktop e multiplataforma

CDU 02:141:005.7

Artur Ribeiro Alfa

APLICATIVO MULTIPLATAFORMA "PROGRESSIVETUTOR" UTILIZANDO PROGRESSIVE WEB APP PARA AUXÍLIO DE ESTUDOS NA ÁREA MOBILE, DESKTOP E MULTIPLATAFORMA

Este(a) Trabalho de Conclusão de Curso foi julgado adequado(a) para obtenção do Título de Bacharel em Ciências da Computação, e foi aprovado em sua forma final pelo Programa de Graduação em Ciências da Computação do INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico da Universidade Federal de Santa Catarina.

Florianópolis, Santa Catarina – Brasil, 27 de junho de 2023.

Lúcia Helena Martins Pacheco, Dr.
Coordenador(a) do Programa de
Graduação em Ciências da Computação

Banca Examinadora:

Raul Sidnei Wazlawick, Dr.
Orientador(a)
Universidade Federal de Santa
Catarina – UFSC

Prof. Antonio Carlos Mariani, Dr.
Instituição UFSC

Prof. Jose Eduardo de Lucca, Dr.
Instituição UFSC

Este trabalho é dedicado à todas as pessoas importantes da minha vida.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à todos os meus amigos e familiares que me ajudaram ao longo desta jornada.

“Assim como aquele pecado da juventude, este documento te perseguirá pelo resto da vida.”
Enio Valmor Kassick

“Estupidez trará mais autoconfiança do que o conhecimento e a bravura juntas. (encoding: T1, family: phv, series: m, shape: it, size: 10, baseline: 12.0pt, linespread: 1.241, linespacing: 14.89197pt)”
Adriano Ruseler

RESUMO

Na atualidade, quando um estudante deseja aprender algo, ele recorre à internet para buscar materiais de aprendizado. No entanto, esse processo frequentemente envolve horas navegando por diversos sites, assistindo a tutoriais em vídeo, lendo livros digitais e fazendo cursos online. Ao final dessa busca, é comum que o estudante tenha várias abas abertas em seu navegador e precise integrar o conhecimento obtido de todas essas fontes de forma coesa. Diante desse problema, o presente trabalho tem como objetivo desenvolver o ProgressiveTutor, uma engine de tutoriais multiplataforma, modular e expansível, utilizando Next.js. Esta engine atuará como um guia para a aprendizagem inicial em diferentes áreas de conhecimento, com foco específico no desenvolvimento multiplataforma, desktop e mobile, direcionado para alunos da área de Tecnologia da Informação. O ProgressiveTutor terá funcionalidades de um Progressive Web App (PWA), termo cunhado pela Google para descrever aplicativos web que aproveitam os recursos dos navegadores modernos, como o modo offline e notificações push. O objetivo é fornecer, em um único lugar, os conhecimentos iniciais sobre essas três áreas de conhecimento. Ao final deste trabalho, foi desenvolvido um protótipo funcional do ProgressiveTutor, composto por três módulos, um para cada área de conhecimento mencionada anteriormente. Os passos finais de cada módulo demonstram o processo de criação de um aplicativo, incluindo front-end, back-end, uma API de comunicação entre eles e funcionalidades de autenticação. Além disso, foi realizada uma avaliação da utilização do ProgressiveTutor e dos três tutoriais/módulos criados na prática.

Palavras-chaves: Tutorial. Engine. PWA.

ABSTRACT

Currently, when a student wants to learn something, they turn to the internet to search for learning materials. However, this process often involves hours of browsing through various websites, watching video tutorials, reading e-books, and taking online courses. By the end of this search, it's common for the student to have multiple tabs open in their browser and need to integrate the knowledge obtained from all these sources in a cohesive manner. Facing this problem, the present work aims to develop ProgressiveTutor, a modular, expandable, cross-platform tutorial engine using Next.js. This engine will act as a guide for initial learning in different knowledge areas, with a specific focus on cross-platform, desktop, and mobile development, targeting Information Technology students. ProgressiveTutor will have the functionalities of a Progressive Web App (PWA), a term coined by Google to describe web applications that leverage the features of modern browsers, such as offline mode and push notifications. The goal is to provide the initial knowledge about these three knowledge areas in one place. At the end of this work, a functional prototype of ProgressiveTutor was developed, composed of three modules, one for each knowledge area mentioned above. The final steps of each module demonstrate the process of creating an application, including front-end, back-end, an API for communication between them, and authentication functionalities. Additionally, an evaluation of the usage of ProgressiveTutor and the three tutorials/modules created was conducted in practice.

Keywords: Tutorial. Engine. PWA.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PWA	Progressive Web App
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
CSS	Cascading Style Sheets
REST	Representational State Transfer
tRPC	TypeScript Remote Procedure Call
JSON	JavaScript Object Notation
JWT	JSON Web Token
SSR	Server-side Rendering
CSR	Client-side Rendering
SEO	Search Engine Optimization
ORM	Object–relational Mapping
CRUD	Create, Read, Update and Delete
SQL	Structured Query Language
MVC	Model-View-Controller

SUMÁRIO

Parte I

Pesquisa

1 INTRODUÇÃO

As aplicações móveis têm crescido de forma expressiva, expandindo sua presença e criando novos mercados, proporcionando maior acessibilidade aos usuários e disponibilizando conteúdo acessível a qualquer hora. Esse crescimento é de grande magnitude, o que tem levado as empresas a repensarem sua preferência em relação à plataforma a ser priorizada, evidenciado pela popularização do termo 'Mobile First' (GOOGLE, 2023a) recentemente, bem como pela mudança no tipo de indexação praticada pela Google.

O conceito de 'Mobile First' se refere a práticas de projeto e desenvolvimento que têm como foco principal os usuários móveis. Mais especificamente, significa priorizar a experiência online dos usuários que utilizam celulares, tablets e dispositivos semelhantes antes daqueles que acessam o conteúdo por meio de desktops ou outros meios. Esta abordagem inverte o fluxo de desenvolvimento anterior, no qual se desenvolvia inicialmente para desktop e posteriormente simplificava o design para dispositivos móveis.

Atualmente, os usuários desejam ter acesso ao conteúdo digital que consomem em qualquer dispositivo com acesso à internet, seja em um telefone, tablet, computador, entre outros. Este fato faz com que as empresas tenham que disponibilizar seu conteúdo para qualquer dispositivo, independentemente da plataforma. Entretanto, este processo pode gerar altos custos caso a empresa tente desenvolver aplicativos nativos para cada tipo de plataforma, uma vez que provavelmente necessitará de uma equipe específica para cada uma delas, considerando que o desenvolvimento para Android, iOS e desktop requer diferentes conhecimentos e códigos específicos. Por esse motivo, as empresas têm buscado alternativas mais econômicas que ofereçam um produto final de boa qualidade, sendo uma delas a tecnologia PWA.

Diante deste cenário, este trabalho tem como objetivo desenvolver o ProgressiveTutor, uma engine de tutoriais multiplataforma, modular e expansível, utilizando Next.js. Inspirado pela necessidade de uma abordagem unificada de aprendizado, o ProgressiveTutor atuará como um guia abrangente para a aprendizagem inicial em diferentes áreas de conhecimento, com foco específico no desenvolvimento multiplataforma, desktop e mobile, direcionado aos alunos da área de Tecnologia da Informação.

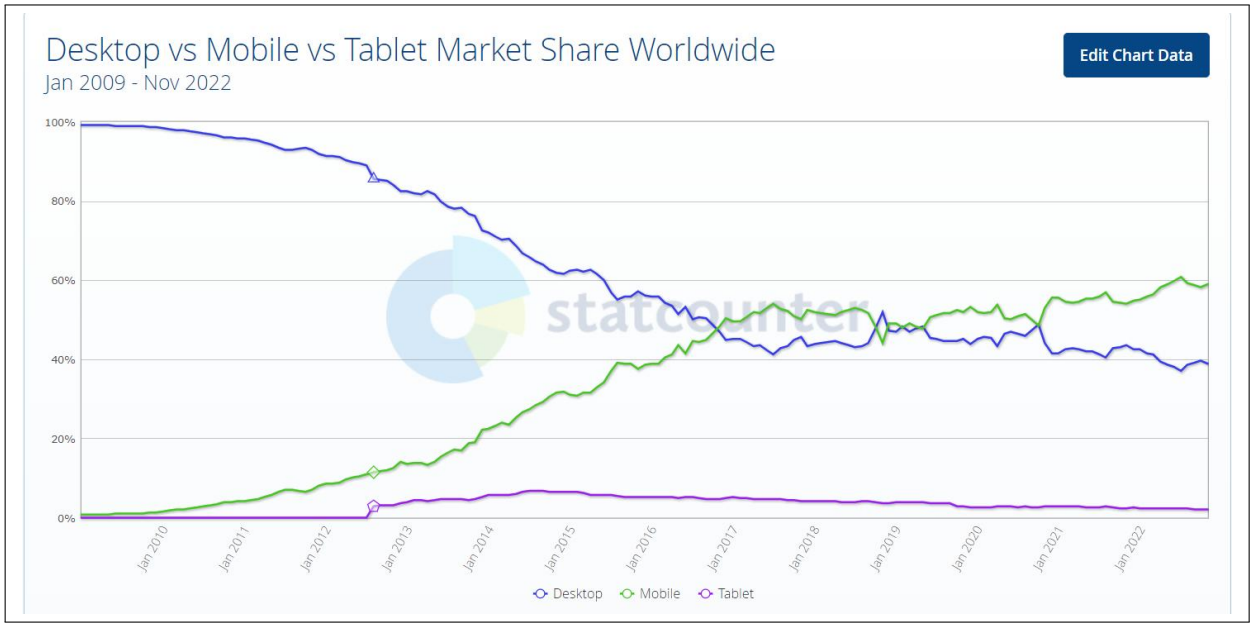
1.1 PROBLEMÁTICA

Há dez anos (STATCOUNTER, 2023), o mercado era dominado primariamente pelos desktops. As empresas desenvolviam seus produtos e serviços primeiro para desktops, sendo a versão para celular geralmente uma versão simplificada daquela criada para desktops.

Esta proporção mudou drasticamente ao longo dos anos, chegando ao estado

atual em que os dispositivos móveis correspondem a mais de 50% do mercado. Este desenvolvimento impactante deu origem à mentalidade Mobile First.

Figura 1 – Proporção de usuários desktop vs mobile vs tablet 2009 a 2022 (STATCOUNTER, 2023)



Fonte: o autor

Diante desta mudança na distribuição do mercado, as empresas, ao criar um produto ou serviço atualmente, têm, em geral, as seguintes opções ao considerar o público-alvo a ser atendido: escolher uma única plataforma (desktop, Android, iOS, web) e desenvolver de maneira nativa, abrindo mão das outras plataformas; escolher mais de uma plataforma e desenvolver de forma nativa para todas elas; ou, por último, fazer uso de uma tecnologia multiplataforma para atingir diferentes tipos de plataformas.

Cada uma destas opções possui seus prós e contras:

- 1) Escolher apenas uma plataforma e desenvolver nativamente limita o público do produto ou serviço, mas é mais econômico e simples de controlar;
- 2) Escolher mais de uma plataforma e desenvolver nativamente para todas elas alcança um público maior, mas tem um custo e complexidade maiores, além de geralmente exigir mais funcionários para desenvolver e manter o produto ou serviço;
- 3) Caso a empresa opte por fazer uso de uma tecnologia multiplataforma, ela poderá atingir um público maior em comparação com a primeira opção e economizar dinheiro, mas terá que lidar com as dificuldades e complicações que geralmente estão associadas a tecnologias multiplataforma.

1.2 MOTIVAÇÃO

Com tantas opções diferentes de tecnologias disponíveis atualmente, acaba sendo complicado para os alunos da área de Tecnologia da Informação adquirirem conhecimento sobre todas elas sem ter que gastar muitas horas em pesquisas e acessar vários sites diferentes para reunir todas as informações necessárias para desenvolver um conhecimento, pelo menos básico, sobre essas tecnologias. Por esse motivo, este projeto tem como objetivo facilitar este processo, reunindo em um só lugar todo o conhecimento necessário para aprender as etapas iniciais, possibilitando que o aluno seja capaz de desenvolver um aplicativo simples utilizando apenas o ProgressiveTutor.

1.3 OBJETIVO GERAL

O presente trabalho tem como objetivo desenvolver o ProgressiveTutor, uma engine de tutoriais multiplataforma, modular e expansível, utilizando Next.js, a fim de facilitar o processo de aprendizado nas áreas de desenvolvimento multiplataforma, desktop e mobile, direcionado para alunos da área de Tecnologia da Informação.

O ProgressiveTutor busca reunir o conteúdo disperso existente na internet em um único aplicativo, proporcionando aos estudantes um ambiente centralizado para suas necessidades de aprendizado. O objetivo é oferecer tutoriais abrangentes, que abordem desde os conceitos básicos até o desenvolvimento completo de um aplicativo, incluindo front-end, back-end, uma API de comunicação entre eles e funcionalidades de autenticação.

Cada módulo do ProgressiveTutor será estruturado de forma sequencial, iniciando com a introdução e compreensão dos conceitos fundamentais de cada área de conhecimento. Em seguida, os alunos serão guiados passo a passo na criação de um aplicativo, aplicando os conhecimentos aprendidos anteriormente. Dessa forma, os alunos poderão consolidar os conceitos e habilidades adquiridos, além de obter uma compreensão prática das tecnologias envolvidas.

O ProgressiveTutor será desenvolvido com uma abordagem modular, permitindo a expansão futura de módulos adicionais para outras áreas de conhecimento ou o aprofundamento dos módulos existentes. Isso possibilitará uma experiência personalizada de aprendizado, em que os alunos poderão escolher os módulos relevantes para suas necessidades e interesses específicos.

Ao final do desenvolvimento do ProgressiveTutor, será criado um protótipo funcional, composto por três módulos representativos das áreas de desenvolvimento multiplataforma, desktop e mobile. Cada módulo apresentará um tutorial completo, abrangendo desde os conceitos básicos até a implementação de um aplicativo com todas as funcionalidades mencionadas. Além disso, será realizada uma avaliação da utilização do ProgressiveTutor e dos três módulos criados na prática, visando analisar

a eficácia do aplicativo como guia de aprendizado.

1.4 OBJETIVOS ESPECÍFICOS

1. Realizar uma análise das funcionalidades oferecidas pela tecnologia PWA, incluindo seus pontos positivos e negativos em comparação com outras tecnologias existentes, e apresentar os resultados em uma tabela comparativa.

2. Desenvolver tutoriais abrangentes em cada módulo do ProgressiveTutor, que abordem desde os conceitos fundamentais até o desenvolvimento completo de um aplicativo do tipo mobile, desktop e multiplataforma PWA. Os tutoriais incluirão a criação do front-end, back-end, uma API de comunicação entre eles e funcionalidades de autenticação, fornecendo aos alunos um conhecimento abrangente e prático nessas áreas.

3. Estruturar cada módulo do ProgressiveTutor de forma sequencial, guiando os alunos passo a passo na criação de um aplicativo, permitindo que consolidem os conceitos e adquiridos, além de obterem uma compreensão prática das tecnologias envolvidas.

4. Desenvolver o ProgressiveTutor com uma abordagem modular, permitindo a expansão futura com a adição de módulos adicionais para outras áreas de conhecimento ou o aprofundamento dos módulos existentes, possibilitando uma experiência personalizada de aprendizado para os alunos.

5. Criar um protótipo funcional do ProgressiveTutor, composto por três módulos representativos das áreas de desenvolvimento multiplataforma, desktop e mobile, que apresentem tutoriais completos, abrangendo desde os conceitos básicos até a implementação de um aplicativo com todas as funcionalidades mencionadas.

6. Realizar uma avaliação prática da utilização do ProgressiveTutor e dos três módulos criados, visando analisar a eficácia do aplicativo como guia de aprendizado nas áreas de desenvolvimento multiplataforma, desktop e mobile.

7. Entregar o código-fonte do ProgressiveTutor e disponibilizá-lo para uso livre, possibilitando que outros estudantes e desenvolvedores possam utilizar e contribuir para o projeto.

1.5 JUSTIFICATIVA

Atualmente, as opções disponíveis para adquirir conhecimento sofrem de limitações que dificultam sua utilização. Os vídeos podem ser incompletos, pesados, de difícil busca e exigem uma conexão constante à internet para visualizá-los. Os livros

online geralmente são extensos, podem ser difíceis de encontrar ou estar disponíveis apenas mediante pagamento. Além disso, os cursos online podem não abordar todas as tecnologias discutidas neste trabalho ou estar disponíveis somente mediante pagamento.

1.6 MÉTODO DE PESQUISA

Nesta seção, são descritos o método de pesquisa utilizado, categorizando-o em relação à sua natureza, abordagem, objetivo e procedimento técnico.

1.6.1 Natureza: Aplicada

Este trabalho tem como objetivo criar uma ferramenta gratuita que facilite o aprendizado sobre o desenvolvimento mobile, desktop e multiplataforma para estudantes da área de Tecnologia da Informação, utilizando as tecnologias mobile nativa, desktop e multiplataforma PWA.

1.6.2 Abordagem: Qualitativa

A aplicação desenvolvida neste projeto busca centralizar as informações já existentes na internet, reduzindo a complexidade da busca por informações básicas e iniciais. Ao final, ensina como criar um projeto inicial simples em cada uma das tecnologias abordadas, com front-end, back-end, uma API para comunicação entre eles e autenticação, a fim de consolidar os conhecimentos adquiridos.

1.6.3 Objetivo: Exploratório

Com o objetivo de compreender completamente o problema, este trabalho abordará o estado da arte da tecnologia multiplataforma PWA, comparando-a com outras soluções utilizadas atualmente. Dessa forma, almeja-se fornecer uma visão abrangente do estado atual de maturidade da tecnologia e sua posição no mercado.

1.6.4 Procedimento técnico: Bibliográfico e estudo de caso

Atualmente, já existem recursos na internet com foco no ensino das tecnologias abordadas neste trabalho, porém esses conhecimentos estão dispersos ou disponíveis apenas mediante pagamento. O objetivo é reunir esses conhecimentos neste trabalho e disponibilizá-los de forma multiplataforma, gratuita, offline e com as funcionalidades de um PWA, utilizando informações de trabalhos existentes com essa tecnologia.

1.7 ESTRUTURA DO TEXTO

Este documento está estruturado em sete capítulos.

O Capítulo 1, Introdução, apresentou uma visão geral do trabalho, incluindo a problematização, motivação, justificativa, método de pesquisa e objetivos do projeto.

O Capítulo 2, Fundamentação tecnológica, explana alguns conceitos e tecnologias abordados neste trabalho, como Aplicações Web, Progressive Web Apps (PWA) e Node.js.

O Capítulo 3, Trabalhos relacionados, apresenta uma análise de outros trabalhos que abordam um tema similar ao do presente documento.

O Capítulo 4, Especificações e métodos, descreve a organização do aplicativo desenvolvido neste trabalho, bem como as especificações técnicas necessárias para seu desenvolvimento.

O Capítulo 5, Desenvolvimento, detalha o processo de desenvolvimento de todas as aplicações produzidas neste trabalho.

O Capítulo 6, Avaliação do Protótipo, apresenta a definição dos testes realizados para avaliar os protótipos criados, bem como os resultados obtidos.

O Capítulo 7, Conclusões e trabalhos futuros, traz as conclusões do trabalho realizado e sugere possíveis direções para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA E TECNOLÓGICA

Neste capítulo, são apresentadas as tecnologias e conceitos utilizados neste trabalho. A escolha dessas tecnologias foi baseada na familiaridade do autor com elas, e essa escolha foi feita com o objetivo de demonstrar a parte prática deste trabalho. No entanto, é importante ressaltar que, devido à natureza modular e expansível do ProgressiveTutor, é possível utilizar outras ferramentas para orientar o aprendizado nas áreas de conhecimento abordadas.

2.1 APLICAÇÕES WEB

Aplicações Web são sistemas ou programas executados diretamente no navegador (AWS-AMAZON, 2023). Para que uma aplicação web funcione, ela depende de um servidor web, das solicitações realizadas pelos usuários, do uso de protocolos e métodos (normalmente o HTTP) e da resposta do protocolo (BETRYBE, 2023).

A principal diferença entre sites e aplicações web é que, enquanto os sites são estáticos, constituindo apenas um conjunto de páginas que, normalmente, têm apenas a função de informar o usuário sobre algo, com pouca interação, as aplicações web são mais interativas, permitindo que o usuário tenha uma maior interação com os elementos presentes nelas.

Algumas das vantagens das aplicações web são:

- Acesso rápido/simples: As aplicações web podem ser acessadas diretamente, sem a necessidade de download através de lojas de aplicativos ou similares. Basta acessar o endereço e, consequentemente, o conteúdo da aplicação web.
- Atualizações constantes: Como não é necessário passar por uma loja de aplicativos ou similar, os desenvolvedores podem facilmente atualizar a aplicação web, e os usuários podem acessar a versão atualizada sem precisar fazer nenhum download ou atualização em seus dispositivos.
- Compatibilidade: Por serem acessadas através de um navegador, desde que um determinado navegador e versão possam acessar uma aplicação web, todos os dispositivos que suportam esse navegador e versão também poderão acessar a aplicação web.

As aplicações web são geralmente divididas em duas partes principais: o front-end e o back-end. O front-end é responsável pela parte visual de um site. Por meio de códigos, uma interface é construída, preferencialmente amigável. As tecnologias mais utilizadas para a construção de um site são HTML e CSS, além de JavaScript para implementação de funcionalidades adicionais (BETRYBE, 2023). Já o back-end, como o próprio nome sugere, é tudo o que está por trás de uma aplicação. É nessa parte que

ocorre a construção da base da aplicação, utilizando linguagens de programação como Java, Ruby, C#, Python, PHP, entre outras. Além disso, é por meio do back-end que ocorre a conexão com bancos de dados e o carregamento de informações (BETRYBE, 2023).

