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

Enzo Coelho Albornoz

AVALIAÇÃO AUTOMÁTICA DE QUESTÕES SOBRE AUTÔMATOS NA PLATAFORMA MOODLE

Florianópolis, Santa Catarina – Brasil
2022

Enzo Coelho Albornoz

AVALIAÇÃO AUTOMÁTICA DE QUESTÕES SOBRE AUTÔMATOS NA PLATAFORMA MOODLE

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): Jerusa Marchi, Dra.

Florianópolis, Santa Catarina – Brasil
2022

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 00:41h do dia 25 de março de 2022.

Enzo Coelho Albornoz
Avaliação automática de questões sobre Autômatos na plataforma Moodle / Enzo Coelho Albornoz; Orientador(a), Jerusa Marchi, Dra. – Florianópolis, Santa Catarina – Brasil, XX de fevereiro de 2022.
115 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. Moodle, 2. Avaliação Automática, 3. Linguagens Formais, I. Jerusa Marchi, Dra. II. Programa de Graduação em Ciências da Computação III. Avaliação automática de questões sobre Autômatos na plataforma Moodle

CDU 02:141:005.7

Enzo Coelho Albornoz

AVALIAÇÃO AUTOMÁTICA DE QUESTÕES SOBRE AUTÔMATOS NA PLATAFORMA MOODLE

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, XX de fevereiro de 2022.

Renato Cislaghi, Dr.
Coordenador(a) do Programa de
Graduação em Ciências da Computação

Banca Examinadora:

Jerusa Marchi, Dra.
Orientador(a)
Universidade Federal de Santa
Catarina – UFSC

Maicon Rafael Zatelli, Dr.
Universidade Federal de Santa Catarina –
UFSC

Elder Rizzon Santos, Dr.
Universidade Federal de Santa Catarina –
UFSC

AGRADECIMENTOS

Os agradecimentos principais são direcionados aos meus amados pais, Denise e Léo, que se esforçaram ao máximo para dar-me a educação que hoje me trouxe aqui. Agradeço a minha irmã Manuella, que me fez buscar ser um exemplo para ela. Agradeço também a meus outros familiares, que também sempre me agraciaram com amor e carinho, principalmente minhas avós Jane e Vânia e meus finados avôs Léo e Nilo. Aos meus colegas de trabalho e amigos pessoais dou meus agradecimentos, pois sempre transformaram momentos tristes e difíceis em momentos engraçados e leves. Agradeço a minha orientadora Jerusa que sempre se dispôs a me ensinar e ajudar neste trabalho, guiando-me neste caminho. À Universidade Federal de Santa Catarina e seus docentes, agradeço o ensino de qualidade que apresentaram, pois tenho certeza que utilizarei dos conhecimentos passados.

“O dom de poder mental vem de Deus, o Ser Divino e se concentrarmos nossas mentes na verdade, ficamos em sintonia com este grande poder.”

Nikola Tesla

RESUMO

O intuito deste projeto é desenvolver um sistema de avaliação automática de questões sobre Máquinas e Linguagens Formais no ambiente virtual de ensino e aprendizagem educacional Moodle. Para isso, será desenvolvido um plugin em PHP que será integrado ao Moodle. O modelo de máquinas seguirá como base um modelo derivado do JFLAP, porém adaptado para integração a banco de dados MySQL. A principal aplicação deste projeto será melhorar o modelo de questões que abordam este assunto enquanto diminui o tempo para obter as correções.

Palavras-chaves: Moodle. Avaliação Automática. Linguagens Formais.

ABSTRACT

The purpose of this project is to develop a system for automatic evaluation of questions about machines and formal languages in the Moodle virtual teaching and learning environment. For this, a PHP plugin will be developed and integrated with Moodle. The machine model will follow a model derived from JFLAP, but adapted for integration with MySQL database. The main application of this project will be to improve the model for questions that address this subject while decreasing the time to obtain corrections.

Keywords: Moodle. Automatic Evaluation. Formal Languages.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AVA	Ambiente Virtual de Aprendizagem
Moodle	Modular Object-Oriented Dynamic Learning Environment
UFSC	Universidade Federal de Santa Catarina
AVEA	Ambiente Virtual de Ensino e Aprendizagem
GNU	GNU's Not Unix
GPL	GNU Public License
HTTP	Hypertext Transfer Protocol
PHP	PHP Hypertext Preprocessor
LAMP	Linux Apache MySql PHP
LXC	Linux Containers
CGI	Common Gateway Interface
HTML	HyperText Markup Language
XMLDB	eXtensible Markup Language Database
XML	eXtensible Markup Language
JFLAP	Java Formal Languages and Automata Package

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

O crescente uso da Internet vêm transformando a sociedade em que vivemos. Mais de 80% dos domicílios brasileiros já possuem algum meio conexão à Web (ASCOM, 2021) , e isso se reflete também na educação. Tanto o ensino à distancia, quanto o presencial hoje se apoiam em ferramentas digitais e conectadas. Esse movimento foi acelerado ainda mais devido ao contexto socio-econômico do ano 2020, onde a pandemia global de COVID-19 afetou todos os sistemas de ensino. Houve, portanto, uma demanda acima do esperado para tais ferramentas digitais, dentre elas estão os Ambientes Virtuais de Aprendizagem (AVAs).

Os Ambientes Virtuais de Aprendizagem (AVAs) são softwares que, disponibilizados na internet, agregam ferramentas para a criação, tutoria e gestão de atividades que normalmente se apresentam na forma de cursos. Sendo constituídos a partir do uso de diferentes mídias e linguagens, a intenção é proporcionar não só a disponibilização de conteúdos, mas principalmente plena interatividade entre pessoas e grupos, viabilizando, por consequência, a construção do conhecimento. (GOIÁS, 2018)

O Moodle (*Modular Object-Oriented Dynamic Learning Environment*), lançado em 2002, é um dos AVAs mais conhecidos mundialmente. Dentre seus diferenciais está a sua licença GPL 3, publicando o código fonte da plataforma e permitindo que o Moodle possa ser utilizado gratuitamente. Também está presente no Moodle um sistema de *plugins* que permite a expansão de seu comportamento, adequando-se a novas necessidades. Tais diferenciais permitiram que milhares de instituições ao redor do mundo utilizassem tal ferramenta.

1.2 JUSTIFICATIVA

Por ser uma ferramenta tão abrangente e generalista, é notório que o Moodle não consiga abordar com perfeição todos os seus casos de uso. Muitos educadores, por lecionarem disciplinas com conteúdo complexo, não conseguem alcançar grande agilidade na correção de atividades. Tal fato ocorre com professores que abordam questões de Teoria da Computação e Linguagens Formais (informação verbal)[1].

Isso acontece pois o modelo de atividades usado por grande parte dos professores não consegue ser devidamente aplicada ao sistema de questões com avaliação automática do Moodle, tendo de serem manualmente corrigidas pelo docente e portanto, atrasando a correção. Isto se complica ainda mais no caso do ensino de teoria dos autômatos, visto que a correção de uma máquina pode rapidamente necessitar de um cálculo exponencial de estados de máquina. Este atraso acaba por influenciar

[1] Em conversa particular com a Profª Jerusa Marchi em 2021

negativamente o ensino, pois muitas vezes o aluno não consegue esclarecer a tempo suas dúvidas sobre uma correção ou assunto abordado, levando à piora na absorção do conteúdo.

Tendo conhecimento do sistema de plugins do Moodle, existe alguma solução integrada ao Moodle que permita agilizar as avalizações e portanto aprimorar o *feedback* do professor ao aluno?

1.3 OBJETIVOS

Os objetivos, geral e específicos, são mostrados a seguir.

1.3.1 Geral

Elaborar e disponibilizar um plugin para o ambiente virtual de aprendizagem Moodle que permita a avaliação automática de questões sobre Autômatos, no escopo de reconhecimento de linguagens, com a finalidade de permitir agilidade nas correções e no retorno dos professores aos os alunos.