2.2 PWA

Um PWA(WEB-DEV, 2023) é uma evolução das aplicações web normais, onde ele tenta trazer as vantagens de aplicativos web e de um aplicativo nativo. Ao acessá-lo por um desktop, ele age como se fosse um simples aplicativo web, mas ao ser utilizado em um celular/tablet é possível utilizar funções que antes só estavam disponíveis para aplicativos nativos, como a habilidade de mandar notificações push. Abaixo está uma tabela comparativa entre algumas das diferenças entre aplicativos nativos, PWA e aplicativos web.

	Aplicativo Nativo	**PWA**	**Aplicativo Web app padrão**
Instalação	Precisa ir na App Store/Play Store e clicar para baixar	No Android, só precisa clicar no botão para adicionar à tela principal	Não precisa instalar
Atualização	Precisa submeter a atualização à loja e aguardar sua disponibilização para que os usuários possam baixá-la	Atualizações são instantâneas	Atualizações são instantâneas
Tamanho	Geralmente tem um tamanho grande, o que pode levar um tempo considerável para baixar, dependendo da qualidade da conexão com a internet do usuário	Possui um tamanho pequeno	Possui um tamanho pequeno
Acesso offline	Disponível	É necessário utilizar o aplicativo pelo menos uma vez online para poder acessar posteriormente o conteúdo armazenado em cache	Disponível (IBM, 2023)

	Aplicativo Nativo	**PWA**	**Aplicativo Web app padrão**
Experiência do usuário	A experiência de uso está diretamente relacionada ao design do aplicativo, quanto melhor for o design, melhor será a experiência do usuário	Pode ser um pouco confuso quando utilizado no navegador, devido à presença de dois menus (o do navegador e o do aplicativo)	Igual à experiência do PWA
Notificações push	Disponível	Disponível apenas no Android	Disponível, porém requer serviços terceirizados (APRIORIT, 2023)
Facilidade de encontrar o aplicativo	Depende fortemente da loja de aplicativos, sendo necessário utilizar todas as funções disponíveis de cada loja, como configurar boas palavras-chave, pagar por promoção do aplicativo e ser listado entre os melhores apps de uma determinada categoria	Boa chance de encontrar o aplicativo, mas é necessário otimizá-lo para motores de busca	Igual ao PWA (GOOGLE, 2023b)

Tabela 1 – Comparativo: Aplicativo Nativo vs PWA vs Aplicativo Web Padrão.

Um detalhe importante a ser analisado é que, como o PWA e o web app não dependem da loja de aplicativos, os desenvolvedores não precisam pagar taxas mensais para publicar seus aplicativos, nem passar por processos de inspeção para verificar se eles estão em conformidade com as diretrizes da loja. Isso é benéfico para os desenvolvedores, mas pode ser prejudicial para os usuários, pois não há uma garantia mínima de qualidade que o aplicativo precise cumprir, uma vez que não passa por

inspeções.

Além disso, é importante considerar que o processo de disponibilização de atualizações nas lojas de aplicativos pode deixar os usuários expostos caso um aplicativo apresente uma falha que precise ser corrigida imediatamente. Mesmo que os desenvolvedores iniciem rapidamente o processo de publicação da atualização, ela pode levar tanto um tempo imediato como vários dias para estar disponível (GOOGLE, 2023c). Isso pode ser vantajoso para os desenvolvedores em termos de agilidade na disponibilização das atualizações, mas, do ponto de vista do usuário, diminui a garantia de que a atualização se comportará corretamente, já que não passará pela inspeção das lojas de aplicativos.

Por fim, é válido destacar a questão financeira: os aplicativos nativos, ao utilizarem a loja de aplicativos para pagamentos de compras internas, são obrigados a pagar uma porcentagem às lojas (CNBC, 2023), enquanto os aplicativos PWA e web apps não possuem essa desvantagem. Mais uma vez, isso é vantajoso para os desenvolvedores, porém expõe os usuários, uma vez que os PWA e web apps não oferecem garantia de que os dados fornecidos, como números de cartão de crédito inseridos para compras, não serão utilizados de forma maliciosa.

2.3 API

Uma API, que significa "Interface de Programação de Aplicações", é um conjunto de regras e protocolos que permite que diferentes softwares se comuniquem entre si (AMAZON, 2023). Ela atua como uma ponte que permite que um aplicativo ou sistema solicite e receba informações de outro, de forma segura e organizada.

Uma API define os métodos e formatos de dados que podem ser usados para acessar os recursos de um serviço ou aplicativo específico. Ela fornece uma maneira padronizada para que os desenvolvedores interajam com uma aplicação, sem precisar conhecer todos os detalhes internos da implementação.

As APIs são amplamente utilizadas na construção de aplicativos e serviços digitais, permitindo a integração entre diferentes plataformas, como redes sociais, serviços de pagamento, sistemas de geolocalização, entre outros. Elas tornam possível a troca de informações e a execução de tarefas específicas, como recuperar dados, enviar dados, autenticar usuários e realizar ações remotamente.

Existem quatro tipos de APIs que funcionam de maneiras diferentes: as APIs SOAP usam XML para troca de mensagens; as APIs RPC permitem que o cliente execute funções no servidor e receba os resultados; as APIs Websocket facilitam a comunicação bidirecional entre cliente e servidor usando JSON; e as APIs REST são as mais populares e flexíveis, permitindo que o cliente envie solicitações ao servidor e receba os dados de retorno. Cada tipo de API tem suas características e finalidades específicas.

2.4 REST

REST (Representational State Transfer)(REDHAT, 2023) é um estilo arquitetural para projetar sistemas em rede, especialmente serviços web. Ele promove simplicidade, escalabilidade e modificabilidade, separando as responsabilidades do cliente e do servidor, utilizando um modelo de comunicação sem estado. Isso significa que cada solicitação feita pelo cliente ao servidor contém todas as informações necessárias para entender e processar a solicitação, sem a necessidade de manter um estado de comunicação contínuo. Além disso, o REST utiliza uma interface padronizada baseada em HTTP, permitindo que os recursos sejam identificados e manipulados por meio de URLs e métodos HTTP, como GET, POST, PUT e DELETE.

Sistemas RESTful aproveitam o cache e a arquitetura em camadas para melhorar o desempenho e a escalabilidade. O cache permite armazenar temporariamente as respostas das solicitações mais frequentes, reduzindo a carga no servidor e melhorando a velocidade de resposta. A arquitetura em camadas permite que o sistema seja dividido em componentes independentes, cada um responsável por uma funcionalidade específica. Isso promove a escalabilidade, pois novas camadas podem ser adicionadas conforme necessário, sem afetar as camadas existentes. As APIs RESTful se tornaram populares por sua simplicidade, escalabilidade e compatibilidade com a infraestrutura web existente. Elas permitem o desenvolvimento de sistemas robustos e interoperáveis em ambientes distribuídos modernos.

2.5 TRPC

tRPC(TRPC, 2023) é um framework em TypeScript projetado para facilitar o desenvolvimento de APIs eficientes e seguras em termos de tipos. Ele oferece uma abordagem amigável para os desenvolvedores na construção de APIs, aproveitando as capacidades de tipagem estática do TypeScript. Com o tRPC, os desenvolvedores podem garantir a integridade das interações com a API e reduzir erros comuns encontrados durante os processos de integração. O framework suporta vários métodos HTTP e incorpora funcionalidades essenciais, como autenticação, validação e tratamento de erros.

Um recurso notável do tRPC é a geração automática de código do cliente, que elimina a necessidade de ajustes manuais e garante que o código do cliente permaneça sincronizado com a API no lado do servidor. O código do cliente gerado fornece métodos seguros em termos de tipos para fazer requisições à API, reduzindo a probabilidade de erros em tempo de execução. Além disso, o tRPC oferece suporte a cache no lado do servidor, suporte a middlewares e integração com frameworks populares como Next.js e Express.

2.6 JWT

JSON Web Token (JWT) é um padrão aberto para criar tokens de acesso seguros e compactos. Ele é utilizado para autenticar e autorizar o acesso a recursos em sistemas web e APIs. Um JWT é uma sequência de caracteres codificada em Base64 que consiste em três partes: um cabeçalho, um payload e uma assinatura. O cabeçalho especifica o tipo de token e o algoritmo de criptografia usado. O payload contém informações adicionais, como identificação do usuário ou permissões (OKTA, 2023). A assinatura é gerada usando uma chave secreta para verificar a integridade do token. O JWT é geralmente usado para autenticação, fornecendo um meio seguro de transmitir informações sobre o usuário entre o cliente e o servidor. Ele pode ser armazenado no lado do cliente, como um cookie ou um cabeçalho de autorização, e validado pelo servidor para verificar a autenticidade e autorização do usuário. O JWT é uma solução popular para implementar autenticação stateless em aplicações web.

2.7 NEXTAUTH.JS

O NextAuth.js(NEXT-AUTH, 2023) é uma biblioteca popular para autenticação em aplicações Next.js. Ele simplifica a implementação de recursos de autenticação, fornecendo suporte a vários provedores de autenticação, gerenciamento de sessões e recursos de autorização. Com métodos de autenticação diversificados e uma API simples, o NextAuth.js facilita a integração de fluxos de autenticação personalizados em projetos Next.js, incluindo autenticação por email/senha, redes sociais e provedores externos.

Uma das principais vantagens do NextAuth.js é sua extensibilidade, permitindo que os desenvolvedores personalizem o processo de autenticação e adicionem lógica personalizada. Além disso, ele lida com tarefas relacionadas à autenticação, como registro de usuários, login, logout e recuperação de senha, e oferece recursos avançados, como gerenciamento de sessões e controle de acesso baseado em papéis. Com sua simplicidade, flexibilidade e recursos abrangentes, o NextAuth.js é amplamente utilizado como uma solução confiável para autenticação em aplicações Next.js.

2.8 TYPESCRIPT

TypeScript(TYPESCRIPT, 2023) é uma linguagem de programação de código aberto desenvolvida pela Microsoft. Ela é uma extensão do JavaScript que adiciona recursos de tipagem estática ao código. Com o TypeScript, os desenvolvedores podem definir tipos para as variáveis, parâmetros de função, retornos de função e outros elementos do código. Isso permite uma detecção mais eficiente de erros durante a fase de desenvolvimento, ajudando a prevenir bugs e melhorando a qualidade do código.

Além da tipagem estática, o TypeScript também traz recursos adicionais, como suporte a classes, interfaces, herança, módulos e uma série de outras funcionalidades avançadas. Ele é amplamente utilizado no desenvolvimento de aplicativos web e é particularmente popular em projetos com grande escala e equipes de desenvolvedores.

Uma das principais vantagens do TypeScript é que ele é compilado para JavaScript, o que significa que o código TypeScript pode ser executado em qualquer navegador ou ambiente que suporte JavaScript. Isso permite que os desenvolvedores aproveitem os recursos avançados do TypeScript durante o desenvolvimento e, em seguida, gerem um código JavaScript compatível para implantação.

2.9 NEXT.JS

Next.js(NEXTJS, 2023) é um framework de desenvolvimento web de código aberto que permite a criação de aplicativos web rápidos, eficientes e escaláveis. Ele é baseado em React, uma biblioteca JavaScript popular para criação de interfaces de usuário. O Next.js oferece recursos avançados, como renderização do lado do servidor (SSR), renderização do lado do cliente (CSR), geração de páginas estáticas e dinâmicas, roteamento avançado e suporte a APIs.

O Next.js simplifica o processo de desenvolvimento web, fornecendo uma estrutura organizada e pronta para uso. Ele oferece recursos como pré-renderização de páginas, o que melhora o desempenho e a experiência do usuário, além de permitir a otimização para mecanismos de busca (SEO). O Next.js também é altamente flexível e extensível, permitindo a integração com outras bibliotecas e ferramentas. Com sua abordagem baseada em componentes reutilizáveis, o Next.js permite o desenvolvimento rápido e eficiente de aplicativos web modernos.

2.10 VERCEL

Vercel(VERCEL, 2023) é uma plataforma de hospedagem e implantação de aplicativos que permite aos desenvolvedores implantar facilmente seus projetos na web. Ela oferece uma maneira simples e eficiente de criar, implantar e dimensionar aplicativos estáticos e dinâmicos. A Vercel é conhecida por sua integração com ferramentas populares de desenvolvimento, como Next.js e React, e por sua capacidade de fornecer implantações rápidas e confiáveis. Com a Vercel, os desenvolvedores podem implantar seus aplicativos em uma infraestrutura de alto desempenho e obter recursos como otimização de imagem, cache de conteúdo e balanceamento de carga automático. Além disso, a Vercel oferece recursos de colaboração e integração contínua, facilitando o trabalho em equipe e a implantação contínua de projetos web.

2.11 PRISMA

O Prisma(PRISMA, 2023) é uma ferramenta de desenvolvimento de banco de dados que utiliza mapeamento objeto-relacional (ORM) para simplificar a interação entre aplicativos e bancos de dados. Com uma sintaxe orientada a objetos e uma camada de abstração, os desenvolvedores podem manipular dados de forma mais fácil e eficiente, sem precisar lidar diretamente com SQL. O Prisma suporta vários bancos de dados e oferece recursos avançados, como migrações automatizadas e otimizações de desempenho.

Com o Prisma, os desenvolvedores podem definir o modelo de dados de maneira declarativa e gerar automaticamente o esquema do banco de dados correspondente. Isso permite executar operações de criação, leitura, atualização e exclusão (CRUD) e consultas complexas usando uma API poderosa. Além disso, o Prisma integra-se a frameworks populares, como o GraphQL, facilitando a criação de APIs flexíveis e eficientes.

2.12 RAILWAY

Railway(RAILWAY, 2023) é uma plataforma de hospedagem e implantação de aplicativos que oferece uma maneira simplificada e eficiente de disponibilizar projetos web. Ela permite que os desenvolvedores implantem seus aplicativos de forma rápida e fácil, com recursos de implantação contínua e integração com ferramentas populares de desenvolvimento. A Railway é conhecida por sua interface amigável e intuitiva, que simplifica o processo de configuração e implantação de aplicativos web. Ela oferece suporte para várias linguagens de programação e frameworks, permitindo que os desenvolvedores escolham a tecnologia mais adequada para seus projetos. Com a Railway, é possível hospedar aplicativos web de forma escalável, confiável e segura, aproveitando recursos como balanceamento de carga automático e gerenciamento de infraestrutura.

2.13 STRAPI

Strapi(STRAPI, 2023a) é uma plataforma de gerenciamento de conteúdo (CMS) de código aberto que permite aos desenvolvedores criar e gerenciar facilmente conteúdo em seus aplicativos e sites. Com o Strapi, é possível criar APIs personalizadas para fornecer conteúdo estruturado aos front-ends. Ele oferece uma interface de administração intuitiva, onde os usuários podem criar e editar conteúdo de forma fácil e visual. O Strapi é altamente flexível e adaptável, permitindo que os desenvolvedores personalizem e estendam suas funcionalidades de acordo com as necessidades do projeto. Além disso, ele oferece recursos avançados, como autenticação de usuários, controle de acesso, tradução de conteúdo e muito mais. O Strapi é uma solução pode-

rosa para criar e gerenciar conteúdo em aplicativos e sites, proporcionando um fluxo de trabalho eficiente e uma experiência de usuário aprimorada.

2.14 SUPABASE

Supabase é uma plataforma de desenvolvimento de aplicativos que combina um banco de dados com linguagem de consulta estruturada (SQL) PostgreSQL com uma camada de API e autenticação integrada. Ele oferece recursos como armazenamento de dados em tempo real, autenticação de usuários e gerenciamento de bancos de dados, proporcionando uma alternativa de código aberto ao Firebase. Com o Supabase, os desenvolvedores podem criar aplicativos escaláveis e seguros, aproveitando o poder do PostgreSQL e da API RESTful. A plataforma é altamente personalizável e flexível, permitindo estender suas funcionalidades por meio de funções serverless e webhooks.

O Supabase simplifica o processo de desenvolvimento, fornecendo uma interface amigável, documentação detalhada e uma variedade de bibliotecas e SDKs para diferentes plataformas. Ele é adequado para criar aplicativos web e móveis, oferecendo uma experiência de desenvolvimento eficiente. Com sua abordagem completa e integrada, o Supabase se destaca como uma opção popular entre os desenvolvedores, permitindo criar aplicativos modernos e robustos de maneira eficaz e produtiva.

2.15 CREATE T3-APP

Create T3-App(T3, 2023) é uma ferramenta de linha de comando ou utilitário que permite aos desenvolvedores gerar rapidamente uma nova estrutura de projeto ou aplicativo com base no framework T3. T3 é um framework de desenvolvimento web que fornece um conjunto de ferramentas pré-configuradas e melhores práticas para a construção de aplicativos escaláveis e de fácil manutenção.

Com o Create T3-App, os desenvolvedores podem inicializar um novo projeto T3 com a estrutura de pastas necessária, arquivos de configuração e dependências iniciais. Ele automatiza o processo de configuração e fornece um ponto de partida padronizado para projetos baseados no T3, economizando tempo e esforço para os desenvolvedores.

A ferramenta oferece várias opções e configurações, permitindo aos desenvolvedores personalizar o projeto gerado de acordo com seus requisitos específicos. O Create T3-App ajuda a agilizar o processo de configuração inicial, permitindo que os desenvolvedores se concentrem mais na construção dos recursos e funcionalidades do aplicativo, em vez de gastar tempo com código repetitivo.

3 REVISÃO BIBLIOGRÁFICA

Com o objetivo de identificar trabalhos relacionados a esta pesquisa, foi realizada uma revisão bibliográfica sistemática da literatura, seguindo o método descrito por (KITCHENHAM, 2004) como base.

3.1 PROTOCOLO DE REVISÃO

Esta revisão buscou analisar trabalhos que, como este trabalho, tem como objetivo criar um tutorial web e/ou mobile para facilitar o aprendizado de uma ferramenta/linguagem de programação para um determinado publico.

A coleta dos artigos foi realizada por meio de um levantamento bibliográfico na base de dados IEEE Xplore no período de agosto de 2022 a dezembro de 2022. Os descritores utilizados são os demonstrados na tabela abaixo.

Termo de busca	Resultados
"Document Title":tutorial	3.761
"Document Title":tutorial AND teach	148
"Document Title":tutorial AND teach AND (web OR mobile)	30

Tabela 2 – Termos utilizados na revisão sistemática

3.2 CRITÉRIOS PARA INCLUSÃO E EXCLUSÃO DE TRABALHOS

O critério de inclusão adotado foi a seleção de trabalhos que tinham como foco a criação de tutoriais para ensinar uma ferramenta/linguagem de programação. Por outro lado, o critério de exclusão consistiu em remover trabalhos que se limitavam a ensinar apenas conceitos teóricos. Além disso, foram excluídos trabalhos relacionados à mesma ferramenta, optando-se por aqueles considerados mais abrangentes. Textos em andamento também foram excluídos. Após a aplicação desses critérios, restaram 4 trabalhos, que são descritos a seguir.

3.2.1 Development and integration of Web-based software tutorials for an undergraduate curriculum: control tutorials for MATLAB

Neste trabalho, os autores (TILBURY; MESSNER, 1997) discutem o desenvolvimento, uso e distribuição de um conjunto de tutoriais baseados na web, bem como sua integração em um currículo de graduação para cursos relacionados a controle de automação.

O conceito inicial do tutorial teve início em 1995, quando os alunos da professora Tilbury estavam tendo dificuldades em utilizar o MATLAB para resolver seus problemas de casa. Foi então concebida a ideia de demonstrar um problema de exemplo no MA-

TLAB, mostrando todo o código utilizado, as etapas tomadas e os resultados gerados pela ferramenta.

Esse tutorial foi disponibilizado na internet para fácil acesso pelos estudantes. Devido à resposta extremamente positiva dos estudantes, a professora decidiu desenvolver um novo tutorial para cada tópico abordado no curso, resultando no conjunto de tutoriais desenvolvidos pelos autores, que representa uma expansão do conjunto original criado pela professora.

A metodologia descrita pelos autores enfatiza que os estudantes aprendem melhor quando podem utilizar rapidamente as informações apresentadas. Os tutoriais desenvolvidos possibilitam que os estudantes aprendam por meio de "visualização e prática". Eles consistem em trechos de código MATLAB acompanhados de explicações textuais, descrevendo o objetivo dos comandos e os resultados gerados (texto ou gráfico). Os estudantes são encorajados a copiar e experimentar o código disponibilizado, realizando alterações nos parâmetros e comandos. Além dos exemplos principais, também são fornecidos mais exemplos para que os alunos possam praticar.

A abordagem proposta pelos autores é que os alunos tenham o MATLAB e o tutorial abertos simultaneamente, permitindo uma transição fácil entre eles e a utilização das informações fornecidas pelo tutorial na ferramenta. Essa mesma abordagem será adotada na utilização do ProgressiveTutor, onde ele será utilizado junto ao ambiente de desenvolvimento das tecnologias abordadas.

3.2.2 Utilizing learning styles for interactive tutorials

Neste trabalho, os autores (AASE; KURFESS, 2004) introduziram um ambiente de aprendizado web para ensinar conceitos de inteligência artificial. Esse ambiente tinha como objetivo ser utilizado como uma ferramenta complementar às aulas padrões já existentes.

O ponto de foco era adaptar as instruções e materiais de estudo para cada estilo de aprendizagem dos estudantes, com a expectativa de que o tutorial aumentasse o conhecimento e interesse do usuário sobre o tópico estudado.

Esse ambiente de aprendizado foi testado em uma sala com 25 alunos, onde foi realizada uma pesquisa sobre o ambiente e coletado feedback sobre sua utilização. Os resultados obtidos mostraram que a grande maioria dos estudantes gostou do ambiente. Alguns pontos positivos citados foram sua utilidade para revisar conhecimentos aprendidos e a capacidade dos estudantes que não tinham o inglês como língua materna de poder visualizar repetidamente os vídeos dos instrutores explicando, acompanhados de slides semi-animados de PowerPoint, permitindo que eles acompanhassem os vídeos no ritmo que lhes fosse mais conveniente.

Um ponto levantado foi o fato de que alguns estudantes disseram que a categorização dos estilos de aprendizagem não condizia completamente com suas observa-

ções sobre suas preferências de estudo.

Devido à escala de complexidade do estudo realizado neste trabalho e com base nas observações feitas no estudo conduzido por (AASE; KURFESS, 2004), o aplicativo ProgressiveTutor, como decisão de projeto, não se adaptará às diferentes preferências de estilos de aprendizagem dos estudantes. Será desenvolvido com uma única abordagem para todos os usuários-alvo.

3.2.3 Interactive Online Tutorial Assistance for a First Programming Course

Neste trabalho, os autores (HULLS *et al.*, 2005) discutem as vantagens dos tutoriais online para enriquecer a experiência de estudo dos estudantes, sendo uma área de interesse a criação de tutoriais para praticar conhecimentos vistos anteriormente em aula, possibilitando assim um melhor aproveitamento do tempo em sala de aula. No entanto, eles mencionam que, devido ao tempo e aos custos envolvidos, o desenvolvimento de um sistema de tutorial que abrange todos os conhecimentos abordados tem viabilidade limitada.

O trabalho desenvolvido por eles tem o foco em módulos online de baixo custo para auxiliar as formas já existentes de ensino. A principal característica desse design é a limitação do tipo de tutorial criado, com ênfase nos desafios que envolvem a replicação de conceitos aprendidos durante as aulas regulares. Na área de introdução aos cursos de programação, eles destacam que essas dificuldades estão principalmente relacionadas ao aprendizado da sintaxe da linguagem de programação, assim como à familiarização com conceitos como loops, declarações condicionais e algoritmos simples.

Através da utilização desses módulos online, os autores buscavam reduzir a necessidade de repetição desses conceitos durante as aulas, permitindo que os estudantes os estudassem e compreendessem em seu próprio ritmo.

O tutorial desenvolvido por (HULLS *et al.*, 2005) deu ênfase principalmente aos elementos visuais, algo que será replicado no desenvolvimento do ProgressiveTutor.

3.2.4 Teaching Mobile Application Development through Lectures, Interactive Tutorials, and Pair Programming

Neste trabalho, os autores (SEYAM *et al.*, 2016) têm como foco a utilização de diferentes estilos de ensino para lidar com os desafios da área de desenvolvimento mobile, que combina aspectos de software, hardware e interações interpessoais, como desenvolver para múltiplos tamanhos de tela, projetar um aplicativo considerando o tempo de resposta do GPS e lidar com dados imprecisos gerados pelos sensores.

Para enfrentar esses diferentes desafios, foram utilizados três tipos diferentes de ensino: palestras, tutoriais interativos e programação em pares. As palestras foram utilizadas para introduzir tópicos teóricos de desenvolvimento, permitindo tempo

para perguntas e discussões dos alunos, mas sem envolver componentes de aprendizado ativo. Duas formas de aprendizado ativo foram utilizadas: tutoriais interativos e programação em pares. Os tutoriais interativos apresentavam conceitos aplicados e exploravam seu uso em atividades individuais, com demonstrações práticas desses conceitos. Por fim, a programação em pares, na qual os participantes assumiam papéis alternados de codificação e revisão do código, foi utilizada para praticar novos conceitos de programação.

Ao final deste trabalho, os autores observaram que a fundamentação teórica era mais adequada para palestras, enquanto os conceitos de programação eram mais bem absorvidos em situações de aprendizado ativo.

O trabalho desenvolvido por (SEYAM *et al.*, 2016) teve foco principal na programação em pares. No entanto, devido às limitações de complexidade deste trabalho, o ProgressiveTutor se concentrará apenas na parte relacionada aos tutoriais.

Parte II

Implementação

4 ESPECIFICAÇÕES E REQUISITOS

Neste capítulo, serão abordados os métodos e ferramentas utilizados no desenvolvimento dos sistemas, bem como os requisitos a serem implementados. É importante ressaltar que este trabalho consiste em quatro sistemas independentes: o engine de tutoriais ProgressiveTutor e três aplicativos desenvolvidos através dos módulos do ProgressiveTutor. Cada módulo corresponde a uma área de conhecimento específica: desenvolvimento multiplataforma, desenvolvimento desktop e desenvolvimento mobile.

Os três aplicativos desenvolvidos representam a aplicação prática do conhecimento adquirido em cada módulo do ProgressiveTutor. Cada aplicativo é desenvolvido utilizando um método distinto, embora compartilhem requisitos semelhantes. O primeiro aplicativo é focado no desenvolvimento multiplataforma, o segundo no desenvolvimento desktop e o terceiro no desenvolvimento mobile.

Esses quatro sistemas são independentes entre si, ou seja, não se comunicam diretamente. No entanto, o ProgressiveTutor serve como base e guia para a criação dos três aplicativos.

Nas próximas seções, serão detalhados os métodos adotados para o desenvolvimento de cada sistema, assim como os requisitos específicos de cada um.

4.1 REQUISITOS DO PROGRESSIVETUTOR

O ProgressiveTutor é um sistema que apresenta uma complexidade maior em termos de quantidade de funcionalidades se comparado aos três sistemas ensinados por ele. No entanto, em relação à dificuldade de implementação técnica, ele é considerado mais simples. Esta seção abordará os requisitos funcionais do sistema, separando-os pelos módulos contemplados. Em seguida, serão discutidos alguns requisitos não funcionais que o ProgressiveTutor deverá atender durante seu desenvolvimento.

O ProgressiveTutor é uma engine de tutoriais multiplataforma, modular e expansível, que possui uma ampla cobertura de assuntos a serem ensinados. No entanto, ele aborda cada assunto de forma superficial, tendo como objetivo principal capacitar os usuários a criar um aplicativo simples de anotações temáticas, abrangendo o desenvolvimento do front-end, back-end, uma API para comunicação e autenticação.

Conforme mencionado anteriormente, o ProgressiveTutor é composto por três módulos, cada um deles voltado para um tipo específico de tecnologia ensinada (multiplataforma, desktop e mobile). A tabela abaixo apresenta os requisitos funcionais listados para cada módulo:

Módulo	**Descrição**
Módulo Desktop	
RF01	O sistema deve ensinar como instalar as ferramentas para programar em desktop
RF02	O sistema deve ensinar o básico na ferramenta escolhida para desktop
RF03	O sistema deve ensinar como fazer uma interface front-end
RF04	O sistema deve ensinar como configurar um back-end
RF05	O sistema deve ensinar como interligar o front-end e o back-end
RF06	O sistema deve ensinar como realizar operações com o back-end utilizando autenticação
Módulo Multiplataforma	
RF07	O sistema deve ensinar como instalar as ferramentas para programar multiplataforma
RF08	O sistema deve ensinar o básico na ferramenta escolhida para multiplataforma
RF09	O sistema deve ensinar como fazer uma interface front-end para multiplataforma
RF10	O sistema deve ensinar como configurar um back-end para multiplataforma
RF11	O sistema deve ensinar como interligar o front-end e o back-end para multiplataforma
RF12	O sistema deve ensinar como realizar operações com o back-end utilizando autenticação para multiplataforma
Módulo Mobile	
RF13	O sistema deve ensinar como instalar as ferramentas para programar mobile
RF14	O sistema deve ensinar o básico na ferramenta escolhida para mobile
RF15	O sistema deve ensinar como fazer uma interface front-end para mobile
RF16	O sistema deve ensinar como configurar um back-end para mobile
RF17	O sistema deve ensinar como interligar o front-end e o back-end para mobile
RF18	O sistema deve ensinar como realizar operações com o back-end utilizando autenticação para mobile

Tabela 4 – Requisitos Funcionais do ProgressiveTutor.

4.1.1 Requisitos Não-Funcionais do ProgressiveTutor

A Tabela 4 apresenta os requisitos não funcionais do ProgressiveTutor. Foram identificados quatro requisitos relacionados à implementação e à tecnologia a ser utilizada, além de um requisito específico sobre o próprio produto, que deve ser totalmente utilizável sem a necessidade de conexão com a internet.

RNF01	O sistema deve ser implementado utilizando Next.js
RNF02	O sistema deve permitir alternar entre as tecnologias ensinadas
RNF03	O sistema deve funcionar offline após ter carregado uma vez
RNF04	O sistema não pode usar banco de dados
RNF05	O sistema não pode usar imagens para economizar o consumo de dados ao utilizá-lo

Tabela 6 – Requisitos Não Funcionais do ProgressiveTutor.

4.2 REQUISITOS DOS TRÊS SISTEMAS DESENVOLVIDOS

Esta seção trata dos requisitos funcionais dos sistemas, separando-os pelos módulos contemplados. Em seguida, serão discutidos os requisitos não funcionais que os sistemas desenvolvidos devem cumprir durante o seu desenvolvimento.

4.2.1 Requisitos Funcionais dos Três Sistemas Desenvolvidos

Os requisitos funcionais dos três sistemas desenvolvidos são apresentados na Tabela 7.

RF01	O sistema deve ser implementado seguindo os passos apresentados pelo ProgressiveTutor
RF02	O sistema deve permitir o cadastro de novos usuários
RF03	O sistema deve permitir o login de usuários cadastrados
RF04	O sistema deve permitir o cadastro de tópicos associados a um único usuário e esta operação só pode ser feita por usuários cadastrados
RF05	O sistema deve permitir a edição de tópicos associados a um único usuário e esta operação só pode ser feita pelo criador do tópico
RF06	O sistema deve permitir a exclusão de tópicos associados a um único usuário e esta operação só pode ser feita pelo criador do tópico
RF07	O sistema deve permitir o cadastro de notas associadas a um único tópico e esta operação só pode ser feita pelo criador do tópico
RF08	O sistema deve permitir a edição de notas associadas a um único tópico e esta operação só pode ser feita pelo criador do tópico
RF09	O sistema deve permitir a exclusão de notas associadas a um único tópico e esta operação só pode ser feita pelo criador do tópico
RF10	O sistema deve excluir todas as notas associadas a um tópico quando o tópico for excluído e esta operação só pode ser feita pelo criador do tópico

Tabela 7 – Requisitos Funcionais dos Três Sistemas Desenvolvidos.

4.2.1.1 Requisitos Não-Funcionais do Sistema Multiplataforma

Os requisitos não funcionais do sistema multiplataforma são apresentados na Tabela 8.

RNF01	O sistema deve ser implementado utilizando Next.js
RNF02	O sistema deve utilizar o banco de dados na nuvem PostgreSQL
RNF03	O sistema deve incorporar um sistema de autenticação

Tabela 8 – Requisitos Não-Funcionais do Sistema Multiplataforma.

4.2.1.2 Requisitos Não-Funcionais do Sistema Desktop

Os requisitos não funcionais do sistema desktop são apresentados na Tabela 9.

RNF01	O sistema deve ser implementado utilizando C#
RNF02	O sistema deve utilizar o banco de dados na nuvem PostgreSQL
RNF03	O sistema deve incorporar um sistema de autenticação

Tabela 9 – Requisitos Não-Funcionais do Sistema Desktop.

4.2.1.3 Requisitos Não-Funcionais do Sistema Mobile

Os requisitos não funcionais do sistema mobile são apresentados na Tabela 10.

RNF01	O sistema deve ser implementado utilizando Kotlin
RNF02	O sistema deve utilizar o banco de dados na nuvem PostgreSQL
RNF03	O sistema deve incorporar um sistema de autenticação

Tabela 10 – Requisitos Não-Funcionais do Sistema Mobile.

4.3 MÉTODOS DE DESENVOLVIMENTO

Esta seção aborda as principais tecnologias e abordagens de desenvolvimento utilizadas na elaboração deste trabalho.

4.3.1 Métodos do ProgressiveTutor

O aplicativo ProgressiveTutor será desenvolvido utilizando a linguagem de programação Next.js, com o auxílio do ambiente de desenvolvimento Visual Studio Code. A interface gráfica será construída de forma declarativa, aproveitando as capacidades fornecidas pelo Next.js. O aplicativo será dividido em três módulos, cada um dedicado a uma das tecnologias ensinadas pelo ProgressiveTutor. Para o preparo do ambiente de desenvolvimento, será utilizada a ferramenta de linha de comando "create t3-app"(T3, 2023), que facilita as etapas iniciais do desenvolvimento. O aplicativo será hospedado na plataforma Vercel, que oferece um plano gratuito e um painel de controle de deploy poderoso.

4.3.2 Métodos do Sistema Multiplataforma

O aplicativo de tomada de notas e tópicos a ser desenvolvido para ser multiplataforma seguirá as instruções do módulo multiplataforma do ProgressiveTutor. Será utilizada a linguagem de programação Next.js, assim como no ProgressiveTutor. A ferramenta "create t3-app"(T3, 2023) também será utilizada, juntamente com o Visual Studio Code. No entanto, diferentemente do ProgressiveTutor, o aplicativo multiplataforma fará uso completo dessa ferramenta, incluindo recursos relacionados a bancos

de dados, autenticação e comunicação com o banco de dados. Assim como o ProgressiveTutor, o aplicativo multiplataforma será hospedado na plataforma Vercel para permitir o acesso online.

4.3.3 Métodos do Sistema Desktop

Este tópico abordará os métodos utilizados no desenvolvimento do sistema desktop.

O aplicativo de tomada de notas e tópicos a ser desenvolvido para desktop seguirá as instruções do módulo multiplataforma do ProgressiveTutor, que indica o uso da linguagem de programação C# e do ambiente de desenvolvimento Visual Studio. O Visual Studio possui uma ferramenta de construção de layouts no estilo "drag and drop", o que facilitará o trabalho. Para hospedar o banco de dados (PostgreSQL) que será utilizado pelo aplicativo, será utilizado o serviço de hospedagem Railway, que oferece um template para configuração do back-end com PostgreSQL e configura automaticamente a parte básica do Strapi. O Strapi será utilizado para fazer a intermediação entre a aplicação e o banco de dados, utilizando as APIs REST fornecidas pelo Strapi. Esta escolha foi feita considerando a possibilidade de usar o mesmo back-end e meio de comunicação tanto para o aplicativo desktop quanto para o aplicativo mobile.

4.3.4 Métodos do Sistema Mobile

O aplicativo de tomada de notas e tópicos a ser desenvolvido para dispositivos móveis seguirá as instruções do módulo multiplataforma do ProgressiveTutor. Será utilizada a linguagem de programação Kotlin e o ambiente de desenvolvimento Android Studio, que possui uma ferramenta de construção de layouts no estilo "drag and drop"e também permite a criação de layouts de forma declarativa, assim como o Next.js. Para hospedar o banco de dados e estabelecer a comunicação com o aplicativo, o Mobile utilizará as mesmas escolhas do aplicativo desktop, utilizando o Railway e o Strapi.

5 DESENVOLVIMENTO

Este capítulo detalha o desenvolvimento de todo o projeto. Primeiramente, serão abordados os detalhes do ProgressiveTutor, incluindo seus comportamentos e implementação. Em seguida, será apresentado o desenvolvimento de cada sistema individual desenvolvido com o auxílio do ProgressiveTutor, abordando suas características e particularidades.

5.1 PROGRESSIVETUTOR

O ProgressiveTutor é a parte central proposta por este trabalho, com o objetivo de facilitar o aprendizado de diferentes tecnologias (multiplataforma, desktop e mobile), conforme discutido na seção de introdução. Ele é voltado para estudantes de Ciência da Computação que já possuem uma base em programação, redes e segurança digital, adquirida em disciplinas obrigatórias ou optativas, e que desejam aprofundar seus conhecimentos por meio da criação dos três aplicativos ensinados pelo ProgressiveTutor.

5.1.1 Comportamento do ProgressiveTutor

O ProgressiveTutor é um aplicativo tutorial desenvolvido em Next.js. Seu fluxo de funcionamento básico pode ser compreendido através das Figuras 2 e 3, onde o usuário inicia o aplicativo na página inicial, que exibe três botões correspondentes às três tecnologias ensinadas. Ao clicar em uma das opções, o usuário é redirecionado para a página específica daquela tecnologia, onde serão apresentados os passos do tutorial na forma de texto. Por exemplo, no caso do sistema multiplataforma, os primeiros passos incluem o download do Next.js, a introdução ao JavaScript, a familiarização com o Next.js, conceitos básicos sobre programação multiplataforma e os passos para a criação do aplicativo. Além do texto explicativo, os botões na página permitem que o usuário volte para a página anterior, retorne à tela inicial ou avance para a próxima etapa do tutorial.

Figura 2 – Tela Inicial do ProgressiveTutor

Fonte: o autor

Figura 3 – Tela do ProgressiveTutor para Aplicação Multiplataforma

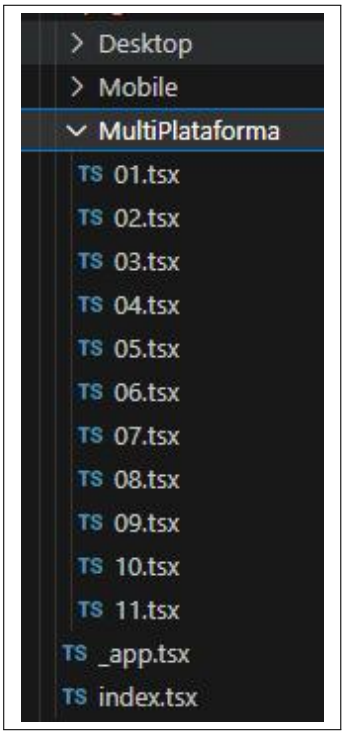
Fonte: o autor

5.1.2 Implementação do ProgressiveTutor

A implementação do ProgressiveTutor começou utilizando o recurso de criação rápida "create t3-app". A partir daí, foram desenvolvidas a página inicial e as implementações dos sistemas web e desktop. Durante esse processo, foram buscadas informações sobre a implementação do sistema mobile. Após a conclusão dos dois primeiros sistemas, o foco foi direcionado para a implementação dos tutoriais correspondentes.

A estrutura do tutorial permite uma fácil expansão, caso seja desejado adicionar mais etapas entre as existentes para suavizar a transição ou adicionar etapas adicionais no final para aprofundar ainda mais o conteúdo. A estrutura dos arquivos do tutorial pode ser observada na Figura 4, onde todas as páginas relacionadas a um sistema são agrupadas em suas próprias pastas.

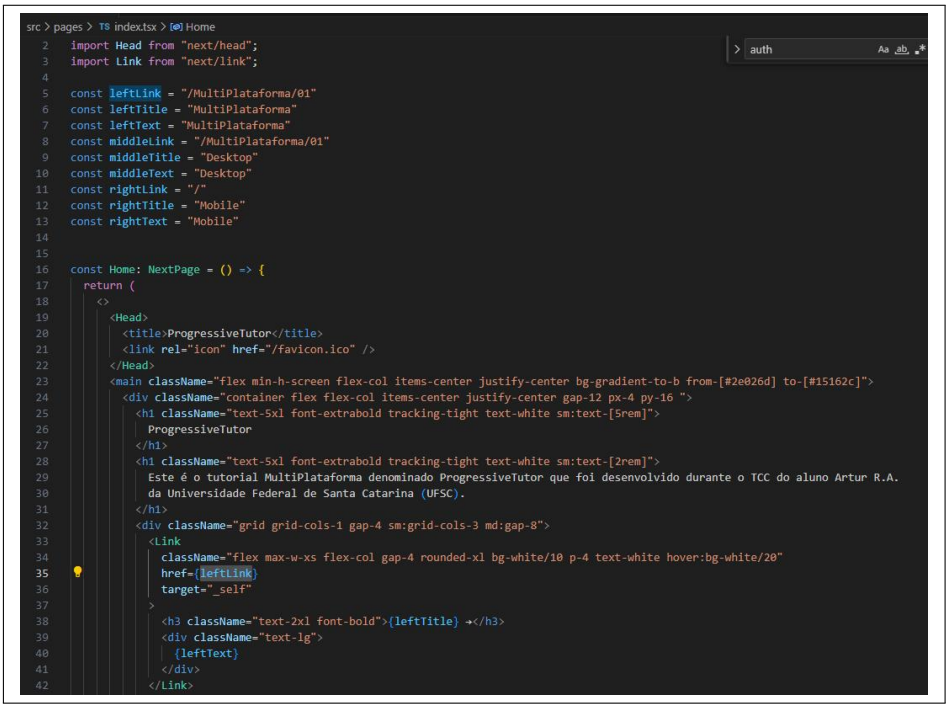
Figura 4 – Organização das Páginas do ProgressiveTutor



Fonte: o autor

A transição entre as páginas do tutorial é realizada por meio de links para outras páginas do aplicativo. Ao pressionar o botão correspondente, a transição é ativada, como ilustrado na Figura 5.

Figura 5 – Mecanismo de Mudança de Página no ProgressiveTutor



```
src > pages > TS index.tsx > Home
import Head from "next/head";
import Link from "next/link";

const leftLink = "/MultiPlataforma/01"
const leftTitle = "MultiPlataforma"
const leftText = "MultiPlataforma"
const middleLink = "/MultiPlataforma/01"
const middleTitle = "Desktop"
const middleText = "Desktop"
const rightLink = "/"
const rightTitle = "Mobile"
const rightText = "Mobile"


const Home: NextPage = () => {
  return (
    <>
      <Head>
        <title>ProgressiveTutor</title>
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <main className="flex min-h-screen flex-col items-center justify-center bg-gradient-to-b from-[#2e026d] to-[#15162c]">
        <div className="container flex flex-col items-center justify-center gap-12 px-4 py-16 ">
          <h1 className="text-5xl font-extrabold tracking-tight text-white sm:text-[5rem]">
            ProgressiveTutor
          </h1>
          <h1 className="text-5xl font-extrabold tracking-tight text-white sm:text-[2rem]">
            Este é o tutorial MultiPlataforma denominado ProgressiveTutor que foi desenvolvido durante o TCC do aluno Artur R.A.
            da Universidade Federal de Santa Catarina (UFSC).
          </h1>
          <div className="grid grid-cols-1 gap-4 sm:grid-cols-3 md:gap-8">
            <Link
              className="flex max-w-xs flex-col gap-4 rounded-xl bg-white/10 p-4 text-white hover:bg-white/20"
              href={leftLink}
              target="_self"
            >
              <h3 className="text-2xl font-bold">{leftTitle} →</h3>
              <div className="text-lg">
                {leftText}
              </div>
            </Link>
```

Fonte: o autor

5.2 SISTEMA MULTIPLATAFORMA

O Sistema Multiplataforma é um dos três tipos de sistemas ensinados pelo ProgressiveTutor. Dentre os três, é o mais distante do conteúdo abordado no curso de Ciência da Computação, pois requer conhecimento intermediário a avançado em JavaScript e React. Assim como nos outros dois sistemas, ao final do tutorial da área de desenvolvimento multiplataforma, o ProgressiveTutor ensina como criar um aplicativo simples para fazer anotações, permitindo que o usuário registre temas e suas respectivas notas. Esse aplicativo possui funcionalidades como um banco de dados online utilizando o Supabase, uma interface de usuário declarativa utilizando os componentes disponíveis no Next.js, uma forma de comunicação entre o frontend e o backend utilizando o tRPC e o NextAuth.js para autenticação.