1.3.2 Específicos

- Compreender a programação do Moodle e como expandir seu comportamento através de plugins;
- Implementar um plugin para avaliação automática de Autômatos (Finitos, Pilha e Turing) através do reconhecimento de linguagens baseado em testes de aceitação;
- Desenvolver um guia para a integração do plugin com Moodle UFSC;
- Desenvolver um guia de utilização do plugin tanto para alunos quanto para professores;
- Publicar o plugin em um repositório aberto para que a comunidade acadêmica o utilize.

1.4 APRESENTAÇÃO DO TRABALHO

O capítulo 2 apresenta os materiais e métodos necessários para o desenvolvimento do plugin, fundamentando tanto o âmbito teórico quanto prático a fim de elucidar o funcionamento do Moodle, seu sistema de plugins e uma breve explicação de Autômatos a fim de entender os pontos necessário para uma modelagem correta de seu funcionamento. No capítulo 3, analisa-se ferramentas pré existentes de ensino de autômatos, bem como outras ferramentas, e ao fim, uma análise de plugins pré-existentes do Moodle. No capítulo 4, é apresentado todo o desenvolvimento do sistema, passando

pela modelagem e mapeamento para banco de dados até a integração com sistemas internos do Moodle, como internacionalização, processamento de questões e interface gráfica. No capítulo 5 é apresentado o guia de uso do sistema e no capítulo 6 tem-se as considerações finais e trabalhos futuros.

2 MATERIAIS E MÉTODOS

2.1 AMBIENTES VIRTUAIS DE APRENDIZAGEM

No ensino, é natural utilizar os ambientes mais adequados que melhor favoreçam o aprendizado. Esses ambientes são conhecidos como Ambientes Virtuais de Ensino e Aprendizagem (AVEA) ou Ambientes Virtuais de Aprendizagem (AVA).

Segundo Milligan (1999), pode-se definir o termo AVA como sendo o software que é executado em um servidor e é projetado para gerir e administrar vários aspectos do aprendizado. Nota-se que o foco do conceito está nos objetivos cumpridos, visto que logo após esta definição em Milligan (1999), é dito que o software não necessariamente está restrito a um pacote único, podendo ser um composto de ferramentas individuais que atendem à uma lista de funcionalidades essenciais. Essa lista de funcionalidades seria composta por:

- Disponibilização e gerenciamento de cursos;
- Controle de acesso;
- Elementos administrativos de registro e acompanhamento e progresso do aluno;
- Acompanhamento de cronograma;
- Avaliação (geralmente formativa);
- Comunicação em diversas formas (um para um, um para muitos, síncronas e assíncronas);
- Espaço pessoal para que participantes armazenem e troquem materiais;
- Um banco de recursos, sendo ele menos formal que os materiais de estudo. Um exemplo seria uma lista de perguntas frequentes ou um banco de dados com procura baseada em palavras chave;
- Serviços de apoio. Uma sistema de ajuda online sobre o AVA, por exemplo.
- Ferramentas de criação e manutenção de materiais de estudo.

Atualmente existem diversas soluções de AVAs. Elas variam desde produtos pagos de código fechado até produtos gratuitos de código aberto. Segundo Bri *et al.* (2009) , o Moodle é o AVA mais utilizado, seguido de Sakai e WebCT.

2.1.1 Moodle

Moodle é uma plataforma de aprendizagem cujo objetivo é permitir a criação de ambientes de aprendizagem personalizados. Ela é uma solução completa, robusta, integrada e segura que abrange todos os aspectos que um AVA deveria possuir. Seu nome é derivado do acrônimo *Modular Object-Oriented Dynamic Learning*, sendo a referência a modular o aspecto mais relevante dentro de seu projeto.

A plataforma nasceu oficialmente em 20 de agosto de 2002, desenvolvida inicialmente por Martin Dougiamas, após inúmeros protótipos descartados. O projeto rapidamente obteve adeptos, visto que as soluções da época não se utilizavam do poder da internet, enquanto na visão de Martin, a rede mundial de computadores poderia ser de grande valia. O Moodle é distribuído sob a licença *GNU Public License v3*, sob os quais regem quatro fundamentos:

- a liberdade de usar o software para qualquer finalidade;
- a liberdade de mudar o software de acordo com suas necessidades;
- a liberdade de compartilhar o software com seus amigos e vizinhos;
- a liberdade de compartilhar as mudanças que você faz.

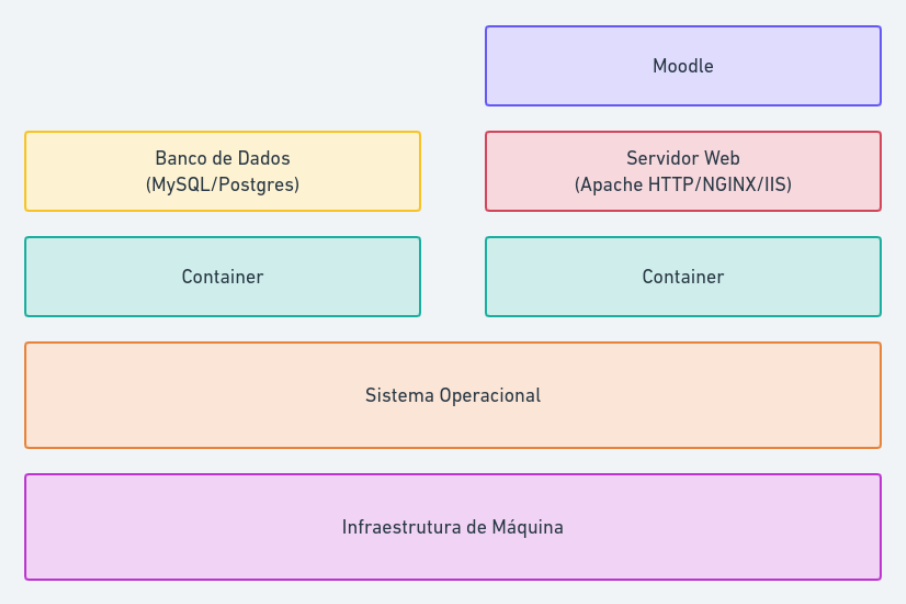
Por esses motivos o Moodle é considerado, segundo GNU (2021), um software livre. Como o software livre permite a contribuição do público no código fonte, a comunidade do Moodle ajuda a manter a ferramenta, melhorando a cada dia seus aspectos e ampliando suas funcionalidades.

2.1.2 Ambiente de Execução

Para entender o Moodle, pode-se utilizar uma visão *top-down*, onde analisa-se a interação entre diversos componentes da plataforma para depois analisá-los mais detalhadamente. Mesmo sendo uma solução, em teoria, única, o Moodle, como diz o próprio nome, é composto de diversos módulos os quais são executados em um ambiente de servidor.

O ambiente recomendado, ilustrado na figura 1, é primariamente composto por um servidor Apache HTTP, responsável por executar o código do Moodle e apresentar o conteúdo pela internet, e um banco de dados MySQL, onde o conteúdo é armazenado em si. O código fonte do Moodle é escrito em PHP, uma linguagem de programação de script. De modo a automatizar a estruturação do ambiente e melhorar a segurança das aplicações pode-se executar estes softwares de suporte em contêineres.

Figura 1 – Estrutura LAMP (Linux Apache MySql PHP) de um servidor executando a plataforma Moodle utilizando contêineres



Fonte: Próprio

2.1.2.1 Contêineres e Docker

Um contêiner é uma unidade padrão de software que empacota o código e todas as suas dependências para que o aplicativo seja executado de forma rápida e confiável de um ambiente de computação para outro (DOCKER, 2021b).