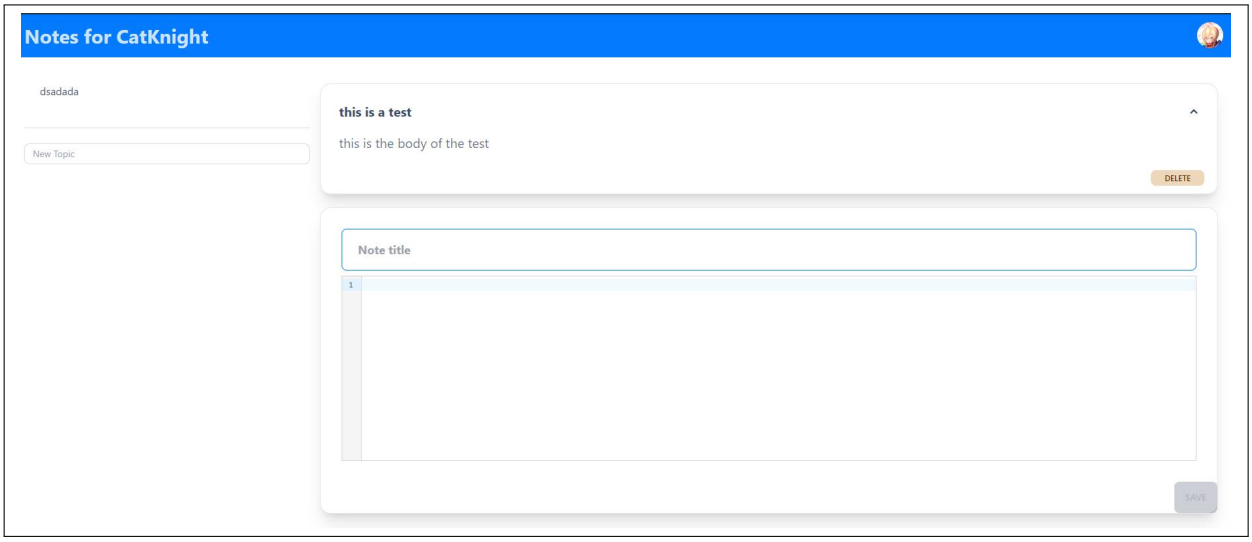
5.2.1 Comportamento do Sistema Multiplataforma

O aplicativo desenvolvido com o ProgressiveTutor, do tipo multiplataforma, utiliza o Next.js como framework principal. A Figura 6 mostra a tela principal do aplicativo. O processo de login é realizado clicando no ícone localizado no canto superior direito (no exemplo da figura, o ícone corresponde à conta GitHub do usuário logado), que redireciona o usuário para a página de login controlada pelo NextAuth.js.

Após fazer o login, o usuário pode criar um novo tópico clicando no campo de

texto com a indicação "Novo Tópico"e inserindo o nome desejado para o tópico. Ao pressionar Enter, é possível criar uma nova nota relacionada a esse tópico. Para isso, o usuário precisa selecionar o tópico desejado, preencher o título e o conteúdo da nota e, em seguida, clicar no botão "Salvar". É importante mencionar que todas as operações de criação, edição e exclusão utilizam o tRPC para se comunicar com o banco de dados PostgreSQL hospedado no Supabase.

Figura 6 – Tela do Aplicativo Multiplataforma



Fonte: o autor

5.2.2 Implementação do Sistema Multiplataforma

A implementação do Sistema Multiplataforma foi iniciada utilizando o recurso de criação rápida fornecido pelo create t3-app. No entanto, ao contrário do ProgressiveTutor, todas as opções foram utilizadas, incluindo tRPC, NextAuth.js, Prisma e Tailwind. A partir daí, foram identificadas as partes da tela que se repetiriam e foram criados componentes para representá-las. Um exemplo de código de um componente pode ser visualizado na Figura 7, onde a aparência e o comportamento do componente foram declarados de forma declarativa. Além disso, é importante mencionar que a aplicação utiliza o padrão de design chamado State, que permite que um objeto altere seu comportamento quando seu estado interno muda. Isso pode ser observado, por exemplo, no botão de salvar da nota, que é desabilitado quando pelo menos um dos campos (título da nota ou corpo da nota) está vazio. O design da página inicial foi concebido de forma a evitar a necessidade de criar diálogos para interagir com a aplicação.

Figura 7 – Componente NoteCard

```tsx
import { useState } from "react";

import ReactMarkdown from "react-markdown";

import { type RouterOutputs } from "../utils/api";

type Note = RouterOutputs["note"]["getAll"][0];

export const NoteCard = ({
  note,
  onDelete,
}: {
  note: Note;
  onDelete: () => void;
}) => {
  const [isExpanded, setIsExpanded] = useState<boolean>(true);

  return (
    <div className="card mt-5 border border-gray-200 bg-base-100 shadow-xl">
      <div className="card-body m-0 p-3">
        <div
          className={`collapse-arrow ${
            isExpanded ? "collapse-open" : ""
          } collapse`}
          onClick={() => setIsExpanded(!isExpanded)}
        >
          <div className="collapse-title text-xl font-bold">{note.title}</div>
          <div className="collapse-content">
            <article className="prose lg:prose-xl">
              <ReactMarkdown>{note.content}</ReactMarkdown>
            </article>
          </div>
        </div>
        <div className="card-actions mx-2 flex justify-end">
          <button className="btn-warning btn-xs btn px-5" onClick={onDelete}>
            Delete
          </button>
        </div>
      </div>
    </div>
  );
};

```

Fonte: o autor

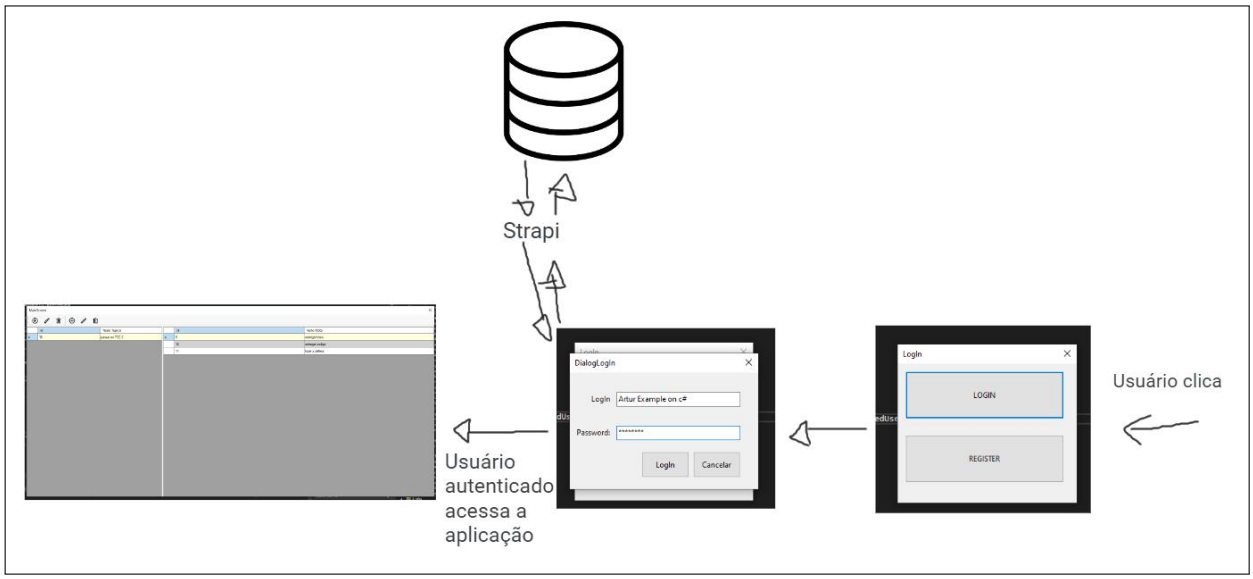
5.3 SISTEMA DESKTOP

O Sistema Desktop é um dos três tipos de sistemas ensinados pelo ProgressiveTutor. Entre os três, é o mais próximo do que é abordado no curso de Ciências da Computação. Assim como nos outros dois sistemas, ao final do tutorial da área de desenvolvimento desktop, o ProgressiveTutor ensina como criar um aplicativo simples para fazer anotações de temas e notas, que possui funcionalidades como um banco de dados online, uma interface estilo Windows Form feita no Visual Studio, uma forma de comunicação entre ambos utilizando o Strapi com chamadas REST, e uma estrutura de arquivos organizada seguindo o padrão MVC, com a adição da entidade Repositorio.

5.3.1 Comportamento do Sistema Desktop

O aplicativo desenvolvido com o ProgressiveTutor do tipo Desktop é uma aplicação em C# com um fluxo básico exemplificado na Figura 8, que ilustra o processo de login de um usuário. Nesse processo, o usuário escolhe entre fazer login ou criar uma conta, e em seguida entra no sistema com essa nova conta. Após essa escolha, as informações necessárias do diálogo são enviadas para o banco de dados por meio do Strapi, que retorna a resposta.

Figura 8 – Diagrama do Funcionamento do Login no Sistema Desktop



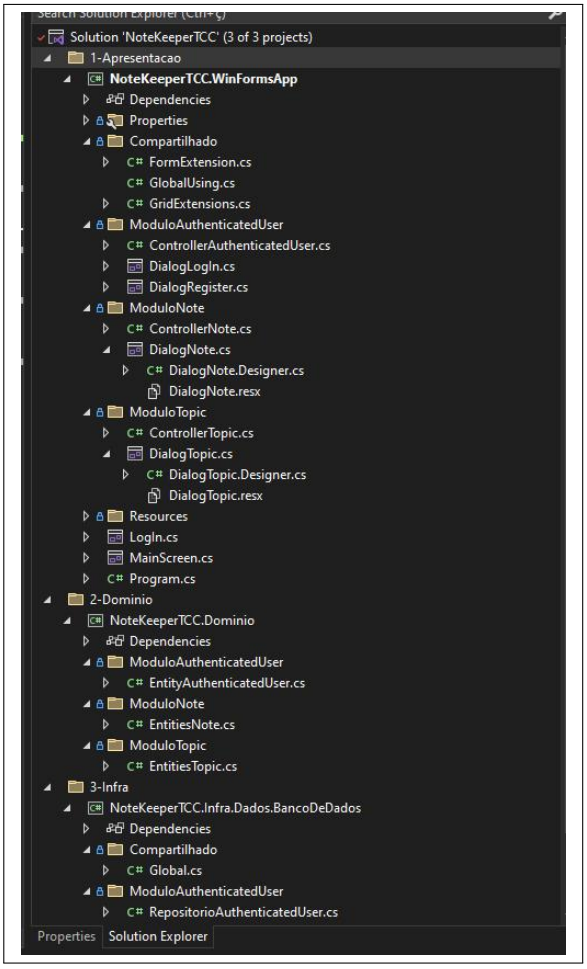
Fonte: o autor

5.3.2 Implementação do Sistema Desktop

A implementação do Sistema Desktop começa com a configuração dos sistemas no site Railway. Nesse processo, é configurado o banco de dados PostgreSQL e estabelecida a conexão com o repositório "strapitcc"no GitHub, que contém o código que descreve o comportamento do Strapi. Em seguida, o Strapi é inicializado no modo de desenvolvedor para utilizar seu sistema de controle de entidades interativo e configurar as entidades do banco de dados, bem como as relações entre elas. Feito isso, é feita uma esquematização das partes da aplicação, optando-se por utilizar a arquitetura MVC para melhor separar as responsabilidades. Além disso, é adicionada uma entidade adicional chamada Repositorio. A aplicação é dividida em três partes principais: Apresentação, Domínio e Infraestrutura. Na parte de Apresentação, responsável pelo View e Controller, são desenvolvidos os principais elementos de UI. A parte de Domínio, responsável pelos Models (denominados Entity nesta aplicação), é criada como uma biblioteca para ser usada pelas outras partes da aplicação. A parte de In-

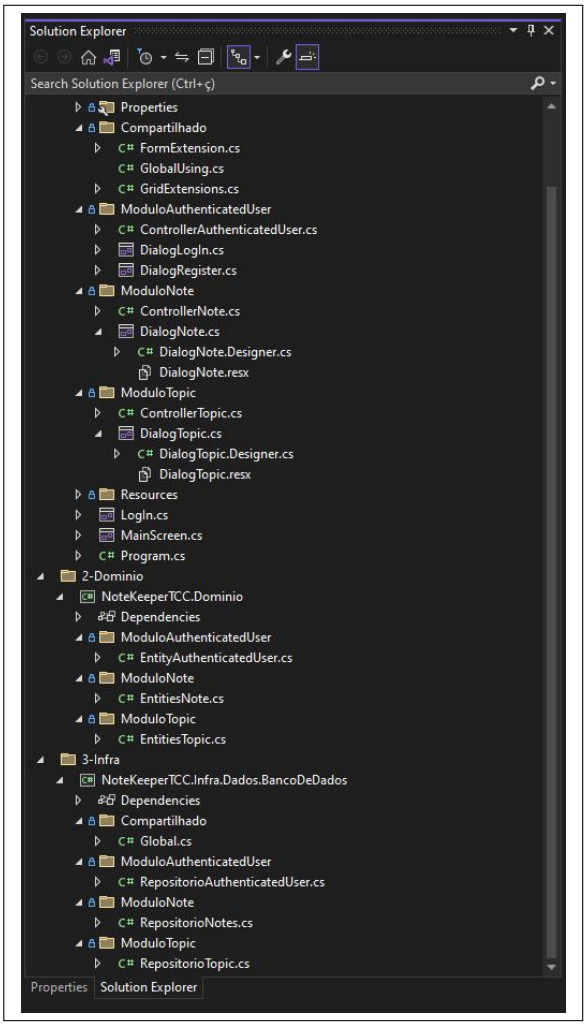
fraestrutura é responsável pelos Repositórios e também é criada como uma biblioteca. Como exemplo, será demonstrada cada parte relacionada à classe modelo chamada EntityAuthenticatedUser.

Figura 9 – Estrutura do Projeto - Parte 1



Fonte: o autor

Figura 10 – Estrutura do Projeto - Parte 2



Fonte: o autor

5.3.2.1 Modelo

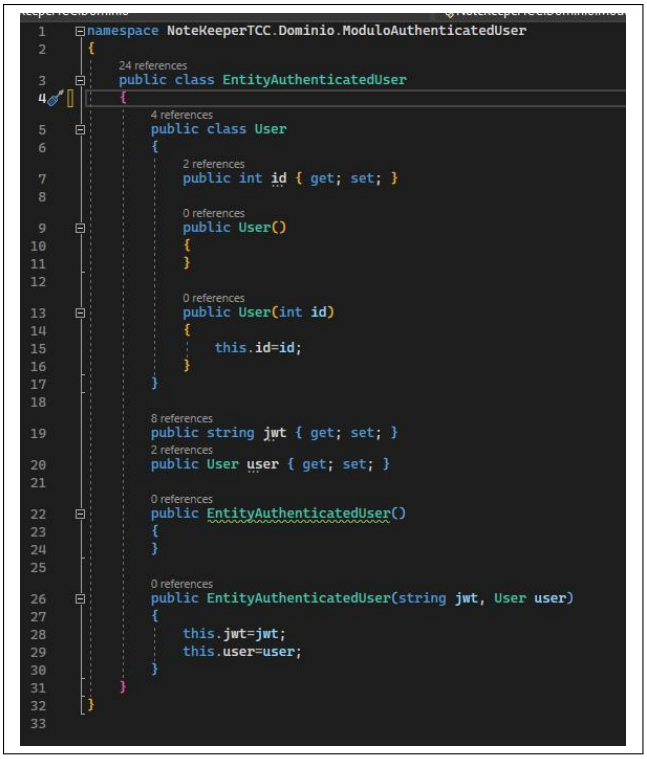
A seção do modelo, com a adição da entidade Repositorio, é responsável pela estruturação da entidade e alterações. Nesta aplicação específica, ela é utilizada como molde para fazer a desserialização da entidade retornada pelo banco de dados quando um pedido REST é realizado.

5.3.2.1.1 Modelo Usuário Autenticado

A classe EntityAuthenticatedUser é estruturada da seguinte forma, conforme ilustrado na Figura 11. Essa estrutura, assim como a estrutura das outras entidades, foi obtida fazendo uma chamada ao endpoint correspondente a cada entidade. No caso específico dessa classe, o endpoint utilizado foi "api/auth/local". Durante essa chamada, foram aplicadas customizações para filtrar as informações desnecessárias, utilizando

a documentação do Strapi (STRAPI, 2023b). Em seguida, o retorno foi utilizado em conjunto com o site (JSON2CSHARP, 2023) para obter uma base da estrutura de cada entidade. Para o processo da classe EntityAuthenticatedUser, é necessário que ela possua um JWT e um User com um ID. Essa estrutura é essencial para realizar a desserialização da resposta de uma solicitação de criação ou login de um usuário existente.

Figura 11 – Entidade Usuário Autenticado



Fonte: o autor

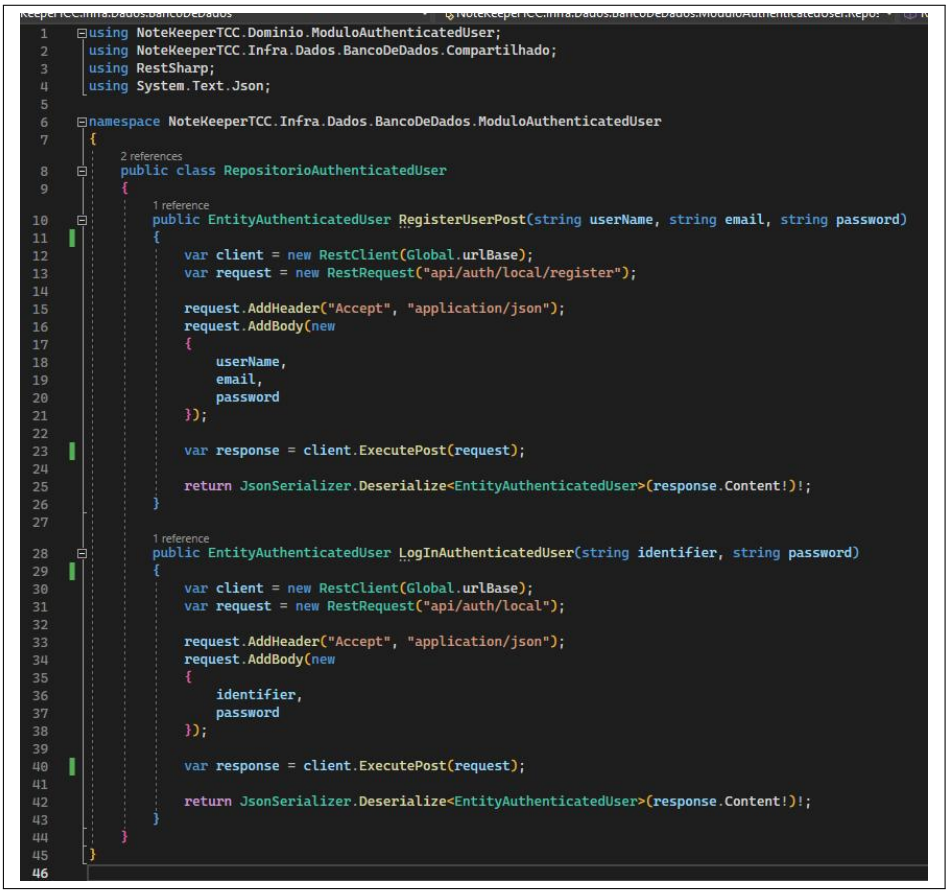
5.3.2.2 Repositório

Como explicado anteriormente, para cada entidade existe um repositório correspondente, que contém a lógica da entidade, além das funcionalidades de alterações de dados no banco de dados. O repositório é responsável pela serialização e desserialização da entidade correspondente ao fazer chamadas para o banco de dados utilizando o Strapi.

5.3.2.2.1 Repositório Usuário Autenticado

É possível ver na Figura 12 a estrutura da classe e como ocorrem as chamadas de login e registro de novos usuários.