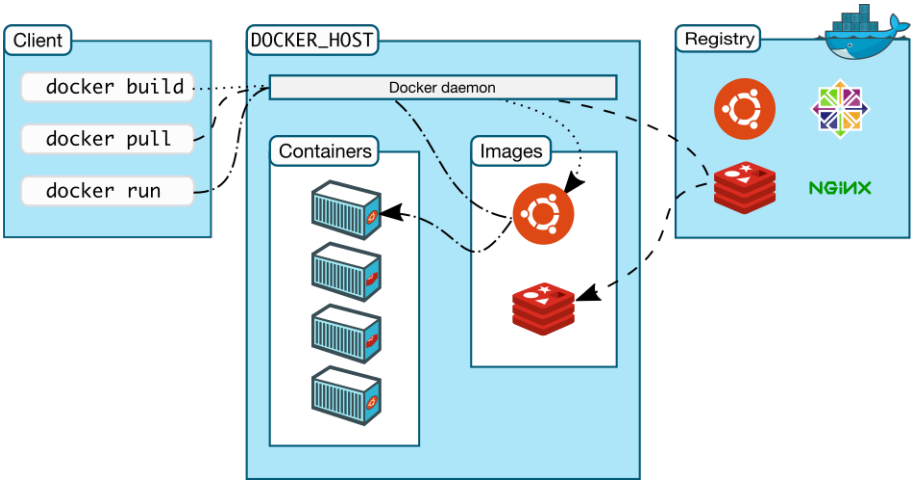
Esse conceito de isolamento de processos não é recente. A ideia estava presente já nos anos 70 através do comando *chroot*, cujo propósito era alterar o diretório onde era montado o sistema de arquivos, isolando assim os subprocessos dos outros arquivos presente na máquina (MOUAT, 2015).

O sistema operacional *FreeBSD* evoluiu essa tecnologia através dos *jails* (jaulas) onde processos eram isolados através da criação de sistema de arquivos, endereço de rede e *hostname* para cada processo dentro das “jaulas”.

Em 2008, nasceu o projeto LXC (*Linux Containers*), patrocinado pela Canonical, cuja principal inovação foi o uso do *CGroups*, uma tecnologia desenvolvida pela Google para o *kernel* (núcleo do sistema operacional) Linux. O *CGroups* permitia o gerenciamento de recursos como processador, memória, disco, rede, etc. e direcionar estes recursos a uma uma coleção qualquer de processos (MOUAT, 2015).

A partir desta base tecnológica, em 2013, surgiu a ferramenta que popularizou os contêineres. O Docker surgiu como um projeto de código aberto com seu principal sistema, o Docker Engine. Este sistema é responsável por gerenciar a montagem, orquestração, distribuição, sistemas de arquivos e rede dos contêineres executados por ele, diferentemente de outras ferramentas como o LXC, onde era requerido grande conhecimento e trabalho manual para configurar corretamente a execução.

Figura 2 – Arquitetura em alto nível do Docker



Fonte: Docker (2021a)

Além do Docker Engine, existe todo um ecossistema montado pela Docker Inc., desenvolvedora da ferramenta. O Docker Hub é uma plataforma, denominada registro, onde a comunidade pode compartilhar contêineres pré configurados contendo todos os arquivos necessários para executar as aplicações. Tais conteires são distribuídos em arquivos denominados imagens. Pode-se ver a sua integração com o Docker Engine na figura 2.

Essas imagens contém uma série de informações, com a mais importante sendo as *layers* (camadas). Elas são uma série de comandos a serem executados a fim de configurarem todo o ambiente dentro daquele ambiente isolado, seja carregar um arquivo contido dentro da imagem ou até mesmo executar um programa. Esse repositório de imagens, que é o Docker Hub também pode ser executado como um contêiner e ser executado localmente. Um exemplo disso é em ambientes corporativos onde um registro é criado para armazenar softwares internos da corporação.

2.1.2.2 Apache HTTP

O Apache HTTP é um servidor web, isto é, uma aplicação que se comunica através do protocolo HTTP (*HyperText Transfer Protocol*). Ele é baseado no software conhecido como *httpd* (*HyperText Transfer Protocol Daemon*), desenvolvido por Rob McCool.

Em 1995, após diversas extensões terem sido criadas pela comunidade, um pequeno grupo de *webmasters* (desenvolvedores de páginas web) decidiu, através de comunicação pela internet, unificar todas essas funcionalidades e desenvolver uma aplicação em comum, distribuindo então o servidor Apache HTTP. Após um ano de seu lançamento o Apache já ultrapassava o *httpd* de McCool. Já em 1997, o Apache já era usado em mais de meio milhão de páginas na rede mundial de computadores

(FIELDING; KAISER, 1997).

McCool, junto com outros especialistas como John Franks, Ari Luotonen, George Phillips e Tony Sanders criaram também o CGI (*Common Gateway Interface*), cuja função é permitir que servidores como o Apache executem programas de linha de comando afim de construírem páginas web dinamicamente. O CGI permitiu que diversas linguagens de programação fossem utilizadas, dentre elas destacam-se o Perl, ASP.NET e principalmente o PHP.

Por todo este histórico, além da estabilidade e funcionalidades do servidor Apache, o Moodle o utiliza como servidor Web recomendado.

2.1.2.3 Linguagem PHP

O PHP é uma linguagem de programação de código aberto, lançada em 1995, desenvolvida por Rasmus Lerdorf para ser utilizada através do CGI. Seu foco é ser extremamente modularizada afim de obter performance em ambientes de servidores Web (CONVERSE; PARK, 2003).

Sua principal característica é possibilidade de incluir fragmentos de códigos em páginas HTML (*HyperText Markup Language*). Assim, conforme o cliente for requisitando novas páginas esse código é executado. Seu uso em páginas HTML é principalmente o acesso ao banco de dados para recuperar informações e aplicar regras de negócio a fim de condicionalmente apresentar fragmentos de páginas ao usuário nas respostas do servidor.

Outras características da linguagem são sua excelente performance e robustez ao mesmo tempo que possui suporte à orientação a objetos e tipagem dinâmica. Tais fatores positivos foram importantes para a escolha da linguagem pelo Moodle, permitindo que ele ficasse à frente de outros concorrentes (MOODLE, 2020c).

2.1.2.4 Banco de Dados e MySQL

Bancos de Dados são, sem dúvida, o ponto chave da maioria das aplicações modernas. A necessidade de armazenar e consultar dados performativamente é um dos principais focos de pesquisas e inovações dos últimos anos. Segundo Elmasri, Navathe, Pinheiro *et al.* (2005), um banco de dados é uma coleção de dados relacionados, onde dados são fatos que podem ser gravados e que possuem um significado implícito.

Contudo, afim de especificar banco de dados em um contexto tecnológico, pode-se restringir essa definição usando, segundo Elmasri, Navathe, Pinheiro *et al.* (2005), as seguintes propriedades implícitas:

- Um banco de dados representa alguns aspectos do mundo real, sendo chamado, às vezes, de mini-mundo ou de universo de discurso (UoD). As mudanças no

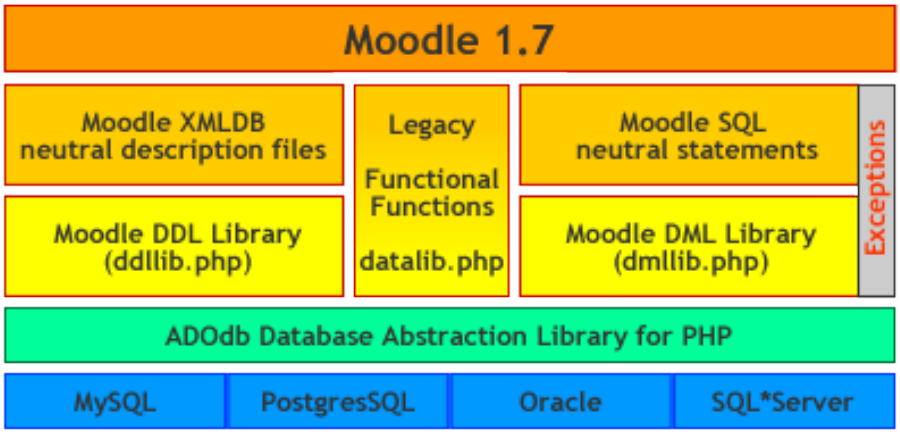
mini-mundo são refletidas em um banco de dados;

- Um banco de dados é uma coleção lógica e coerente de dados com algum significado inerente. Uma organização de dados ao acaso (randômica) não pode ser corretamente interpretada como um banco de dados;
- Um banco de dados é projetado, construído e povoado por dados, atendendo a uma proposta específica. Possui um grupo de usuários definido e algumas aplicações preconcebidas, de acordo com o interesse desse grupo de usuários;

O banco de dados recomendado pelo Moodle é MySQL, visto que fora o banco de dados mais utilizado na época de concepção da plataforma. Por questões de licença, hoje é utilizado o MariaDB, um projeto baseado na última versão do MySQL sem o problema citado (MOODLE, 2020c).