Figura 12 – Entidade Repositório Usuário Autenticado



```
using NoteKeeperTCC.Dominio.ModuloAuthenticatedUser;
using NoteKeeperTCC.Infra.Dados.BancoDeDados.Compartilhado;
using RestSharp;
using System.Text.Json;

namespace NoteKeeperTCC.Infra.Dados.BancoDeDados.ModuloAuthenticatedUser
{
    2 references
    public class RepositorioAuthenticatedUser
    {
        1 reference
        public EntityAuthenticatedUser RegisterUserPost(string userName, string email, string password)
        {
            var client = new RestClient(Global.urlBase);
            var request = new RestRequest("api/auth/local/register");

            request.AddHeader("Accept", "application/json");
            request.AddBody(new
            {
                userName,
                email,
                password
            });

            var response = client.ExecutePost(request);

            return JsonSerializer.Deserialize<EntityAuthenticatedUser>(response.Content!)!;
        }

        1 reference
        public EntityAuthenticatedUser LogInAuthenticatedUser(string identifier, string password)
        {
            var client = new RestClient(Global.urlBase);
            var request = new RestRequest("api/auth/local");

            request.AddHeader("Accept", "application/json");
            request.AddBody(new
            {
                identifier,
                password
            });

            var response = client.ExecutePost(request);

            return JsonSerializer.Deserialize<EntityAuthenticatedUser>(response.Content!)!;
        }
    }
}
```

Fonte: o autor

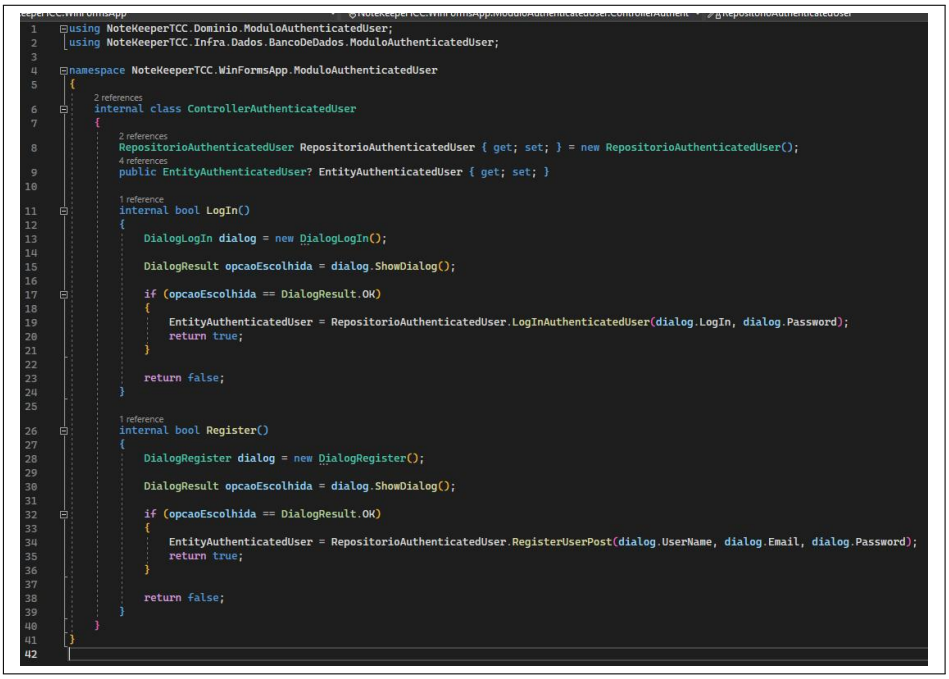
5.3.2.3 Controller

Assim como o repositório, para cada entidade existe um controlador correspondente, que contém os controles mais gerais. Ele se comunica com o repositório correspondente para obter o retorno da comunicação com o banco de dados e repassá-lo à visão.

5.3.2.3.1 Controller Usuário Autenticado

É possível ver na Figura 13 a estrutura da classe e como ocorrem as chamadas de login e registro de novos usuários, que chamam os métodos correspondentes do repositório.

Figura 13 – Entidade Controller Usuário Autenticado



```
using NoteKeeperTCC.Dominio.ModuloAuthenticatedUser;
using NoteKeeperTCC.Infra.Dados.BancoDeDados.ModuloAuthenticatedUser;

namespace NoteKeeperTCC.WinFormsApp.ModuloAuthenticatedUser
{
    2 references
    internal class ControllerAuthenticatedUser
    {
        2 references
        RepositorioAuthenticatedUser RepositorioAuthenticatedUser { get; set; } = new RepositorioAuthenticatedUser();
        4 references
        public EntityAuthenticatedUser? EntityAuthenticatedUser { get; set; }

        1 reference
        internal bool LogIn()
        {
            DialogLogIn dialog = new DialogLogIn();

            DialogResult opcaoEscolhida = dialog.ShowDialog();

            if (opcaoEscolhida == DialogResult.OK)
            {
                EntityAuthenticatedUser = RepositorioAuthenticatedUser.LogInAuthenticatedUser(dialog.LogIn, dialog.Password);
                return true;
            }

            return false;
        }

        1 reference
        internal bool Register()
        {
            DialogRegister dialog = new DialogRegister();

            DialogResult opcaoEscolhida = dialog.ShowDialog();

            if (opcaoEscolhida == DialogResult.OK)
            {
                EntityAuthenticatedUser = RepositorioAuthenticatedUser.RegisterUserPost(dialog.UserName, dialog.Email, dialog.Password);
                return true;
            }

            return false;
        }
    }
}
```

Fonte: o autor

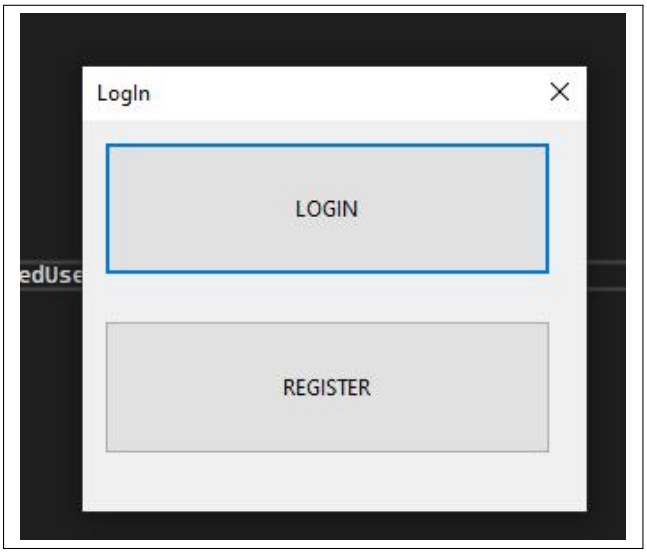
5.3.2.4 View

As interfaces gráficas do sistema foram implementadas utilizando a ferramenta de criação de UI arrastar e soltar embutida no Visual Studio, que permite agilizar o processo de criação das interfaces. Esta seção destaca os principais pontos de desenvolvimento do Sistema Desktop. Ao final desta seção, serão apresentados os principais elementos de interface construídos para esta aplicação.

5.3.2.5 Visão Geral

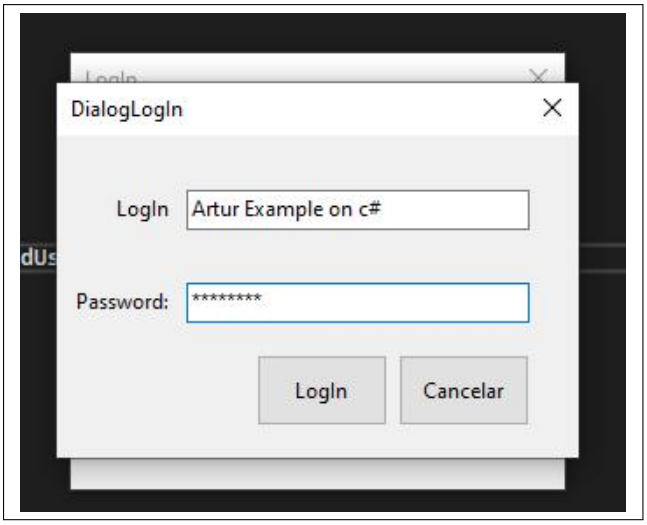
Primeiramente, foi desenvolvida a tela inicial, que funciona como a tela de login e registro de novos usuários, conforme mostrado na Figura 14. Para realizar as ações de login e registro, optou-se por utilizar diálogos, como ilustrado na Figura 15.

Figura 14 – Tela Login ou Registro de Usuários



Fonte: o autor

Figura 15 – Tela Login



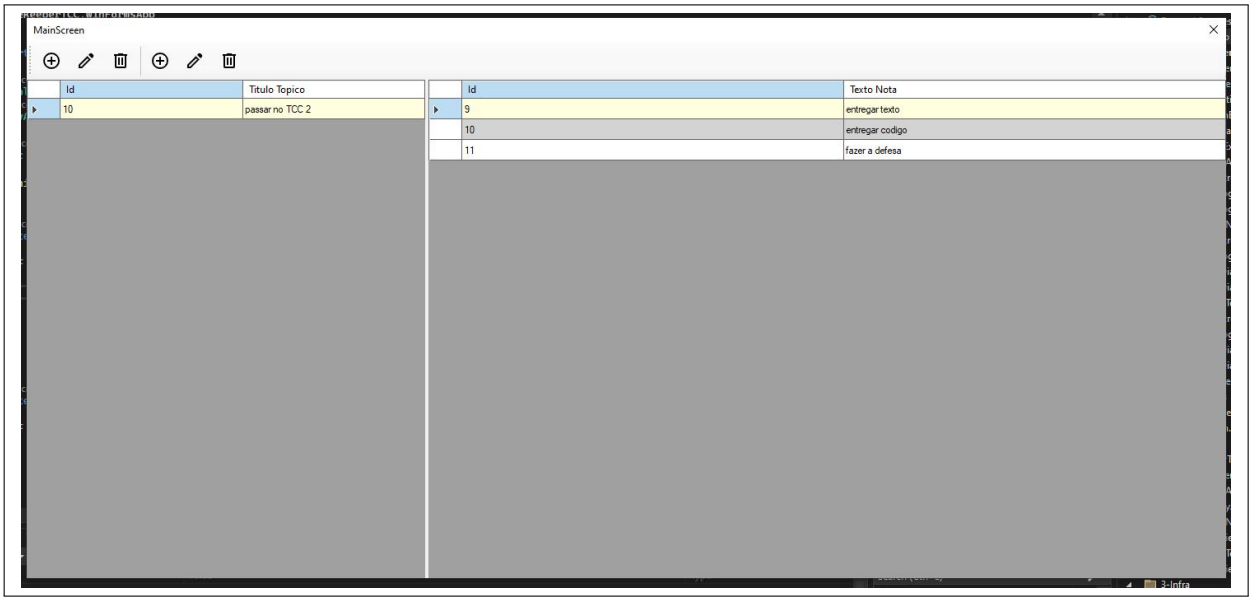
Fonte: o autor

Após efetuar o login ou registro, o aplicativo transita para uma segunda tela chamada MainScreen, onde estão localizadas duas estruturas do tipo DataGridView, responsáveis por exibir a lista de tópicos e a lista de notas relacionadas ao tópico selecionado. As operações de criação, edição e exclusão de tópicos são realizadas por meio dos três primeiros botões no topo da tela, e as mesmas operações podem ser executadas para as notas utilizando os três últimos botões. Na Figura 16, é possível visualizar a aparência da tela quando existem tópicos e notas presentes. Na Figura 17, é possível visualizar a continuação da tela com mais elementos de interface.

Figura 16 – Tela para manipulação de tópicos e notas 1

Fonte: o autor

Figura 17 – Tela para manipulação de tópicos e notas 2



Fonte: o autor

5.4 SISTEMA MOBILE

O Sistema Mobile é um dos três tipos de sistemas que o ProgressiveTutor ensina, e entre os três é o mais próximo ao que é ensinado no curso de Ciências da Computação. Assim como os outros dois sistemas, ao final do tutorial da área desktop,

o ProgressiveTutor ensina como fazer um aplicativo simples para fazer anotações de temas e notas, tendo as funcionalidades de um banco de dados online, uma interface construída através do construtor de UI arrastar e soltar do Android Studio, e uma API para se comunicar entre as duas partes usando o Strapi e chamadas REST, além do controle de autenticação por meio de JWT enviado junto com chamadas REST que precisam de autenticação.

5.4.1 Comportamento e Implementação do Sistema Mobile

Infelizmente, devido a restrições de tempo, não foi possível esquematizar o comportamento e a implementação do Sistema Mobile. No entanto, o aplicativo ProgressiveTutor terá uma seção inicial abordando os dados iniciais relacionados aos Sistemas Mobile.

6 AVALIAÇÃO DO PROTÓTIPO

Este capítulo descreve em detalhes os métodos utilizados para avaliar os protótipos desenvolvidos. A avaliação envolveu a análise da experiência de uso do ProgressiveTutor como ferramenta de apoio ao aprendizado das tecnologias abordadas, bem como a verificação dos requisitos estabelecidos no Capítulo 4.

Devido a restrições de tempo, não foi possível realizar testes com usuários voluntários. Portanto, optou-se por conduzir uma simulação por parte do autor deste trabalho, atuando como um usuário potencial da aplicação ProgressiveTutor desenvolvida ao longo deste projeto.

6.1 DEFINIÇÃO DOS TESTES

Levando em consideração os pontos levantados no capítulo 4 e na introdução, foram criadas as seguintes métricas de avaliação do ProgressiveTutor e dos aplicativos que ele ensina a fazer:

- Foi possível reduzir o número de abas abertas durante o processo de aprendizado para um número mínimo, mantendo apenas o ProgressiveTutor e, no máximo, duas outras abas (uma para o banco de dados e outra para algum recurso necessário em alguma etapa do tutorial)?
- Foram necessárias pesquisas adicionais para obter informações complementares que não estavam incluídas no tutorial, a fim de concluir a aplicação proposta na última parte de cada módulo do tutorial?
- O aplicativo do tipo Sistema Desktop ensinado pelo ProgressiveTutor oferece a funcionalidade de CRUD?
- O aplicativo do tipo Sistema Mobile ensinado pelo ProgressiveTutor oferece a funcionalidade de CRUD?
- O aplicativo do tipo Sistema Multiplataforma ensinado pelo ProgressiveTutor oferece a funcionalidade de CRUD?
- O aplicativo do tipo Sistema Desktop ensinado pelo ProgressiveTutor oferece a funcionalidade de PWA?
- O aplicativo do tipo Sistema Multiplataforma ensinado pelo ProgressiveTutor oferece a funcionalidade de PWA?

6.2 EXECUÇÃO DO TESTE

Para a realização dos testes, o autor fechou todos os recursos que utilizou para criar o tutorial do ProgressiveTutor e utilizou apenas o próprio tutorial para seguir cada etapa das trilhas até a criação dos aplicativos de cada tipo de sistema. Durante cada etapa, o autor analisou, tentando simular um possível usuário, e fez uma análise imparcial se cada etapa conseguiu transmitir corretamente o que se queria ensinar.

6.3 RESULTADOS DOS TESTES

Para a análise dos resultados obtidos, é importante ressaltar que eles foram elaborados pelo próprio autor, e, apesar de ter tentado manter a imparcialidade na análise, pode ter havido uma tendência a enfatizar os aspectos positivos dos resultados.

Ao analisar a eficácia do ProgressiveTutor em reduzir a quantidade de páginas visitadas e abertas simultaneamente, observou-se que o autor, durante a pesquisa e produção de cada parte do tutorial para cada sistema, costumava ter entre 25 a 40 abas abertas. Em contraste, ao utilizar apenas as informações do tutorial, foram abertas, no máximo, três abas ao mesmo tempo. Houve uma melhoria significativa nesse aspecto. Embora possa parecer um resultado óbvio, é comum ao buscar soluções na internet encontrar versões diferentes das tecnologias que o usuário está utilizando, o que leva o usuário a manter várias abas abertas até que o problema seja resolvido ou a funcionalidade seja implementada. A preocupação é fechar uma aba e não conseguir encontrar novamente a solução correta que, inicialmente, pode ter falhado, mas que poderia ser a solução adequada após a correção de outro aspecto.

Em relação à necessidade de recorrer a outras fontes de informação além do aplicativo, observou-se que não foi necessário, mas houve a necessidade de retornar a uma parte anterior do tutorial multiplataforma na configuração de variáveis de ambiente, devido à dificuldade encontrada nessa etapa de configuração da aplicação.

Quanto à capacidade de realizar operações CRUD nos aplicativos desenvolvidos, houve uma diferença significativa entre quando o autor os configurou como desenvolvedor e como usuário. Enquanto como desenvolvedor foi necessário estudar a documentação das ferramentas e realizar muitas tentativas e erros para obter os resultados desejados nas chamadas de API, como usuário, a "resposta correta"já estava disponível, resultando em um processo mais rápido. A análise revela que, como desenvolvedor, o processo foi mais trabalhoso, mas proporcionou um maior entendimento do funcionamento das ferramentas. Já como usuário, ao avançar no tutorial para ver a resposta, o resultado desejado foi obtido de maneira mais rápida, porém sem um aprendizado tão aprofundado quanto o do autor. Esta é uma consideração importante, e versões futuras do tutorial podem implementar formas de "travar"o progresso até que certas métricas sejam atingidas para medir o real esforço e aprendizado do usuário.

Essas métricas podem incluir o tempo gasto em cada etapa, bem como a submissão do código desenvolvido até o momento, seguida de uma análise deste código para avaliar o nível de compreensão do usuário em relação ao tutorial.

Em relação à capacidade de transformar tanto o ProgressiveTutor quanto o aplicativo desenvolvido para o Sistema Multiplataforma em PWAs, o processo foi desafiador como desenvolvedor devido a uma incompatibilidade inicial entre o código desenvolvido e as bibliotecas utilizadas para adicionar esta funcionalidade. Felizmente, ao longo dos testes, a ferramenta utilizada recebeu atualizações que corrigiram esse problema. Por outro lado, como usuário, ocorreu uma situação semelhante à mencionada anteriormente, em que o usuário não precisou fazer tanto esforço quanto o autor.

Em conclusão, o protótipo desenvolvido obteve resultados satisfatórios. No entanto, é perceptível que houve uma facilidade talvez excessiva em comparação com a implementação de um aplicativo sem o auxílio do protótipo em cada tipo de sistema. Portanto, sugere-se como trabalho futuro expandir cada um dos módulos existentes para permitir que o usuário tente desenvolver um aplicativo semelhante, mas com menos assistência, a fim de avaliar melhor o quanto o usuário realmente está aprendendo com o aplicativo, em comparação com simplesmente seguir as instruções fornecidas pelo mesmo.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi desenvolvido o protótipo de um aplicativo chamado ProgressiveTutor, que adota a abordagem proposta por (TILBURY; MESSNER, 1997). O aplicativo incorpora exemplos de código e seus respectivos resultados, apresentados em um formato de trilha. O usuário é guiado ao longo de cada etapa do tutorial, seguindo a trilha de aprendizado de um sistema ou área específica de interesse.

A principal ideia por trás do ProgressiveTutor é reunir e organizar conteúdos fragmentados disponíveis na internet em uma única ferramenta, facilitando o processo de aprendizado em áreas específicas de Tecnologia da Informação, como multiplataforma, desktop e mobile.

O protótipo do ProgressiveTutor tem como objetivo otimizar o tempo e a energia investidos no processo de aprendizado desses sistemas, fornecendo uma trilha que abrange desde os conceitos básicos até a etapa final, que engloba a implementação de front-end, back-end, API de comunicação entre eles e um sistema de autenticação. Para fins didáticos, foi selecionado um conjunto de tecnologias para cada tipo de sistema, a fim de fornecer exemplos práticos de implementação de todos esses requisitos. É importante ressaltar que o foco principal é ensinar os conceitos subjacentes, em vez das tecnologias específicas. Além disso, o ProgressiveTutor foi projetado de forma flexível, permitindo a adição de módulos adicionais ou a expansão dos módulos existentes.

7.1 LIÇÕES APRENDIDAS

No início do projeto, estabelecemos um escopo ambicioso para os protótipos, com alta complexidade e quantidade de módulos. Infelizmente, devido a restrições de tempo, não conseguimos concluir o terceiro módulo dedicado ao desenvolvimento de um aplicativo mobile, embora tenhamos realizado a parte teórica e definido as especificações.

Essa experiência nos ensinou a importância de gerenciar adequadamente o escopo de um projeto e estabelecer expectativas realistas em relação aos prazos. Agora reconhecemos que é crucial avaliar cuidadosamente os recursos disponíveis e alocar o tempo necessário para cada etapa do desenvolvimento.

Além disso, aprendemos a importância de priorizar as tarefas e adaptar o projeto quando necessário. Diante das restrições de tempo, foi essencial focar na conclusão dos módulos mais avançados, mesmo que isso significasse adiar o desenvolvimento do aplicativo mobile, cuja implementação foi designada para trabalhos futuros.

Essas lições serão valiosas para projetos futuros, ajudando-nos a evitar a sobrecarga de trabalho e garantir que nossas metas sejam alcançadas dentro dos prazos estabelecidos.

7.2 TRABALHOS FUTUROS

Como o ProgressiveTutor, o aplicativo central deste trabalho, é facilmente adaptável, existem várias possíveis opções para trabalhos futuros:

- Completar o módulo do Sistema Mobile.
- Adicionar etapas intermediárias adicionais nos tutoriais existentes para facilitar a curva de aprendizado.
- Adicionar etapas posteriores adicionais nos tutoriais existentes para aprofundar os conteúdos já abordados.
- Adicionar mais um módulo com uma tecnologia mais compatível/fácil para implementar a funcionalidade de um PWA.
- Adicionar mais um módulo que não esteja listado acima.

REFERÊNCIAS

AASE, M.; KURFESS, F. Utilizing learning styles for interactive tutorials. *In:* IEEE International Conference on Advanced Learning Technologies, 2004. Proceedings. [*S.l.*: *s.n.*], 2004. P. 828–830. DOI: 10.1109/ICALT.2004.1357674. Citado nas pp. 32, 33.

AMAZON. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://aws.amazon.com/what-is/api/>. Citado na p. 25.

APRIORIT. **Como mandar notificações push com aplicações web**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://www.apriorit.com/dev-blog/740-web-how-to-add-push-notifications-to-your-web-application-using-google-firebase>. Citado na p. 24.

AWS-AMAZON. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://aws.amazon.com/pt/what-is/web-application/>. Citado na p. 22.

BETRYBE. **Aplicações Web**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/>. Citado nas pp. 22, 23.

CNBC. **Custo de publicação de aplicativos na Play Store**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://www.cnbc.com/2021/03/16/google-app-store-fees-cut-for-developers-on-first-million>. Citado na p. 25.

GOOGLE. **Mobilefirst**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://developers.google.com/search/mobile-sites/mobile-first-indexing>. Citado na p. 16.