O Moodle evoluiu e hoje possuiu suporte também ao PostgresSQL e Microsoft SQL Server utilizando uma abstração de conexão e modelagem de banco de dados utilizando XMLDB como visto na figura 3.

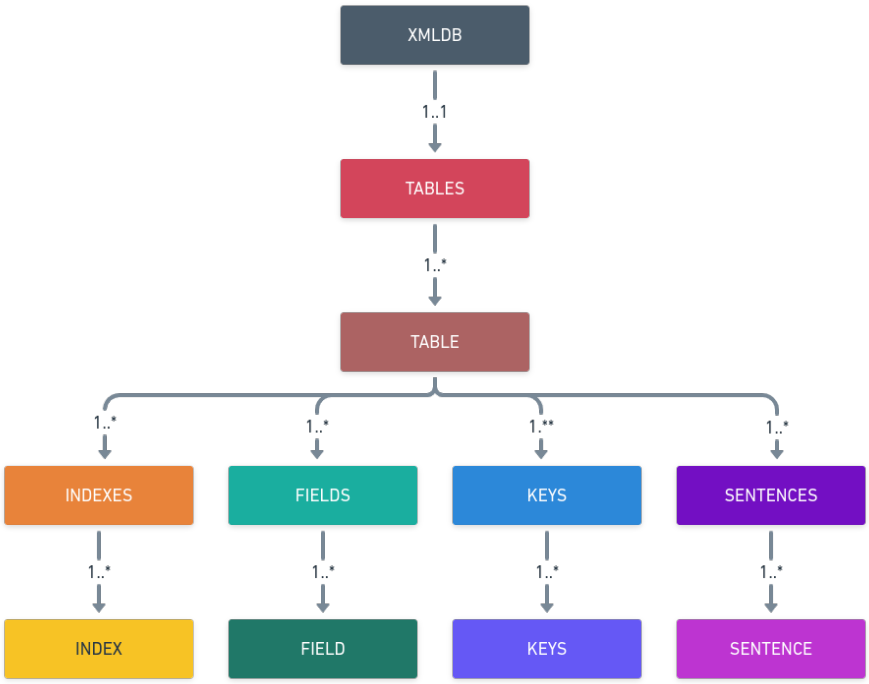
Figura 3 – Diagrama exemplificando o sistema de abstração de banco de dados do Moodle



Fonte: Moodle (2020c)

O XMLDB é um arquivo XML que descreve uma tabela de banco de dados em alto nível. O arquivo é composto por um elemento “XMLDB”, esse elemento contem uma lista de tabelas através de um elemento "TABLES", que por sua vez possuem elementos do tipo "FIELDS"(campos), "KEYS"(chaves), "INDEXES"(índices) e "SENTENCES"(sentenças). Pode-se ver essa estrutura na figura 4 e um exemplo na Figura 5.