GOOGLE. **Otimização do motor de busca**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://developers.google.com/search/docs/fundamentals/seo-starter-guide>. Citado na p. 24.

GOOGLE. **Publicação de aplciativo na Play Store**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://support.google.com/googleplay/android-developer/answer/9859751?hl=en>. Citado na p. 25.

HULLS, C.C.W. *et al.* Interactive online tutorial assistance for a first programming course. **IEEE Transactions on Education**, v. 48, n. 4, p. 719–728, 2005. DOI: 10.1109/TE.2005.858400. Citado na p. 33.

IBM. **Criar aplicações offline**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://developer.ibm.com/tutorials/x-html5mobile3/>. Citado na p. 23.

JSON2CSHARP. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://json2csharp.com/>. Citado na p. 51.

KITCHENHAM, Barbara. Procedures for Performing Systematic Reviews. **Keele, UK, Keele Univ.**, v. 33, jul. 2004. Citado na p. 31.

NEXT-AUTH. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://next-auth.js.org/getting-started/introduction>. Citado na p. 27.

NEXTJS. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs>. Citado na p. 28.

OKTA. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://jwt.io/introduction/>. Citado na p. 27.

PRISMA. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://www.prisma.io/>. Citado na p. 29.

RAILWAY. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://railway.app/>. Citado na p. 29.

REDHAT. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>. Citado na p. 26.

SEYAM, Mohammed *et al.* Teaching mobile application development through lectures, interactive tutorials, and Pair Programming. *In:* 2016 IEEE Frontiers in Education Conference (FIE). [*S.l.*: *s.n.*], 2016. P. 1–9. DOI: 10.1109/FIE.2016.7757533. Citado nas pp. 33, 34.

STATCOUNTER. **Proporção de usuários 2012 a 2022**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-200901-202211>. Citado nas pp. 16, 17.

STRAPI. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://www.adservio.fr/post/introduction-to-strapi>. Citado na p. 29.

STRAPI. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://docs.strapi.io/dev-docs/api/rest/interactive-query-builder>. Citado na p. 51.

T3. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://create.t3.gg/en/introduction>. Citado nas pp. 30, 40.

TILBURY, D.; MESSNER, W. Development and integration of Web-based software tutorials for an undergraduate curriculum: control tutorials for MATLAB. *In:* PROCEEDINGS Frontiers in Education 1997 27th Annual Conference. Teaching and Learning in an Era of Change. [*S.l.*: *s.n.*], 1997. 1070–1075 vol.2. DOI: 10.1109/FIE.1997.636040. Citado nas pp. 31, 60.

TRPC. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://trpc.io/docs>. Citado na p. 26.

TYPESCRIPT. **TypeScript**. [*S.l.*: *s.n.*], jul. 2023. Disponível em: <https://www.typescriptlang.org/>. Citado na p. 27.

VERCEL. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://vercel.com/blog/what-is-vercel>. Citado na p. 28.

WEB-DEV. [*S.l.*: *s.n.*], jun. 2023. Disponível em: <https://web.dev/what-are-pwas/>. Citado na p. 23.

Apêndices

APÊNDICE A – CÓDIGO FONTE

ProgressiveTutor Disponível em: https://github.com/ArturRA/progressive_tutor_-tutorial

Sistema Multiplataforma Disponível em: https://github.com/ArturRA/progressive_-tutor_multiplataform

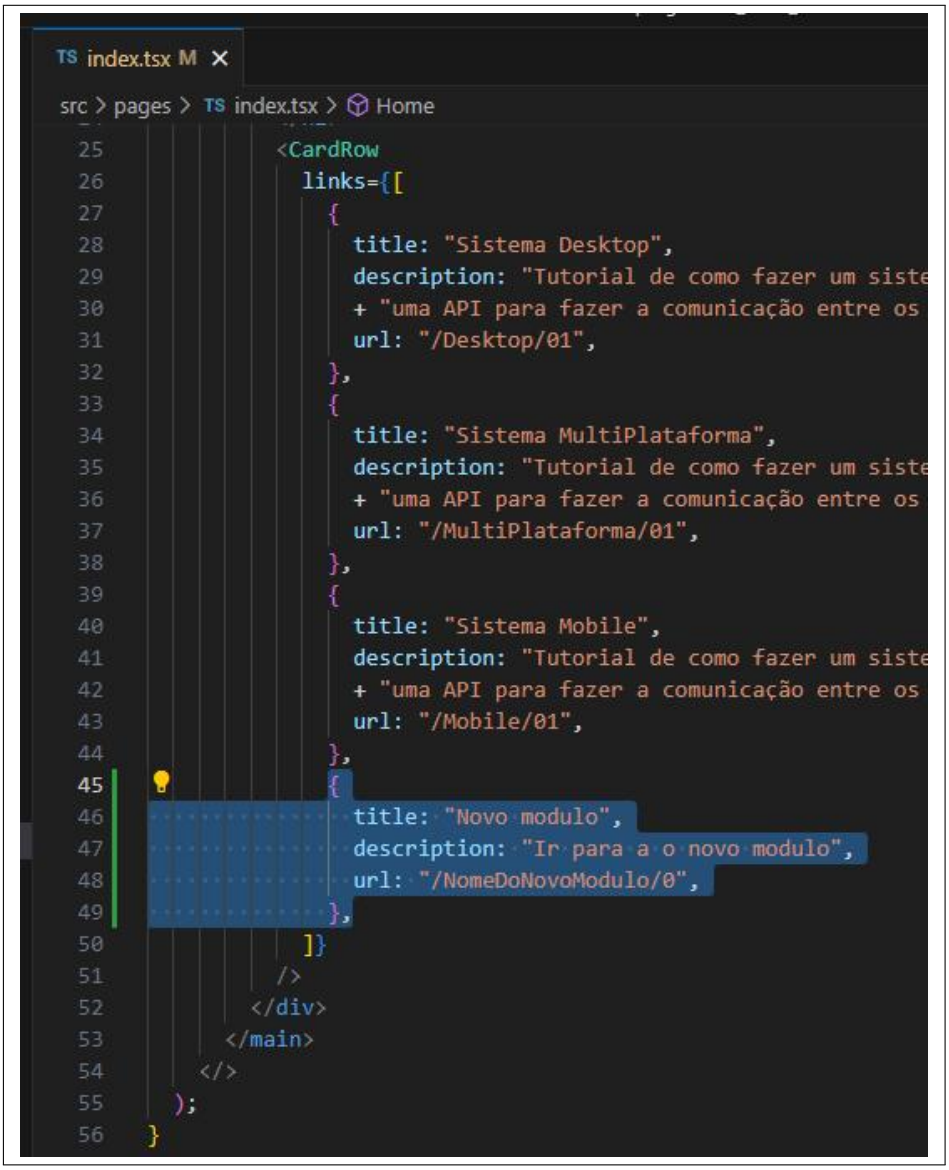
Sistema Desktop Disponível em: https://github.com/ArturRA/progressive_tutor_-desktop

B EXPANDINDO O PROGRESSIVETUTOR

O ProgressiveTutor foi idealizado desde o início com o objetivo de ser modular, permitindo a expansão dos módulos disponíveis de forma fácil.

O primeiro passo é adicionar um novo botão ao índice que levará ao novo módulo, conforme demonstrado na Figura 18.

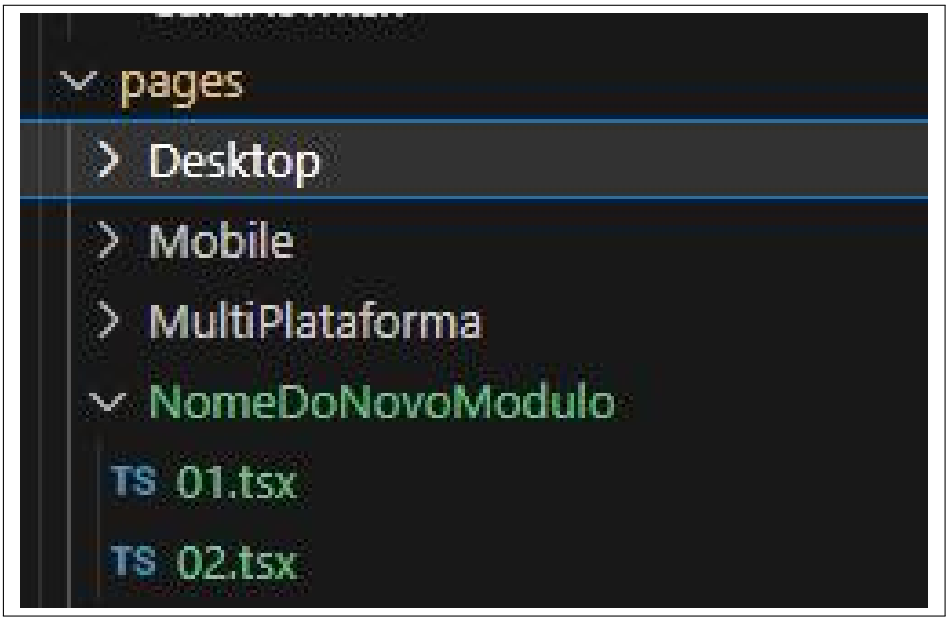
Figura 18 – Expansão de módulos etapa 1



Fonte: o autor

Em seguida, deve-se criar uma nova pasta para o novo módulo dentro da pasta chamada "pages"e fazer uma cópia do arquivo "01.tsx"de algum outro módulo para servir como base, como demonstrado na Figura 19.

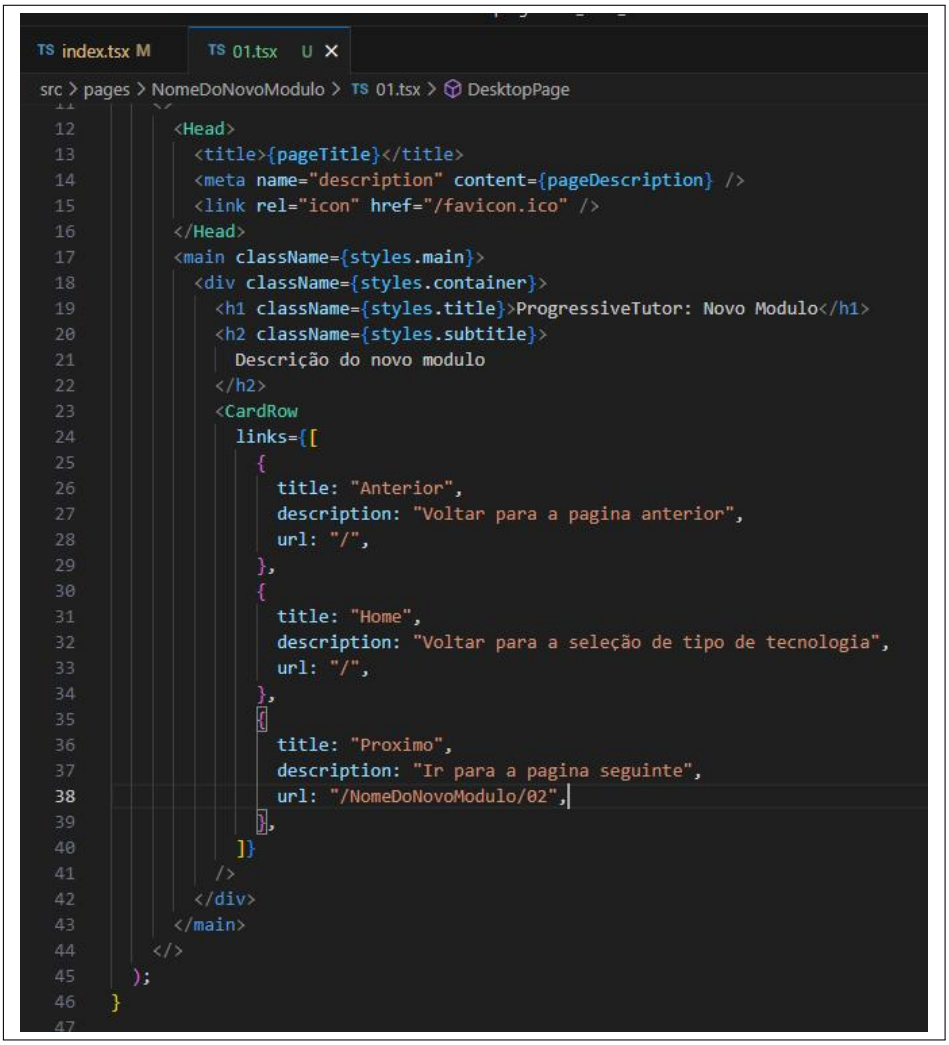
Figura 19 – Expansão de módulos etapa 2



Fonte: o autor

Por fim, é necessário editar o título, a descrição e os links para a página anterior e para a página seguinte, como demonstrado na Figura 20. Esse processo deve ser repetido para cada etapa desejada no tutorial do novo módulo.

Figura 20 – Expansão de módulos etapa 3



Fonte: o autor

C ARTIGO SBC

Aplicativo multiplataforma ”ProgressiveTutor” utilizando Progressive Web App para auxílio de estudos na área mobile, desktop e multiplataforma

Artur R. Alfa[1]

[1]1Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

artur.alfa@grad.ufsc.br

Abstract. *The objective of this work is to develop ProgressiveTutor, a cross-platform, modular, and scalable tutorial engine using Next.js. This engine will serve as an initial learning guide in different areas, with a specific focus on multi-platform, desktop, and mobile development for Information Technology students. ProgressiveTutor will be a Progressive Web App (PWA), providing introductory knowledge on these three areas in one place. A functional prototype of ProgressiveTutor has been developed, consisting of three modules covering each mentioned area. Each module demonstrates the app creation process, including front-end, back-end, communication API, and authentication functionalities. Furthermore, a practical evaluation of the usage of ProgressiveTutor and the three tutorials/modules was conducted.*

Resumo. *O objetivo deste trabalho é desenvolver o ProgressiveTutor, uma engine de tutoriais multiplataforma, modular e expansível, utilizando Next.js. Essa engine atuará como um guia de aprendizagem inicial em diferentes áreas, com foco em desenvolvimento multiplataforma, desktop e mobile, para estudantes de Tecnologia da Informação. O ProgressiveTutor será um Progressive Web App (PWA), oferecendo conhecimentos iniciais sobre as três áreas em um único lugar. O protótipo funcional do ProgressiveTutor foi desenvolvido, com três módulos abordando cada área mencionada. Cada módulo demonstra o processo de criação de um aplicativo, incluindo front-end, back-end, API de comunicação e autenticação. Além disso, foi realizada uma avaliação prática do uso do ProgressiveTutor e dos três tutoriais/módulos criados.*

1. Introdução

As aplicações móveis têm crescido de forma expressiva, expandindo sua presença e criando novos mercados, proporcionando maior acessibilidade aos usuários e disponibilizando conteúdo acessível a qualquer hora. Esse crescimento é de grande magnitude, o que tem levado as empresas a repensarem sua preferência em relação à plataforma a ser priorizada, evidenciado pela popularização do termo 'Mobile First' (GOOGLE, 2023a) recentemente, bem como pela mudança no tipo de indexação praticada pela Google.

O conceito de 'Mobile First' se refere a práticas de projeto e desenvolvimento que têm como foco principal os usuários móveis. Mais especificamente, significa priorizar a experiência online dos usuários que utilizam celulares, tablets e dispositivos semelhantes antes daqueles que acessam o conteúdo por meio de desktops ou outros meios. Esta abordagem inverte o fluxo de desenvolvimento anterior, no qual se desenvolvia inicialmente para desktop e posteriormente simplificava o design para dispositivos móveis.

Atualmente, os usuários desejam ter acesso ao conteúdo digital que consomem em qualquer dispositivo com acesso à internet, seja em um telefone, tablet, computador, entre outros. Este fato faz com que as empresas tenham que disponibilizar seu conteúdo para qualquer dispositivo, independentemente da plataforma. Entretanto, este processo pode gerar altos custos caso a empresa tente desenvolver aplicativos nativos para cada tipo de plataforma, uma vez que provavelmente necessitará de uma equipe específica para cada uma delas, considerando que o desenvolvimento para Android, iOS e desktop requer diferentes conhecimentos e códigos específicos. Por esse motivo, as empresas têm buscado alternativas mais econômicas que ofereçam um produto final de boa qualidade, sendo uma delas a tecnologia PWA.

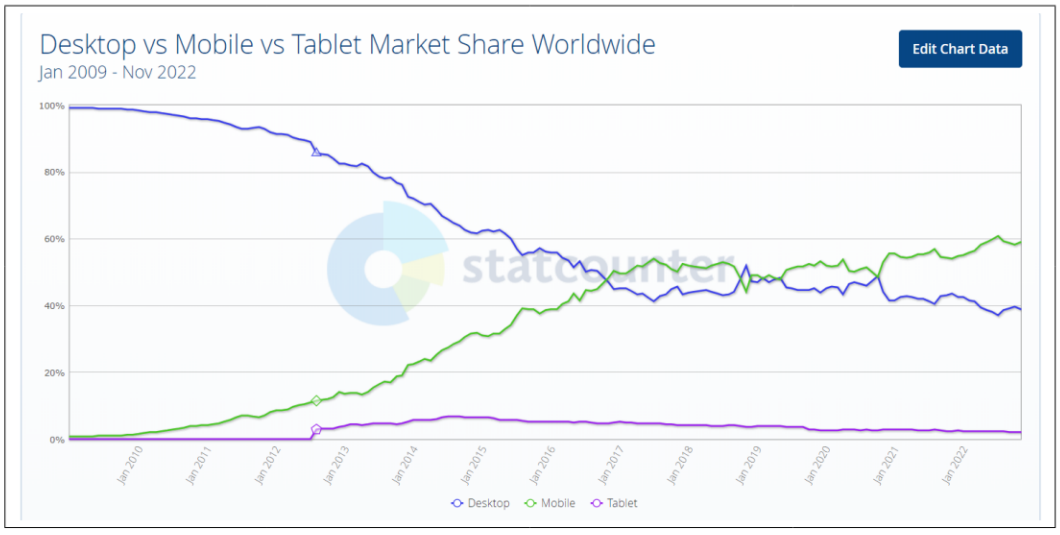
Diante deste cenário, este trabalho tem como objetivo desenvolver o Progressive-Tutor, uma engine de tutoriais multiplataforma, modular e expansível, utilizando Next.js. Inspirado pela necessidade de uma abordagem unificada de aprendizado, o Progressive-Tutor atuará como um guia abrangente para a aprendizagem inicial em diferentes áreas de conhecimento, com foco específico no desenvolvimento multiplataforma, desktop e mobile, direcionado aos alunos da área de Tecnologia da Informação.

2. Problemática

Há dez anos (STATCOUNTER, 2023), o mercado era dominado primariamente pelos desktops. As empresas desenvolviam seus produtos e serviços primeiro para desktops, sendo a versão para celular geralmente uma versão simplificada daquela criada para desktops.

Esta proporção mudou drasticamente ao longo dos anos, chegando ao estado atual em que os dispositivos móveis correspondem a mais de 50% do mercado. Este desenvolvimento impactante deu origem à mentalidade Mobile First.

Figure 1. Proporção de usuários desktop vs mobile vs tablet 2009 a 2022 (STATCOUNTER, 2023)



Diante desta mudança na distribuição do mercado, as empresas, ao criar um produto ou serviço atualmente, têm, em geral, as seguintes opções ao considerar o público-alvo a ser atendido: escolher uma única plataforma (desktop, Android, iOS, web) e desenvolver de maneira nativa, abrindo mão das outras plataformas; escolher mais de uma plataforma e desenvolver de forma nativa para todas elas; ou, por último, fazer uso de uma tecnologia multiplataforma para atingir diferentes tipos de plataformas.

Cada uma destas opções possui seus prós e contras:

- 1) Escolher apenas uma plataforma e desenvolver nativamente limita o público do produto ou serviço, mas é mais econômico e simples de controlar;
- 2) Escolher mais de uma plataforma e desenvolver nativamente para todas elas alcança um público maior, mas tem um custo e complexidade maiores, além de geralmente exigir mais funcionários para desenvolver e manter o produto ou serviço;
- 3) Caso a empresa opte por fazer uso de uma tecnologia multiplataforma, ela poderá atingir um público maior em comparação com a primeira opção e economizar dinheiro, mas terá que lidar com as dificuldades e complicações que geralmente estão associadas a tecnologias multiplataforma.

3. Motivação

Com tantas opções diferentes de tecnologias disponíveis atualmente, acaba sendo complicado para os alunos da área de Tecnologia da Informação adquirirem conhecimento sobre todas elas sem ter que gastar muitas horas em pesquisas e acessar vários sites diferentes para reunir todas as informações necessárias para desenvolver um conhecimento, pelo menos básico, sobre essas tecnologias. Por esse motivo, este projeto tem como objetivo facilitar este processo, reunindo em um só lugar todo o conhecimento necessário para aprender as etapas iniciais, possibilitando que o aluno seja capaz de desenvolver um aplicativo simples utilizando apenas o ProgressiveTutor.

4. Objetivo Geral

O presente trabalho tem como objetivo desenvolver o ProgressiveTutor, uma engine de tutoriais multiplataforma, modular e expansível, utilizando Next.js, a fim de facilitar o processo de aprendizado nas áreas de desenvolvimento multiplataforma, desktop e mobile, direcionado para alunos da área de Tecnologia da Informação.

O ProgressiveTutor busca reunir o conteúdo disperso existente na internet em um único aplicativo, proporcionando aos estudantes um ambiente centralizado para suas necessidades de aprendizado. O objetivo é oferecer tutoriais abrangentes, que abordem desde os conceitos básicos até o desenvolvimento completo de um aplicativo, incluindo front-end, back-end, uma API de comunicação entre eles e funcionalidades de autenticação.

Cada módulo do ProgressiveTutor será estruturado de forma sequencial, iniciando com a introdução e compreensão dos conceitos fundamentais de cada área de conhecimento. Em seguida, os alunos serão guiados passo a passo na criação de um aplicativo, aplicando os conhecimentos aprendidos anteriormente. Dessa forma, os alunos poderão consolidar os conceitos e habilidades adquiridos, além de obter uma compreensão prática das tecnologias envolvidas.

O ProgressiveTutor será desenvolvido com uma abordagem modular, permitindo a expansão futura de módulos adicionais para outras áreas de conhecimento ou o aprofundamento dos módulos existentes. Isso possibilitará uma experiência personalizada de aprendizado, em que os alunos poderão escolher os módulos relevantes para suas necessidades e interesses específicos.

Ao final do desenvolvimento do ProgressiveTutor, será criado um protótipo funcional, composto por três módulos representativos das áreas de desenvolvimento multiplataforma, desktop e mobile. Cada módulo apresentará um tutorial completo, abrangendo desde os conceitos básicos até a implementação de um aplicativo com todas as funcionalidades mencionadas. Além disso, será realizada uma avaliação da utilização do ProgressiveTutor e dos três módulos criados na prática, visando analisar a eficácia do aplicativo como guia de aprendizado.

5. Desenvolvimento

Este capítulo detalha o desenvolvimento de todo o projeto. Primeiramente, serão abordados os detalhes do ProgressiveTutor, incluindo seus comportamentos e implementação. Em seguida, será apresentado o desenvolvimento de cada sistema individual desenvolvido com o auxílio do ProgressiveTutor, abordando suas características e particularidades.

6. ProgressiveTutor

O ProgressiveTutor é a parte central proposta por este trabalho, com o objetivo de facilitar o aprendizado de diferentes tecnologias (multiplataforma, desktop e mobile), conforme discutido na seção de introdução. Ele é voltado para estudantes de Ciência da Computação que já possuem uma base em programação, redes e segurança digital, adquirida em disciplinas obrigatórias ou optativas, e que desejam aprofundar seus conhecimentos por meio da criação dos três aplicativos ensinados pelo ProgressiveTutor.

6.1. Comportamento do ProgressiveTutor

O ProgressiveTutor é um aplicativo tutorial desenvolvido em Next.js. Seu fluxo de funcionamento básico pode ser compreendido através das Figuras 2 e 3, onde o usuário inicia o aplicativo na página inicial, que exibe três botões correspondentes às três tecnologias ensinadas. Ao clicar em uma das opções, o usuário é redirecionado para a página específica daquela tecnologia, onde serão apresentados os passos do tutorial na forma de texto. Por exemplo, no caso do sistema multiplataforma, os primeiros passos incluem o download do Next.js, a introdução ao JavaScript, a familiarização com o Next.js, conceitos básicos sobre programação multiplataforma e os passos para a criação do aplicativo. Além do texto explicativo, os botões na página permitem que o usuário volte para a página anterior, retorne à tela inicial ou avance para a próxima etapa do tutorial.

Figure 2. Tela Inicial do ProgressiveTutor

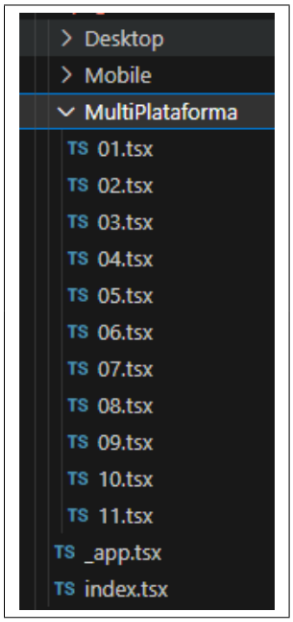
Figure 3. Tela do ProgressiveTutor para Aplicação Multiplataforma

6.2. Implementação do ProgressiveTutor

A implementação do ProgressiveTutor começou utilizando o recurso de criação rápida ”create t3-app”. A partir daí, foram desenvolvidas a página inicial e as implementações dos sistemas web e desktop. Durante esse processo, foram buscadas informações sobre a implementação do sistema mobile. Após a conclusão dos dois primeiros sistemas, o foco foi direcionado para a implementação dos tutoriais correspondentes.

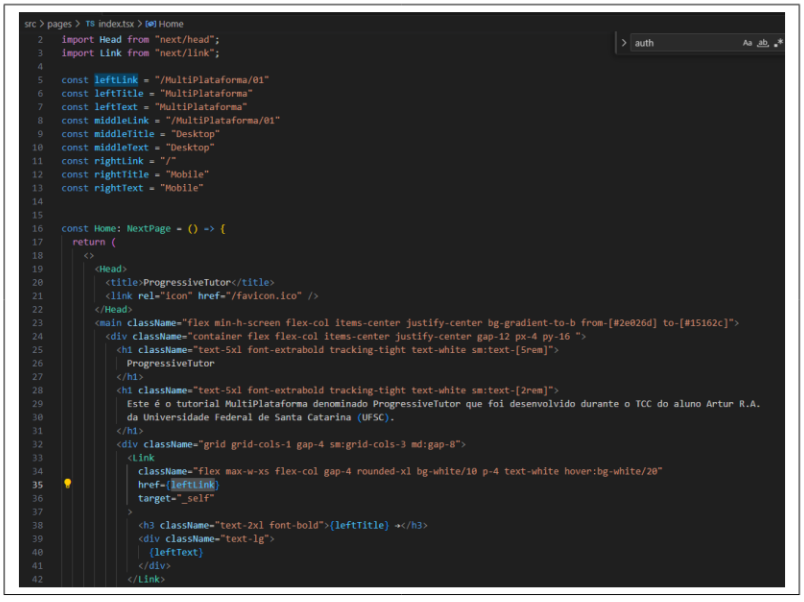
A estrutura do tutorial permite uma fácil expansão, caso seja desejado adicionar mais etapas entre as existentes para suavizar a transição ou adicionar etapas adicionais no final para aprofundar ainda mais o conteúdo. A estrutura dos arquivos do tutorial pode ser observada na Figura 4, onde todas as páginas relacionadas a um sistema são agrupadas em suas próprias pastas.

Figure 4. Organização das Páginas do ProgressiveTutor



A transição entre as páginas do tutorial é realizada por meio de links para outras páginas do aplicativo. Ao pressionar o botão correspondente, a transição é ativada, como ilustrado na Figura 5.

Figure 5. Mecanismo de Mudança de Página no ProgressiveTutor



```
src > pages > TS index.tsx > [@] Home
import Head from "next/head";
import Link from "next/link";

const leftLink = "/MultiPlataforma/01"
const leftTitle = "MultiPlataforma"
const leftText = "MultiPlataforma"
const middleLink = "/MultiPlataforma/01"
const middleTitle = "Desktop"
const middleText = "Desktop"
const rightLink = "/"
const rightTitle = "Mobile"
const rightText = "Mobile"


const Home: NextPage = () => {
  return (
    <>
      <Head>
        <title>ProgressiveTutor</title>
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <main className="flex min-h-screen flex-col items-center justify-center bg-gradient-to-b from-[#2e026d] to-[#15162c]">
        <div className="container flex flex-col items-center justify-center gap-12 px-4 py-16 ">
          <h1 className="text-5xl font-extrabold tracking-tight text-white sm:text-[5rem]">
            ProgressiveTutor
          </h1>
          <h1 className="text-5xl font-extrabold tracking-tight text-white sm:text-[2rem]">
            Este é o tutorial MultiPlataforma denominado ProgressiveTutor que foi desenvolvido durante o TCC do aluno Artur R.A.
            da Universidade Federal de Santa Catarina (UFSC).
          </h1>
          <div className="grid grid-cols-1 gap-4 sm:grid-cols-3 md:gap-8">
            <Link
              className="flex max-w-xs flex-col gap-4 rounded-xl bg-white/10 p-4 text-white hover:bg-white/20"
              href={leftLink}
              target="_self"
            >
              <h3 className="text-2xl font-bold">{leftTitle} →</h3>
              <div className="text-lg">
                {leftText}
              </div>
            </Link>
```

7. Sistema Multiplataforma

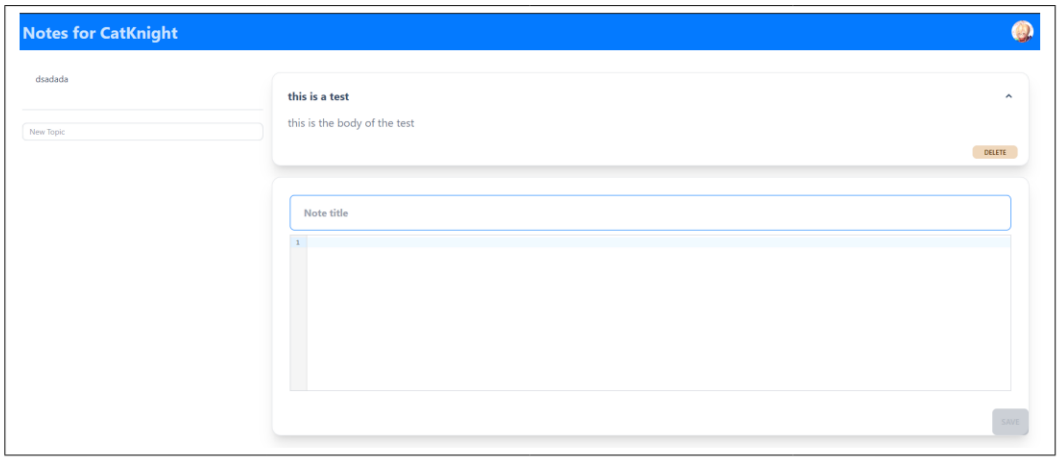
O Sistema Multiplataforma é um dos três tipos de sistemas ensinados pelo ProgressiveTutor. Dentre os três, é o mais distante do conteúdo abordado no curso de Ciência da Computação, pois requer conhecimento intermediário a avançado em JavaScript e React. Assim como nos outros dois sistemas, ao final do tutorial da área de desenvolvimento multiplataforma, o ProgressiveTutor ensina como criar um aplicativo simples para fazer anotações, permitindo que o usuário registre temas e suas respectivas notas. Esse aplicativo possui funcionalidades como um banco de dados online utilizando o Supabase, uma interface de usuário declarativa utilizando os componentes disponíveis no Next.js, uma forma de comunicação entre o frontend e o backend utilizando o tRPC e o NextAuth.js para autenticação.

7.1. Comportamento do Sistema Multiplataforma

O aplicativo desenvolvido com o ProgressiveTutor, do tipo multiplataforma, utiliza o Next.js como framework principal. A Figura 6 mostra a tela principal do aplicativo. O processo de login é realizado clicando no ícone localizado no canto superior direito (no exemplo da figura, o ícone corresponde à conta GitHub do usuário logado), que redireciona o usuário para a página de login controlada pelo NextAuth.js.

Após fazer o login, o usuário pode criar um novo tópico clicando no campo de texto com a indicação ”Novo Tópico” e inserindo o nome desejado para o tópico. Ao pressionar Enter, é possível criar uma nova nota relacionada a esse tópico. Para isso, o usuário precisa selecionar o tópico desejado, preencher o título e o conteúdo da nota e, em seguida, clicar no botão ”Salvar”. É importante mencionar que todas as operações de criação, edição e exclusão utilizam o tRPC para se comunicar com o banco de dados PostgreSQL hospedado no Supabase.

Figure 6. Tela do Aplicativo Multiplataforma



7.2. Implementação do Sistema Multiplataforma

A implementação do Sistema Multiplataforma foi iniciada utilizando o recurso de criação rápida fornecido pelo create t3-app. No entanto, ao contrário do ProgressiveTutor, todas

as opções foram utilizadas, incluindo tRPC, NextAuth.js, Prisma e Tailwind. A partir daí, foram identificadas as partes da tela que se repetiriam e foram criados componentes para representá-las. Um exemplo de código de um componente pode ser visualizado na Figura 7, onde a aparência e o comportamento do componente foram declarados de forma declarativa. Além disso, é importante mencionar que a aplicação utiliza o padrão de design chamado State, que permite que um objeto altere seu comportamento quando seu estado interno muda. Isso pode ser observado, por exemplo, no botão de salvar da nota, que é desabilitado quando pelo menos um dos campos (título da nota ou corpo da nota) está vazio. O design da página inicial foi concebido de forma a evitar a necessidade de criar diálogos para interagir com a aplicação.

Figure 7. Componente NoteCard

```tsx
import { useState } from "react";

import ReactMarkdown from "react-markdown";

import { type RouterOutputs } from "../utils/api";

type Note = RouterOutputs["note"]["getAll"][0];

export const NoteCard = ({
  note,
  onDelete,
}: {
  note: Note;
  onDelete: () => void;
}) => {
  const [isExpanded, setIsExpanded] = useState<boolean>(true);

  return (
    <div className="card mt-5 border border-gray-200 bg-base-100 shadow-xl">
      <div className="card-body m-0 p-3">
        <div
          className={`collapse-arrow ${
            isExpanded ? "collapse-open" : ""
          } collapse`}
          onClick={() => setIsExpanded(!isExpanded)}
        >
          <div className="collapse-title text-xl font-bold">{note.title}</div>
          <div className="collapse-content">
            <article className="prose lg:prose-xl">
              <ReactMarkdown>{note.content}</ReactMarkdown>
            </article>
          </div>
        </div>
        <div className="card-actions mx-2 flex justify-end">
          <button className="btn-warning btn-xs btn px-5" onClick={onDelete}>
            Delete
          </button>
        </div>
      </div>
    </div>
  );
};
```

8. Sistema Desktop

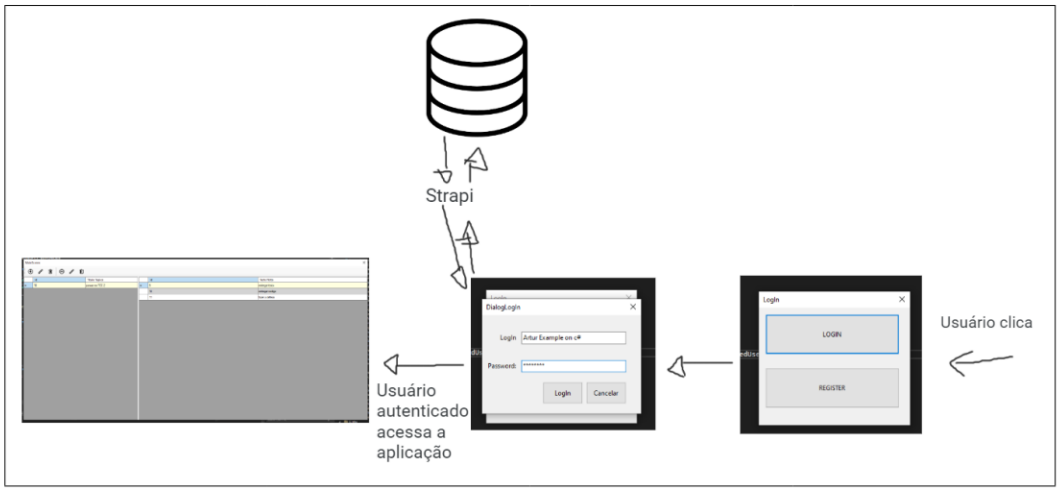
O Sistema Desktop é um dos três tipos de sistemas ensinados pelo ProgressiveTutor. Entre os três, é o mais próximo do que é abordado no curso de Ciências da Computação. Assim

como nos outros dois sistemas, ao final do tutorial da área de desenvolvimento desktop, o ProgressiveTutor ensina como criar um aplicativo simples para fazer anotações de temas e notas, que possui funcionalidades como um banco de dados online, uma interface estilo Windows Form feita no Visual Studio, uma forma de comunicação entre ambos utilizando o Strapi com chamadas REST, e uma estrutura de arquivos organizada seguindo o padrão MVC, com a adição da entidade Repositorio.

8.1. Comportamento do Sistema Desktop

O aplicativo desenvolvido com o ProgressiveTutor do tipo Desktop é uma aplicação em C# com um fluxo básico exemplificado na Figura 8, que ilustra o processo de login de um usuário. Nesse processo, o usuário escolhe entre fazer login ou criar uma conta, e em seguida entra no sistema com essa nova conta. Após essa escolha, as informações necessárias do diálogo são enviadas para o banco de dados por meio do Strapi, que retorna a resposta.

Figure 8. Diagrama do Funcionamento do Login no Sistema Desktop



8.2. Implementação do Sistema Desktop

A implementação do Sistema Desktop começa com a configuração dos sistemas no site Railway. Nesse processo, é configurado o banco de dados PostgreSQL e estabelecida a conexão com o repositório ”strapitcc” no GitHub, que contém o código que descreve o comportamento do Strapi. Em seguida, o Strapi é inicializado no modo de desenvolvedor para utilizar seu sistema de controle de entidades interativo e configurar as entidades do banco de dados, bem como as relações entre elas. Feito isso, é feita uma esquematização das partes da aplicação, optando-se por utilizar a arquitetura MVC para melhor separar as responsabilidades. Além disso, é adicionada uma entidade adicional chamada Repositorio. A aplicação é dividida em três partes principais: Apresentação, Domínio e Infraestrutura. Na parte de Apresentação, responsável pelo View e Controller, são desenvolvidos os principais elementos de UI. A parte de Domínio, responsável pelos Models (denominados Entity nesta aplicação), é criada como uma biblioteca para ser usada pelas outras partes da

aplicação. A parte de Infraestrutura é responsável pelos Repositórios e também é criada como uma biblioteca. Como exemplo, será demonstrada cada parte relacionada à classe modelo chamada EntityAuthenticatedUser.

Figure 9. Estrutura do Projeto - Parte 1

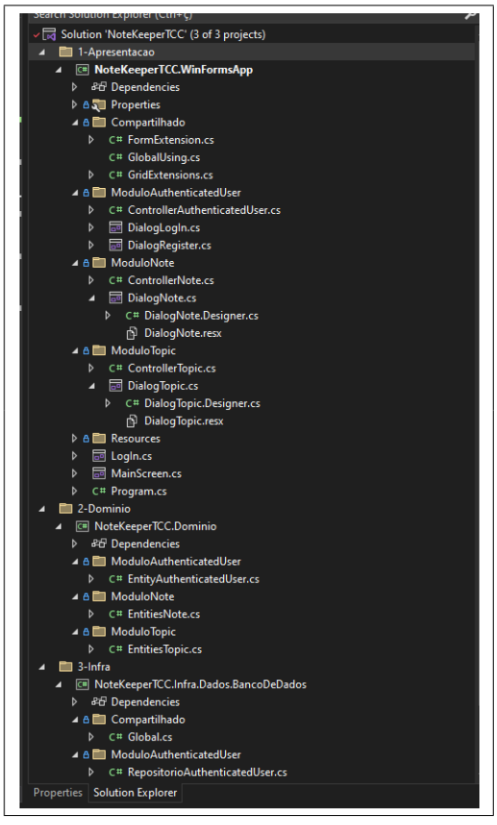
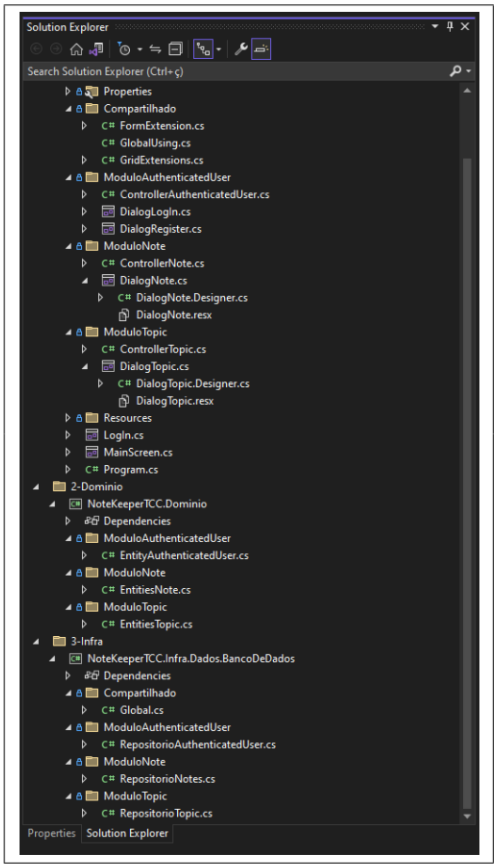


Figure 10. Estrutura do Projeto - Parte 2



8.2.1. Modelo

A seção do modelo, com a adição da entidade Repositorio, é responsável pela estruturação da entidade e alterações. Nesta aplicação específica, ela é utilizada como molde para fazer a desserialização da entidade retornada pelo banco de dados quando um pedido REST é realizado.

8.2.2. Modelo Usuário Autenticado

A classe EntityAuthenticatedUser é estruturada da seguinte forma, conforme ilustrado na Figura 11. Essa estrutura, assim como a estrutura das outras entidades, foi obtida fazendo uma chamada ao endpoint correspondente a cada entidade. No caso específico dessa classe, o endpoint utilizado foi ”api/auth/local”. Durante essa chamada, foram aplicadas customizações para filtrar as informações desnecessárias, utilizando a documentação do Strapi (STRAPI, 2023b). Em seguida, o retorno foi utilizado em conjunto com o site (JSON2CSHARP, 2023) para obter uma base da estrutura de cada entidade. Para o processo da classe EntityAuthenticatedUser, é necessário que ela possua um JWT e um User

com um ID. Essa estrutura é essencial para realizar a desserialização da resposta de uma solicitação de criação ou login de um usuário existente.