Figura 4 – Organização dos elementos do XMLDB



Fonte: Próprio

Figura 5 – Exemplo de Arquivo XMLDB

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<XMLDB PATH="question/type/formulas/db" VERSION="20130130" COMMENT="XMLDB file for Moodle question/type/formulas"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../../../../lib/xmldb/xmldb.xsd"
>
  <TABLES>
    <TABLE NAME="qtype_formulas_options" COMMENT="Options for formulas question type">
      <FIELDS>
        <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="true"/>
        <FIELD NAME="questionid" TYPE="int" LENGTH="10" NOTNULL="true" DEFAULT="0" SEQUENCE="false"/>
        <FIELD NAME="varsrandom" TYPE="text" NOTNULL="false" SEQUENCE="false"/>
        <FIELD NAME="varsglobal" TYPE="text" NOTNULL="false" SEQUENCE="false"/>
        <FIELD NAME="correctfeedback" TYPE="text" NOTNULL="true" SEQUENCE="false" COMMENT="Feedback shown for any correct response."/>
        <FIELD NAME="correctfeedbackformat" TYPE="int" LENGTH="2" NOTNULL="true" DEFAULT="0" SEQUENCE="false"/>
        <FIELD NAME="partiallycorrectfeedback" TYPE="text" NOTNULL="true" SEQUENCE="false" COMMENT="Feedback shown for any partially correct response."/>
        <FIELD NAME="partiallycorrectfeedbackformat" TYPE="int" LENGTH="2" NOTNULL="true" DEFAULT="0" SEQUENCE="false"/>
        <FIELD NAME="incorrectfeedback" TYPE="text" NOTNULL="true" SEQUENCE="false" COMMENT="Feedback shown for any incorrect response."/>
        <FIELD NAME="incorrectfeedbackformat" TYPE="int" LENGTH="2" NOTNULL="true" DEFAULT="0" SEQUENCE="false"/>
        <FIELD NAME="shownumcorrect" TYPE="int" LENGTH="2" NOTNULL="true" DEFAULT="0" SEQUENCE="false" COMMENT="If true, then when the user gets a question partially correct, tell them how many parts they got correct alongside the feedback."/>
        <FIELD NAME="answernumbering" TYPE="char" LENGTH="10" NOTNULL="true" DEFAULT="none" SEQUENCE="false" COMMENT="Indicates how and whether the choices should be numbered."/>
      </FIELDS>
      <KEYS>
        <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
        <KEY NAME="questionid" TYPE="foreign" FIELDS="questionid" REFTABLE="question" REFFIELDS="id"/>
      </KEYS>
    </TABLE>
  </TABLES>
</XMLDB>
```

Fonte: Bauer (2021)

2.1.3 Sistema de Plugins

O Moodle apresenta um sistema de plugins extremamente poderoso. Ele permite que a comunidade interaja com os módulos da plataforma de maneira a expandir suas funcionalidades. Por ser um projeto de grande porte, o Moodle possui centenas de páginas de documentação, cada uma abordando um tópico diferente do sistema.

Exitem mais de 61 categorias de *plugins* disponíveis para aprimorarem o Moodle. Contudo, as mais conhecidas são:

- *Blocks*: Pequenos blocos de informações ou ferramentas que podem ser arrastados pelas páginas;
- *Webservice*: Permite definir novos protocolos para comunicação entre cliente e servidor, tal como SOAP, XML-RPC, REST, etc.;
- *Themes*: Permite alterar a aparência do Moodle alterando o código que apresenta as páginas;
- *Activity*: Permite desenvolver novas atividades disponíveis em um curso. Esse plugin é a base para os sistemas de Questões, Fóruns e Trabalhos.
- *QType*: Permite adicionar novos tipos de questões, como é o caso deste trabalho.

Cada um dos 61 plugins possui sua documentação trazendo suas nuances. Contudo alguns pontos em comum existem, como o arquivo de manifesto *version.php* , onde o plugin define um nome, versões miníma e máxima e outras definições. Tem-se também o arquivo *db/install.xml* , cuja função é definir as tabelas necessárias para o plugin usando o formato XMLDB. Tem-se também o arquivo de textos *lang/<código de língua>/<nome do plugin>.php* , cuja utilização permite abstrair os textos apresentados aos usuários e dar suporte à internacionalização.

A estrutura de plugins de tipo *QTypes* necessita de alguns outros arquivos a mais. É preciso definir um arquivo para o formulário preenchido pelo professor ao criar uma nova questão. Também é necessário o arquivo que define a apresentação da questão e coleta das respostas dos alunos.

2.2 CONCEITOS SOBRE AUTÔMATOS

Segundo (HOPCROFT; MOTWANI; ULLMAN, 2001), pode-se definir autômatos como modelos matemáticos que nos permitem especificar linguagens formais através do reconhecimento delas. Uma linguagem é qualquer conjunto de sentenças. Uma sentença é formada pela concatenação de um número finito de símbolos pertencentes a um alfabeto. Um alfabeto é um conjunto finito de símbolos, que podem ser das mais variadas formas, como caracteres latinos, gregos, ou até mesmo binários (zeros e uns).

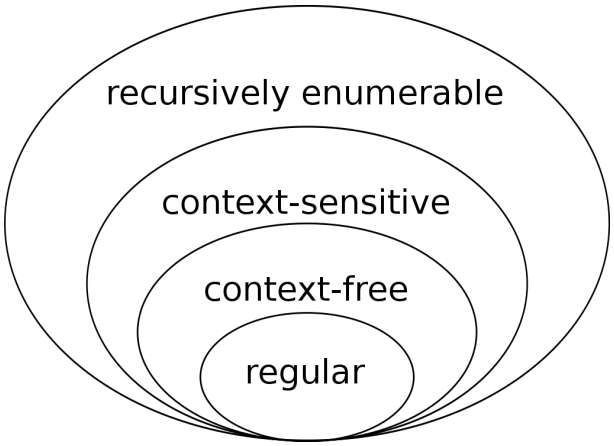
2.2.1 Gramáticas

Uma linguagem também pode descrita através de gramáticas. Gramáticas são mecanismo que descrevem as regras de formação das sentenças de uma linguagem.

2.2.2 Tipos de Linguagens

Segundo (CHOMSKY, 1956), pode-se categorizar as gramáticas em uma hierarquia de 4 tipos, sendo cada uma delas um subconjunto do tipo acima na ordem como visto na Figura 6. Cada tipo de gramática descreve um conjunto específico de linguagens.

Figura 6 – Ordem de conjuntos descritos pela hierarquia de Chomsky



Fonte: Finkelstein (2010)

2.2.2.1 Linguagens recursivamente enumeráveis

Esta categoria de linguagens é a de mais alta ordem e compreende os subconjuntos recursivamente enumeráveis pertencente ao conjunto de todas a sentenças possíveis de um dado alfabeto.

Um conjunto é dito recursivamente enumerável quando existe um algoritmo capaz de enumerar todos os membros do conjunto (uma gramática). É equivalente dizer que se um conjunto possui um algoritmo que reconhece apenas os membros dele, ele então será um conjunto recursivamente enumerável (reconhecido por um autômato).

As Máquinas de Turing (MT), conforme será explicado na seção 2.2.5 são responsáveis por reconhecer esta classe linguagem. Dá-se se o nome de tipo 0 àquelas gramáticas que reconhece essa categoria de linguagens.

2.2.2.2 Linguagens sensíveis ao contexto

As linguagens sensíveis ao contexto são definidas pelo conjunto de linguagens reconhecíveis por uma variante da MT, a Máquina de Turing Linearmente Limitada

(MTLL), isto é, com memória finita. Uma definição formal para Máquinas de Turing é dada na seção 2.2.5. Às gramáticas que geram esse conjunto de linguagens, é dado o nome de tipo 1.

2.2.2.3 Linguagens livre de contexto

Linguagens livres de contexto, são aquelas reconhecidas por autômatos de pilha não determinísticos (explicado na seção 2.2.4). As gramáticas geradoras desse tipo de linguagem são conhecidas como tipo 2. Suas aplicações no dia a dia da computação são enormes, visto que a maioria das expressões aritméticas fazem parte de alguma linguagem livre de contexto.

2.2.2.4 Linguagens regulares

Linguagens regulares é o conjunto de linguagens reconhecidas por autômatos finitos. Seu mecanismo gerador são as gramáticas de tipo 3, conhecidas também como gramáticas regulares.

2.2.3 Autômatos Finitos

Um autômato finito (AF) é, segundo Hopcroft, Motwani e Ullman (2001) um mecanismo simples para o reconhecimento de gramáticas do tipo 3 e portanto, de linguagens regulares.

Os AF apresentam duas variantes, sendo elas os Autômatos Finitos Determinísticos (AFD) e os Autômatos Finitos Não Determinísticos (AFND).

2.2.3.1 Autômatos Finitos Determinísticos

Formalmente, segundo Sipser (2013), podemos definir um AFD como uma quíntupla:

$$(Q, \Sigma, \delta, q_0, F) \tag{1}$$

Onde,

1. Q é um conjunto finito denominado **estados**.
2. Σ é um conjunto finito de símbolos denominado **alfabeto**.
3. $\delta : Q \times \Sigma \rightarrow Q$ é a **função de transição**. A função de transição define o comportamento de alteração de um estado para outro baseado no simbolo atual na entrada e no estado atual.
4. $q_0 \in Q$ é o **estado inicial**.
5. $F \subseteq Q$ é o **conjunto de estados de aceitação**.

Para uma dada máquina *M*, o conjunto *A* reconhecido pela máquina *M* é denominado linguagem (*L*) da máquina, tal que $L(M) = A$.

2.2.3.2 Autômatos Finitos Não Determinísticos

Diferentemente de um AFD, um AFND pode estar em mais de estado ao mesmo tempo. Formalmente, segundo Sipser (2013), podemos definir um AFND como uma quíntupla:

$$(Q, \Sigma, \delta, q_0, F) \tag{2}$$

Onde,

1. *Q* é um conjunto finito denominado **estados**.
2. Σ é um conjunto finito de símbolos denominado **alfabeto**.
3. $\delta : Q \times (\Sigma \cup \varepsilon) \rightarrow P(Q)$, onde $P(Q)$ denota o fecho positivo de Q e ε representa a palavra vazia, isto é, uma sentença de tamanho 0 é a **função de transição**.
4. $q_0 \in Q$ é o **estado inicial**.
5. $F \subseteq Q$ é o **conjunto de estados de aceitação**.

Diz-se que uma máquina $N = (Q, \Sigma, \delta, q_0, F)$ aceita w, sendo w uma cadeia sobre o alfabeto Σ, se é possível escrever $w = y_1, y_2, ..., y_m$, onde cada y_i é um membro de Σ_ε e uma sequência de estados $r_0, r_1, ..., r_m$ existe em Q com três condições (SIPSER, 2013):

1. $r_0 = q_0$,
2. $r_{i+1}\delta(r_i, y_{i+1})$, para $i = 0, ..., m - 1$, e
3. $r_m \in F$.

Uma propriedade interessante sobre os AFND é que eles são equivalentes aos AFD. Para todo AFND M_n, existe um AFD M_d que reconhece a mesma linguagem que M_n, isto é, $L(M_n) = L(M_d)$. Os algoritmos que realizam a transformação AFND para AFD foram descritos tanto por Sipser (2013) quanto por Hopcroft, Motwani e Ullman (2001).

2.2.4 Autômatos de Pilha

Os Autômato de Pilha (AP) são semelhantes aos autômatos finitos não determinísticos, com a diferença estando na adição de um componente de pilha. A pilha permite um recurso de memória adicional (de tamanho teórico infinito), permitindo-a

reconhecer linguagens gerada por gramáticas livre de contexto (tipo 2). A pilha permite que o autômato armazene símbolos através de operações de *empilhar*, adicionando um símbolo ao topo da pilha, e *desempilhar*, onde o símbolo no topo da pilha é removido (SIPSER, 2013).

Embora exista a variante determinística do autômato de pilha, ela não é equivalente à variante não determinística em poder de expressão e portanto não consegue reconhecer todas as linguagens geradas pelas gramáticas de tipo 2 (SIPSER, 2013).

Este modelo computacional não determinístico é, segundo Sipser (2013), uma sêxtupla:

$$(Q, \Sigma, \Gamma, \delta, q_0, F) \tag{3}$$

Onde,

1. Q é o conjunto de **estados**.
2. Σ é o **alfabeto de entrada**.
3. Γ é o **alfabeto de pilha**.
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \rho(Q \times \Gamma_\varepsilon)$, onde $\Sigma_\varepsilon = \Sigma \cup \varepsilon$ e $\Gamma_\varepsilon = \Gamma \cup \varepsilon$ é a **função de transição**. A função de transição define a mudança de um estado para outro e o comportamento da pilha (empilhar e desempilhar) baseado no simbolo atual na entrada, estado atual e no símbolo no topo da pilha.
5. $q_0 \in Q$ é o **estado inicial**.
6. $F \subseteq Q$ é o **conjunto de estados de aceitação**.

Um AP $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ aceita uma entrada w se w pode ser escrito como $w = y_1, y_2, ..., y_m$, onde cada $w_i \in \Sigma_\varepsilon$ e sequências de estados $r_0, r_1, ..., r_m \in Q$ e cadeias $s_0, s_1, ..., s_m \in \Gamma_\varepsilon$, que representam a sequencia de conteúdo da pilha de M, existem e satisfazem as seguintes três condições:

1. $r_0 = q_0$ e $s_0 = \varepsilon$. Isto é, a maquina começa no estado q_0 e com a pilha vazia.
2. $\forall i \in \mathbb{N}$ e $0 \leq i < m$, temos $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, onde $s_i = at$ e $s_{i+1} = bt$ para $a, b \in \Gamma_\varepsilon$ e $t \in \Gamma^*$. Esta condição define a aplicação correta das transições, atualizando a pilha, estado e símbolo de entrada.
3. $r_m \in F$. Esta condição define que a máquina deve terminar em um estado de aceitação ao final do processamento da entrada.

2.2.5 Autômatos de Turing

Uma máquina (autômato) de Turing (MT) é, segundo Sipser (2013), semelhante a uma autômato finito, com a diferença de conter uma memória ilimitada e irrestrita, no formato de uma fita. Este é o modelo computacional mais próximo de um computador de propósito geral, conseguindo reconhecer linguagens geradas por gramáticas de tipo 0. Seu modelo de fita é divida em células e possui um cabeçote, isto é, um indicador para qual célula na fita a máquina está operando. Uma MT pode tanto ler quanto escrever sobre o atual símbolo apontado pelo cabeçote. Além disso, outra possibilidade é deslocar este cabeçote sobre os símbolos na fita. De início, a fita contém apenas a sentença de entrada e está em branco em todo o restante. A parte em branco denota-se através do símbolo $\sqcup$. A máquina para ao entrar em um estado listado como aceitação ou rejeição, independente do estado da fita. Caso esses estados não sejam atingidos a máquina não para. Para casos onde não há parada garantida, diz-se que a sentença faz parte de uma linguagem indecidível.

Formalmente, em Sipser (2013) é definido uma MT como uma séptupla:

$$(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r) \tag{4}$$

Onde,

1. Q é o conjunto de **estados**.
2. Σ é o **alfabeto de entrada**, não contendo o símbolo em branco $\sqcup$.
3. Γ é o **alfabeto de fita**, onde $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times E, D$ é a **função de transição**. A função de transição define o comportamento de movimentação de um estado para outro, o comportamento do cabeçote (E para à esquerda ou D para à direita) e a escrita na fita baseado no simbolo atual na fita e estado atual da máquina.
5. $q_0 \in Q$ é o **estado inicial**.
6. $q_a \in Q$ é o **estado de aceitação**.
7. $q_r \in Q, q_a \neq q_r$ é o **estado de rejeição**.

Conforme a máquina computa, a fita, a posição do cabeçote e o estado da máquina sofrem alterações. Ao estado destes três componentes é dado o nome de **configuração**. Conforme Sipser (2013), diz-se que uma configuração C_1 **origina** C_2 se é possível a máquina ir de C_1 a C_2 em um único passo de computação, isto é, uma aplicação de uma função de transição. Uma MT M aceita uma sentença w se existir uma sequência de configurações $C_1, C_2, ..., C_k$ atendam à estas três condições:

1. C_1 é a configuração inicial de M sobre a entrada w,

2. cada C_i origina C_{i+1}, e

3. C_k é uma configuração de aceitação.

2.2.5.1 Variantes de Autômatos de Turing

1. **Máquina de Turing Não Determinística**: Variante com mesmo poder de reconhecimento (equivalente à uma MT), porém que pode estar em múltiplas configuração ao mesmo tempo e apresentar transições por ε;

2. **Máquinas de Turing Multi-fitas**: Variante com mesmo poder de reconhecimento (equivalente à uma MT), porém possui mais de uma fita, de número $k \geq 1$. Sua função de transição é dada por $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times E, D, P^k$.

3. **Autômato Linearmente Limitado**: Variante com menor poder de reconhecimento, sendo possível reconhecer apenas linguagens sensíveis ao contexto, isto é, geradas por gramáticas de tipo 1. A principal diferença desta variante é que sua fita possui tamanho fixo. Pelo tamanho limitado da fita, é possível enumerar todas as configurações existentes e por isso esta máquina possui parada garantida e é tal fator que reduz o conjunto de linguagens reconhecíveis.

2.2.6 Decidibilidade de Algoritmos de Comparação entre Máquinas

Tanto em Sipser (2013) quanto em Hopcroft, Motwani e Ullman (2001) há a discussão sobre a decidibilidade de comparar se duas máquinas, sendo elas de um mesmo modelo computacional, M_1 e M_2, reconhecem a mesma linguagem, isto é $L(M_1) = L(M_2)$. Como resultado da discussão, é dito que para Autômatos Finitos, o algoritmos de comparação existe e é bem definido, porém para Autômatos de Pilha e Autômatos de Turing não existe um algoritmo, visto que o problema é indecidível.

3 TRABALHOS CORRELATOS

Como qualquer projeto referente à adição de novas funcionalidades a ferramentas de terceiros, é necessário buscar documentação, exemplos e outros trabalhos envolvendo o que se busca implementar.

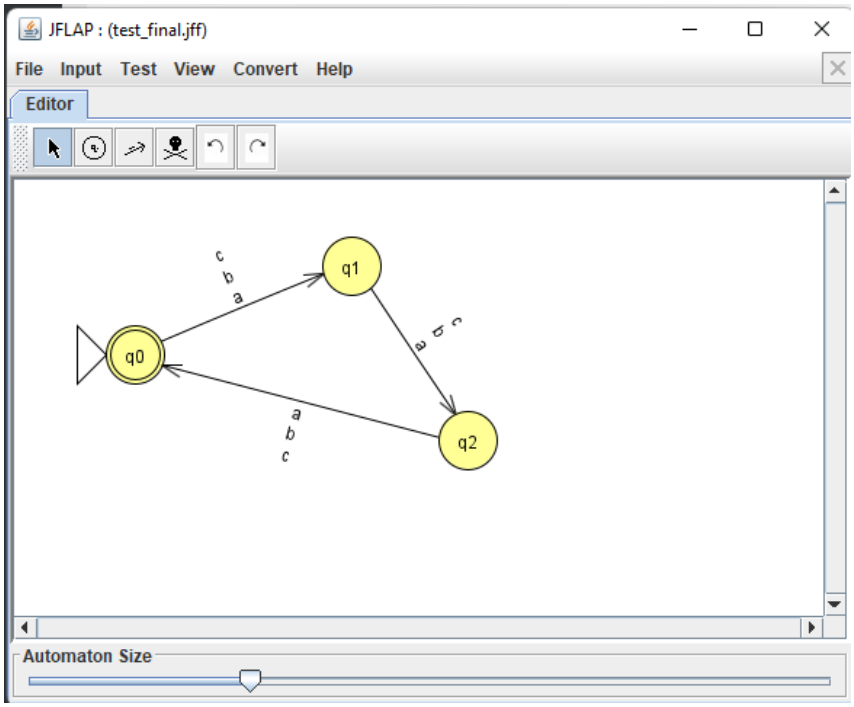
Como este trabalho visa implementar a avaliação automática de questões sobre autômatos, têm-se dois principais focos: avaliação de autômatos e avaliação automática no Moodle. Atualmente não existem ferramentas que implementam tal funcionalidade. Portanto, foram buscadas referências em cada um dos focos.

3.1 JFLAP

JFLAP (*Java Formal Languages and Automata Package*) é uma ferramenta educacional desenvolvida por Susan H. Rodge e publicada em 1996. Previamente, Rodge desenvolvera muitas ferramentas relacionada ao a teoria dos autômatos, sendo alguma delas o NPDA em 1992, o FLAP (*Formal Languages and Automata Package*) em 1993.

O JFLAP possui uma interface simples e fácil de utilizar (como é possível ver na Figura 7), contudo, devido à idade do programa, pouco reativa em comparação à softwares modernos. Outros problemas estão relacionados a usabilidade, tendo que o usuário, ao querer trocar de tipo de máquina, ter que fechar a tela do programa e voltar para o menu inicial.