Figure 11. Entidade Usuário Autenticado

```
namespace NoteKeeperTCC.Dominio.ModuloAuthenticatedUser
{
    24 references
    public class EntityAuthenticatedUser
    {
        4 references
        public class User
        {
            2 references
            public int id { get; set; }

            0 references
            public User()
            {
            }

            0 references
            public User(int id)
            {
                this.id=id;
            }
        }

        8 references
        public string jwt { get; set; }
        2 references
        public User user { get; set; }

        0 references
        public EntityAuthenticatedUser()
        {
        }

        0 references
        public EntityAuthenticatedUser(string jwt, User user)
        {
            this.jwt=jwt;
            this.user=user;
        }
    }
}
```

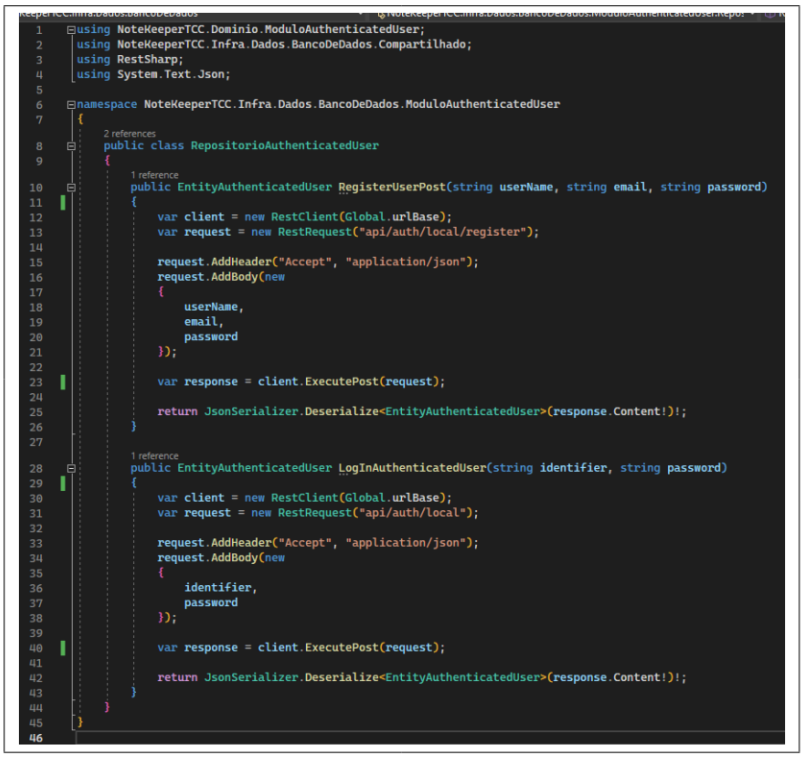
8.2.3. Repositório

Como explicado anteriormente, para cada entidade existe um repositório correspondente, que contém a lógica da entidade, além das funcionalidades de alterações de dados no banco de dados. O repositório é responsável pela serialização e desserialização da entidade correspondente ao fazer chamadas para o banco de dados utilizando o Strapi.

8.2.4. Repositório Usuário Autenticado

É possível ver na Figura 12 a estrutura da classe e como ocorrem as chamadas de login e registro de novos usuários.

Figure 12. Entidade Repositório Usuário Autenticado



```
using NoteKeeperTCC.Dominio.ModuloAuthenticatedUser;
using NoteKeeperTCC.Infra.Dados.BancoDeDados.Compartilhado;
using RestSharp;
using System.Text.Json;

namespace NoteKeeperTCC.Infra.Dados.BancoDeDados.ModuloAuthenticatedUser
{
    2 references
    public class RepositorioAuthenticatedUser
    {
        1 reference
        public EntityAuthenticatedUser RegisterUserPost(string userName, string email, string password)
        {
            var client = new RestClient(Global.urlBase);
            var request = new RestRequest("api/auth/local/register");

            request.AddHeader("Accept", "application/json");
            request.AddBody(new
            {
                userName,
                email,
                password
            });

            var response = client.ExecutePost(request);

            return JsonSerializer.Deserialize<EntityAuthenticatedUser>(response.Content!)!;
        }

        1 reference
        public EntityAuthenticatedUser LogInAuthenticatedUser(string identifier, string password)
        {
            var client = new RestClient(Global.urlBase);
            var request = new RestRequest("api/auth/local");

            request.AddHeader("Accept", "application/json");
            request.AddBody(new
            {
                identifier,
                password
            });

            var response = client.ExecutePost(request);

            return JsonSerializer.Deserialize<EntityAuthenticatedUser>(response.Content!)!;
        }
    }
}
```

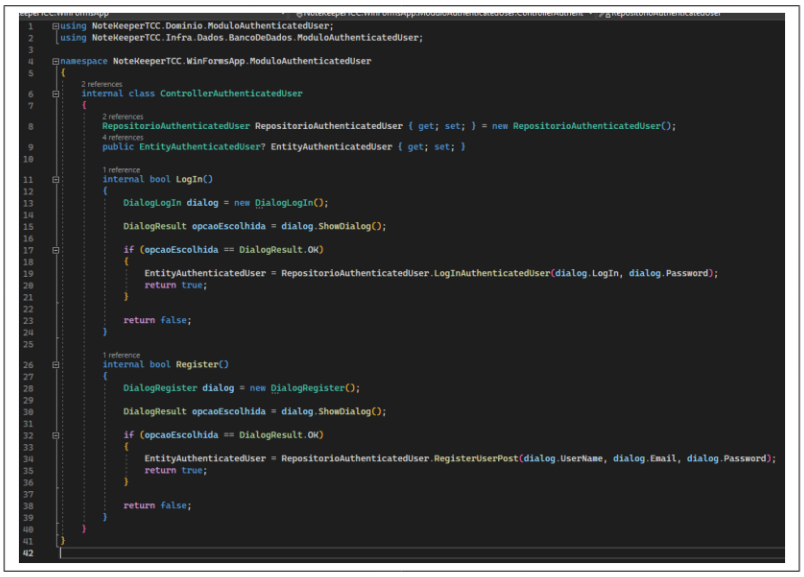
8.2.5. Controller

Assim como o repositório, para cada entidade existe um controlador correspondente, que contém os controles mais gerais. Ele se comunica com o repositório correspondente para obter o retorno da comunicação com o banco de dados e repassá-lo à visão.

8.2.6. Controller Usuário Autenticado

É possível ver na Figura 13 a estrutura da classe e como ocorrem as chamadas de login e registro de novos usuários, que chamam os métodos correspondentes do repositório.

Figure 13. Entidade Controller Usuário Autenticado



```
using NoteKeeperTCC.Dominio.ModuloAuthenticatedUser;
using NoteKeeperTCC.Infra.Dados.BancoDeDados.ModuloAuthenticatedUser;

namespace NoteKeeperTCC.WinFormsApp.ModuloAuthenticatedUser
{
    2 references
    internal class ControllerAuthenticatedUser
    {
        2 references
        RepositorioAuthenticatedUser RepositorioAuthenticatedUser { get; set; } = new RepositorioAuthenticatedUser();
        4 references
        public EntityAuthenticatedUser? EntityAuthenticatedUser { get; set; }

        1 reference
        internal bool LogIn()
        {
            DialogLogIn dialog = new DialogLogIn();

            DialogResult opcaoEscolhida = dialog.ShowDialog();

            if (opcaoEscolhida == DialogResult.OK)
            {
                EntityAuthenticatedUser = RepositorioAuthenticatedUser.LogInAuthenticatedUser(dialog.LogIn, dialog.Password);
                return true;
            }

            return false;
        }

        1 reference
        internal bool Register()
        {
            DialogRegister dialog = new DialogRegister();

            DialogResult opcaoEscolhida = dialog.ShowDialog();

            if (opcaoEscolhida == DialogResult.OK)
            {
                EntityAuthenticatedUser = RepositorioAuthenticatedUser.RegisterUserPost(dialog.UserName, dialog.Email, dialog.Password);
                return true;
            }

            return false;
        }
    }
}
```

8.2.7. View

As interfaces gráficas do sistema foram implementadas utilizando a ferramenta de criação de UI arrastar e soltar embutida no Visual Studio, que permite agilizar o processo de criação das interfaces. Esta seção destaca os principais pontos de desenvolvimento do Sistema Desktop. Ao final desta seção, serão apresentados os principais elementos de interface construídos para esta aplicação.

8.2.8. Visão Geral

Primeiramente, foi desenvolvida a tela inicial, que funciona como a tela de login e registro de novos usuários, conforme mostrado na Figura 14. Para realizar as ações de login e registro, optou-se por utilizar diálogos, como ilustrado na Figura 15.

Figure 14. Tela Login ou Registro de Usuários

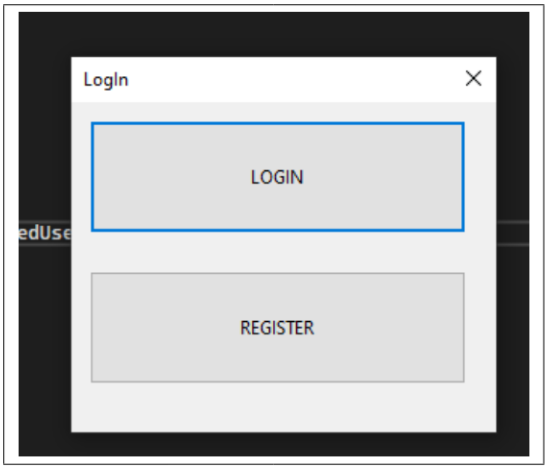
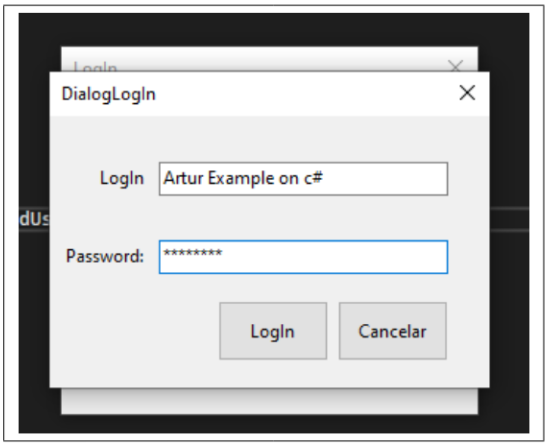


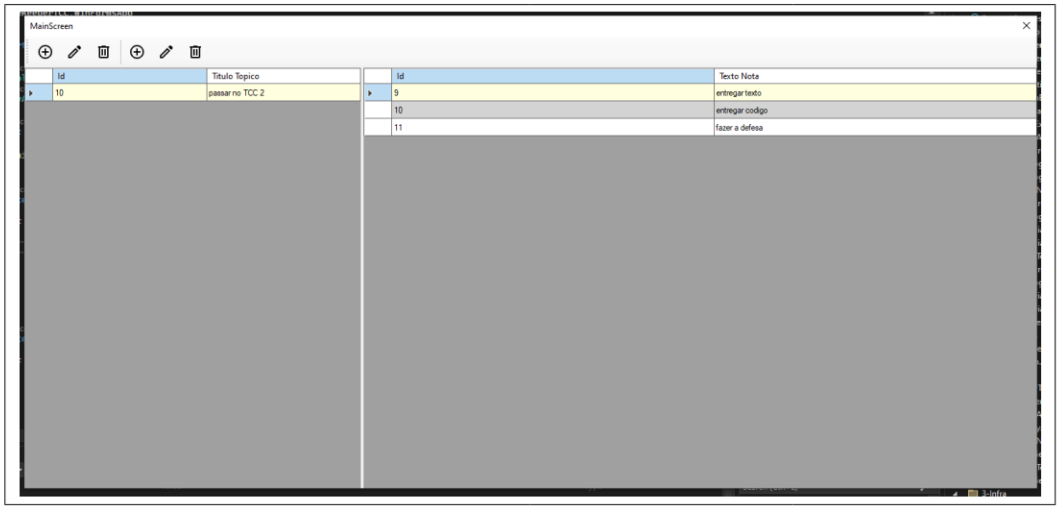
Figure 15. Tela Login



Após efetuar o login ou registro, o aplicativo transita para uma segunda tela chamada MainScreen, onde estão localizadas duas estruturas do tipo DataGridView, responsáveis por exibir a lista de tópicos e a lista de notas relacionadas ao tópico selecionado. As operações de criação, edição e exclusão de tópicos são realizadas por meio dos três primeiros botões no topo da tela, e as mesmas operações podem ser executadas para as notas utilizando os três últimos botões. Na Figura 16, é possível visualizar a aparência da tela quando existem tópicos e notas presentes. Na Figura 17, é possível visualizar a continuação da tela com mais elementos de interface.

Figure 16. Tela para manipulação de tópicos e notas 1

Figure 17. Tela para manipulação de tópicos e notas 2



9. Sistema Mobile

O Sistema Mobile é um dos três tipos de sistemas que o ProgressiveTutor ensina, e entre os três é o mais próximo ao que é ensinado no curso de Ciências da Computação. Assim como os outros dois sistemas, ao final do tutorial da área desktop, o ProgressiveTutor ensina como fazer um aplicativo simples para fazer anotações de temas e notas, tendo as funcionalidades de um banco de dados online, uma interface construída através do construtor de UI arrastar e soltar do Android Studio, e uma API para se comunicar entre as duas partes usando o Strapi e chamadas REST, além do controle de autenticação por meio de JWT enviado junto com chamadas REST que precisam de autenticação.

9.1. Comportamento e Implementação do Sistema Mobile

Infelizmente, devido a restrições de tempo, não foi possível esquematizar o comportamento e a implementação do Sistema Mobile. No entanto, o aplicativo ProgressiveTutor terá uma seção inicial abordando os dados iniciais relacionados aos Sistemas Mobile.

9.2. Avaliação do Protótipo

Este capítulo descreve em detalhes os métodos utilizados para avaliar os protótipos desenvolvidos. A avaliação envolveu a análise da experiência de uso do ProgressiveTutor como ferramenta de apoio ao aprendizado das tecnologias abordadas, bem como a verificação dos requisitos estabelecidos no Capítulo 4.

Devido a restrições de tempo, não foi possível realizar testes com usuários voluntários. Portanto, optou-se por conduzir uma simulação por parte do autor deste trabalho, atuando como um usuário potencial da aplicação ProgressiveTutor desenvolvida ao longo deste projeto.

10. Definição dos Testes

Levando em consideração os pontos levantados no capítulo 4 e na introdução, foram criadas as seguintes métricas de avaliação do ProgressiveTutor e dos aplicativos que ele ensina a fazer:

- Foi possível reduzir o número de abas abertas durante o processo de aprendizado para um número mínimo, mantendo apenas o ProgressiveTutor e, no máximo, duas outras abas (uma para o banco de dados e outra para algum recurso necessário em alguma etapa do tutorial)?
- Foram necessárias pesquisas adicionais para obter informações complementares que não estavam incluídas no tutorial, a fim de concluir a aplicação proposta na última parte de cada módulo do tutorial?
- O aplicativo do tipo Sistema Desktop ensinado pelo ProgressiveTutor oferece a funcionalidade de CRUD?
- O aplicativo do tipo Sistema Mobile ensinado pelo ProgressiveTutor oferece a funcionalidade de CRUD?
- O aplicativo do tipo Sistema Multiplataforma ensinado pelo ProgressiveTutor oferece a funcionalidade de CRUD?
- O aplicativo do tipo Sistema Desktop ensinado pelo ProgressiveTutor oferece a funcionalidade de PWA?
- O aplicativo do tipo Sistema Multiplataforma ensinado pelo ProgressiveTutor oferece a funcionalidade de PWA?

11. Execução do Teste

Para a realização dos testes, o autor fechou todos os recursos que utilizou para criar o tutorial do ProgressiveTutor e utilizou apenas o próprio tutorial para seguir cada etapa das trilhas até a criação dos aplicativos de cada tipo de sistema. Durante cada etapa, o autor analisou, tentando simular um possível usuário, e fez uma análise imparcial se cada etapa conseguiu transmitir corretamente o que se queria ensinar.

12. Resultados dos Testes

Para a análise dos resultados obtidos, é importante ressaltar que eles foram elaborados pelo próprio autor, e, apesar de ter tentado manter a imparcialidade na análise, pode ter havido uma tendência a enfatizar os aspectos positivos dos resultados.

Ao analisar a eficácia do ProgressiveTutor em reduzir a quantidade de páginas visitadas e abertas simultaneamente, observou-se que o autor, durante a pesquisa e produção de cada parte do tutorial para cada sistema, costumava ter entre 25 a 40 abas abertas. Em contraste, ao utilizar apenas as informações do tutorial, foram abertas, no máximo, três abas ao mesmo tempo. Houve uma melhoria significativa nesse aspecto. Embora possa parecer um resultado óbvio, é comum ao buscar soluções na internet encontrar versões diferentes das tecnologias que o usuário está utilizando, o que leva o usuário a manter várias abas abertas até que o problema seja resolvido ou a funcionalidade seja implementada. A preocupação é fechar uma aba e não conseguir encontrar novamente a solução correta que, inicialmente, pode ter falhado, mas que poderia ser a solução adequada após a correção de outro aspecto.

Em relação à necessidade de recorrer a outras fontes de informação além do aplicativo, observou-se que não foi necessário, mas houve a necessidade de retornar a uma parte anterior do tutorial multiplataforma na configuração de variáveis de ambiente, devido à dificuldade encontrada nessa etapa de configuração da aplicação.

Quanto à capacidade de realizar operações CRUD nos aplicativos desenvolvidos, houve uma diferença significativa entre quando o autor os configurou como desenvolvedor e como usuário. Enquanto como desenvolvedor foi necessário estudar a documentação das ferramentas e realizar muitas tentativas e erros para obter os resultados desejados nas chamadas de API, como usuário, a ”resposta correta” já estava disponível, resultando em um processo mais rápido. A análise revela que, como desenvolvedor, o processo foi mais trabalhoso, mas proporcionou um maior entendimento do funcionamento das ferramentas. Já como usuário, ao avançar no tutorial para ver a resposta, o resultado desejado foi obtido de maneira mais rápida, porém sem um aprendizado tão aprofundado quanto o do autor. Essa é uma consideração importante, e versões futuras do tutorial podem implementar formas de ”travar” o progresso até que certas métricas sejam atingidas para medir o real esforço e aprendizado do usuário. Essas métricas podem incluir o tempo gasto em cada etapa, bem como a submissão do código desenvolvido até o momento, seguida de uma análise desse código para avaliar o nível de compreensão do usuário em relação ao tutorial.

Em relação à capacidade de transformar tanto o ProgressiveTutor quanto o aplicativo desenvolvido para o Sistema Multiplataforma em PWAs, o processo foi desafiador como desenvolvedordevido a uma incompatibilidade inicial entre o código desenvolvido e as bibliotecas utilizadas para adicionar essa funcionalidade. Felizmente, ao longo dos testes, a ferramenta utilizada recebeu atualizações que corrigiram esse problema. Por outro lado, como usuário, ocorreu uma situação semelhante à mencionada anteriormente, em que o usuário não precisou fazer tanto esforço quanto o autor.

Em conclusão, o protótipo desenvolvido obteve resultados satisfatórios. No entanto, é perceptível que houve uma facilidade talvez excessiva em comparação com a implementação de um aplicativo sem o auxílio do protótipo em cada tipo de sistema. Portanto, sugere-se como trabalho futuro expandir cada um dos módulos existentes para

permitir que o usuário tente desenvolver um aplicativo semelhante, mas com menos assistência, a fim de avaliar melhor o quanto o usuário realmente está aprendendo com o aplicativo, em comparação com simplesmente seguir as instruções fornecidas pelo mesmo.

13. Conclusões e Trabalhos Futuros

Neste trabalho, foi desenvolvido o protótipo de um aplicativo chamado ProgressiveTutor, que adota a abordagem proposta por (TILBURY; MESSNER, 1997). O aplicativo incorpora exemplos de código e seus respectivos resultados, apresentados em um formato de trilha. O usuário é guiado ao longo de cada etapa do tutorial, seguindo a trilha de aprendizado de um sistema ou área específica de interesse.

A principal ideia por trás do ProgressiveTutor é reunir e organizar conteúdos fragmentados disponíveis na internet em uma única ferramenta, facilitando o processo de aprendizado em áreas específicas de Tecnologia da Informação, como multiplataforma, desktop e mobile.

O protótipo do ProgressiveTutor tem como objetivo otimizar o tempo e a energia investidos no processo de aprendizado desses sistemas, fornecendo uma trilha que abrange desde os conceitos básicos até a etapa final, que engloba a implementação de front-end, back-end, API de comunicação entre eles e um sistema de autenticação. Para fins didáticos, foi selecionado um conjunto de tecnologias para cada tipo de sistema, a fim de fornecer exemplos práticos de implementação de todos esses requisitos. É importante ressaltar que o foco principal é ensinar os conceitos subjacentes, em vez das tecnologias específicas. Além disso, o ProgressiveTutor foi projetado de forma flexível, permitindo a adição de módulos adicionais ou a expansão dos módulos existentes.

14. Lições Aprendidas

No início do projeto, estabelecemos um escopo ambicioso para os protótipos, com alta complexidade e quantidade de módulos. Infelizmente, devido a restrições de tempo, não conseguimos concluir o terceiro módulo dedicado ao desenvolvimento de um aplicativo mobile, embora tenhamos realizado a parte teórica e definido as especificações.

Essa experiência nos ensinou a importância de gerenciar adequadamente o escopo de um projeto e estabelecer expectativas realistas em relação aos prazos. Agora reconhecemos que é crucial avaliar cuidadosamente os recursos disponíveis e alocar o tempo necessário para cada etapa do desenvolvimento.

Além disso, aprendemos a importância de priorizar as tarefas e adaptar o projeto quando necessário. Diante das restrições de tempo, foi essencial focar na conclusão dos módulos mais avançados, mesmo que isso significasse adiar o desenvolvimento do aplicativo mobile, cuja implementação foi designada para trabalhos futuros.

Essas lições serão valiosas para projetos futuros, ajudando-nos a evitar a sobrecarga de trabalho e garantir que nossas metas sejam alcançadas dentro dos prazos estabelecidos.

15. Trabalhos Futuros

Como o ProgressiveTutor, o aplicativo central deste trabalho, é facilmente adaptável, existem várias possíveis opções para trabalhos futuros:

- Completar o módulo do Sistema Mobile.
- Adicionar etapas intermediárias adicionais nos tutoriais existentes para facilitar a curva de aprendizado.
- Adicionar etapas posteriores adicionais nos tutoriais existentes para aprofundar os conteúdos já abordados.
- Adicionar mais um módulo com uma tecnologia mais compatível/fácil para implementar a funcionalidade de um PWA.
- Adicionar mais um módulo que não esteja listado acima.