Figura 7 – Interface do JFLAP 7



Fonte: *Próprio*

Além de permitir a criação de autômatos finitos, de pilha e de Turing, ambos determinísticos e não determinísticos, o JFLAP também permite a simulação destas máquinas. Através de uma árvore de execução na parte de baixo da tela é possível ver a execução da máquina, com sua entrada, pilha (se existir) e fita (se existir).

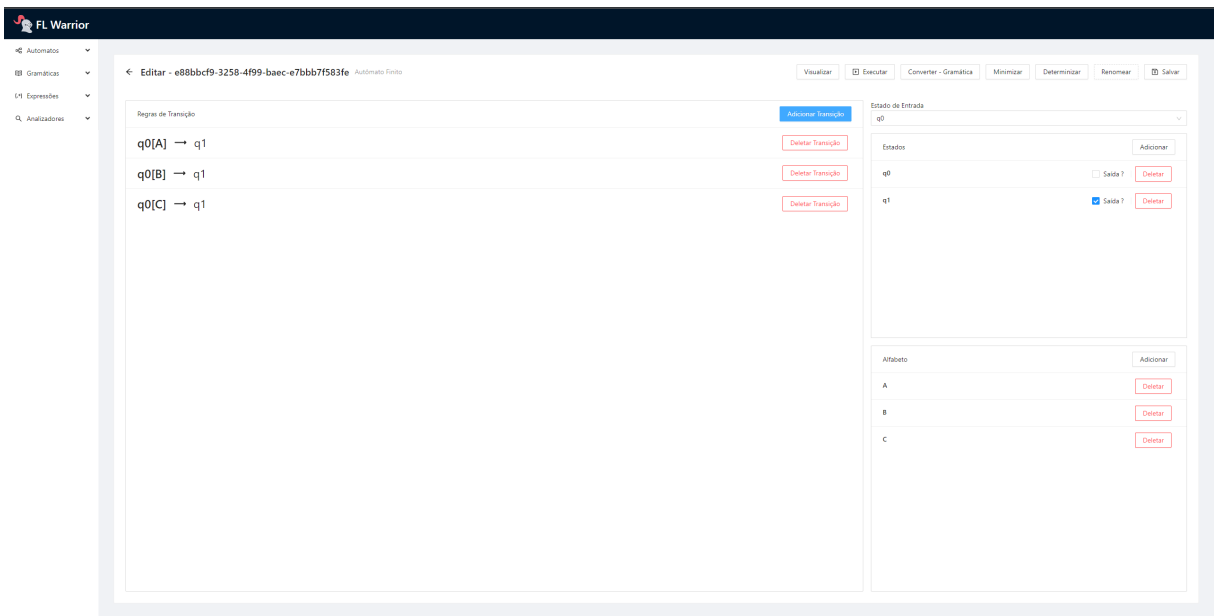
Por ser uma ferramenta *standalone*, o JFLAP está sujeito apenas às restrições que a linguagem e o sistema operacional impõem, podendo otimizar e arquitetar devidamente o comportamento do programa. Isso permite uma sólida orientação a objeto em seu código fonte, além da alta performance.

Uma característica interessante para este trabalho é a capacidade do JFLAP conseguir exportar um arquivo XML contendo a descrição da máquina. Isso nos permite interoperar com os arquivos de definição da ferramenta, conseguindo importar e executar as máquinas.

3.2 FLWARRIOR

Ferramenta desenvolvida por Enzo Coelho Albornoz e Arthur Philippi Bianco em 2021, inspirada no JFLAP, que permite a definição, execução e conversão entre máquinas finitas, gramáticas regulares e expressões regulares. Possui também operações de minimização e determinização de autômatos finitos, além da visualização da máquina. Uma das grandes vantagens dessa ferramenta é a interface simples, performática e reativa, como pode-se ver na Figura 8.

Figura 8 – Interface do FLWarrior



Fonte: *Próprio*

Uma outra vantagem é a exportação e importação de arquivos de texto que descrevem autômatos, aos moldes do JFLAP, permitindo a interoperabilidade com

outros sistemas. Diferentemente do JFLAP, o FLWarrior é uma ferramenta executada pelo navegador, sendo distribuído automaticamente pela internet, sem necessidade de download ou configuração do computador.

Outra contribuição grande desse trabalho está na definição de um modelo de tratamento de Autômatos, no quesito de execução e aceitação de sentenças. Isto é, o conhecimento gerado por essa ferramenta define os meios necessários para executarmos um modelo genérico de máquina no servidor do Moodle.

3.3 COMPARATIVO ENTRE FERRAMENTAS

Após analisar estas ferramentas, podemos traçar um comparativo entre elas. A principal diferença é que o plugin desenvolvido neste trabalho tem o grande diferencial de estar integrado ao Moodle, além disso, ele não refaz todos os mecanismos das outras ferramentas, mas utiliza definições prévias e reutilizando mecanismos internos das outras, como a interoperabilidade com o JFLAP através dos arquivos JFLAP 4.

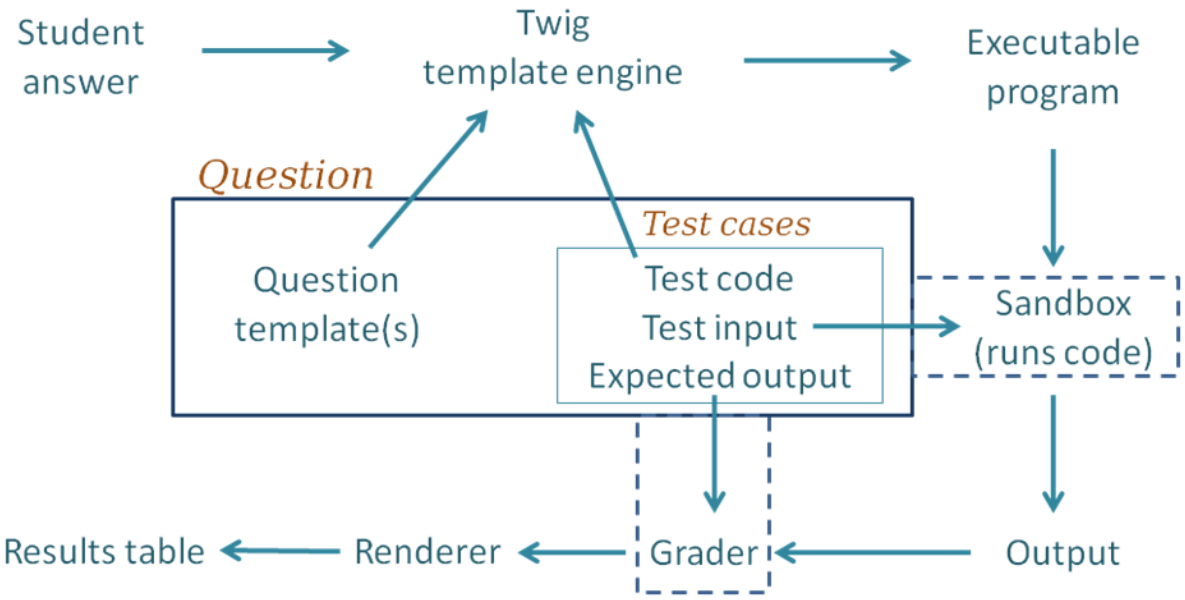
Ferramenta	**JFLAP**	**FLWarrior**	**QType-FLWarrior (Este Trabalho)**
Autômatos Finitos	✓	✓	✓
Autômatos de Pilha	✓	✗	✓
Máquinas de Turing	✓	✗	✓
Expressões Regulares	✓	✓	✗
Gramáticas	✓	Regulares	✗
Executado na Web	✗	✓	✓
Integrado ao Moodle	✗	✗	✓

3.4 CODE RUNNER

Code Runner é um plugin para o Moodle o qual permite a avaliação automática de códigos fonte. Desenvolvido por Richard Lobb da Universidade de Canterbury e Tim Hunt da *The Open University*, esta ferramenta vem sendo utilizada a mais de meia década em múltiplas disciplinas envolvendo programação na universidade de Canterbury. Segundo dados do autor da biblioteca, mais de 1800 cursos ao redor do mundo já utilizam o *Code Runner* em suas atividades (LOBB; HUNT, 2021).

O modo de avaliação automática é baseada em testes e tem seu esquemático na Figura 9. O professor, ao definir as questões, insere uma série da validações a serem executadas com o código do aluno. Após isso, quando o aluno insere sua resposta, o código é enviado via rede para uma ambiente virtualizado de execução de código, onde os testes são feitos e sua pontuação é recebida de volta no ambiente Moodle. Essa arquitetura é extremamente complexa. Embora tenha uma base de código imensa, sua performance é notória, sendo esse mais um ponto importante para a utilização dessa ferramenta.

Figura 9 – Esquema de avaliação do Code Runner

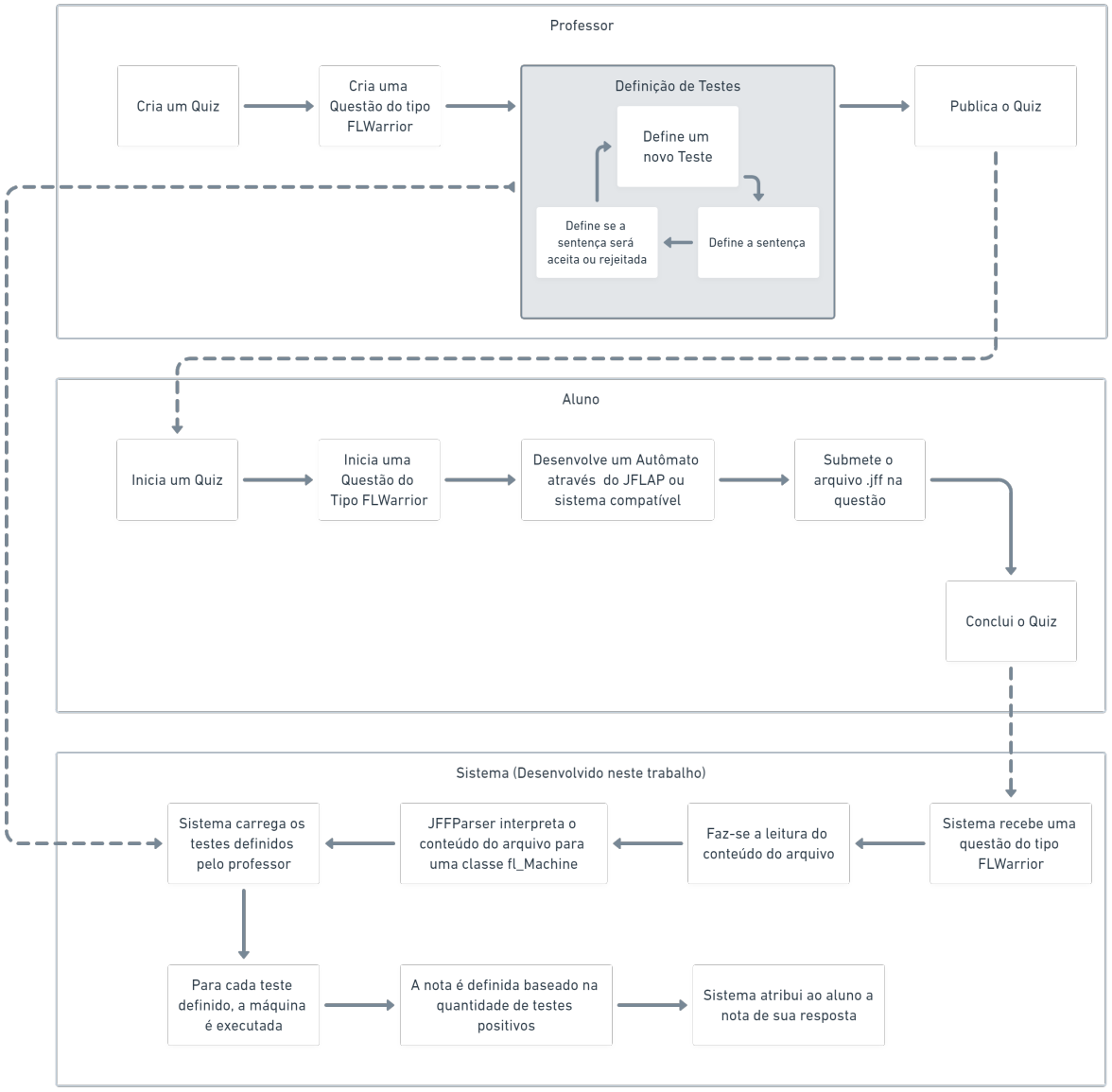


Fonte: Lobb e Hunt (2021)

4 DESENVOLVIMENTO DE UM PLUGIN PARA AUTÔMATOS

A abordagem de solução adotada envolve a possibilidade de interoperabilidade do JFLAP com o método de correção por teste do *Code Runner*. Para isso o professor poderá inserir uma série de palavras, indicando se deverão ser rejeitadas ou não pela máquina. O aluno, para responder a questão, poderá anexar uma máquina exportada diretamente do JFLAP. O plugin de avaliação automático interpretará a resposta do aluno e executará a maquina com as entradas disponibilizadas pelo professor. A pontuação do aluno na questão será computado pelo número de testes positivos. As máquinas suportadas neste trabalho serão os autômatos finitos, de pilha e de Turing, tanto em suas versões determinísticas quanto nas não determinísticas. Não haverá suporte ao caso de múltiplas pilhas ou múltiplas fitas. O Fluxo de correção previsto é definido na Figura 10.

Figura 10 – Visão de alto nível do processo de avaliação automático



Fonte: Próprio

Portanto, para o devido cumprimento dos objetivos deste trabalho, o *plugin* tem os seguintes requisitos:

- O professor deverá ser capaz de introduzir uma série de sentenças, durante a definição de uma questão, as quais devem ser aceitas ou rejeitadas pela máquina
- O aluno deve ser capaz de anexar um arquivo compatível (neste caso, um arquivo no formato JFLAP 4), o qual representará sua máquina e consequentemente sua resposta.
- O sistema deve armazenar estas informações e ao final do envio de um *quiz*, o sistema deve executar a máquina a fim de verificar os testes de aceitação

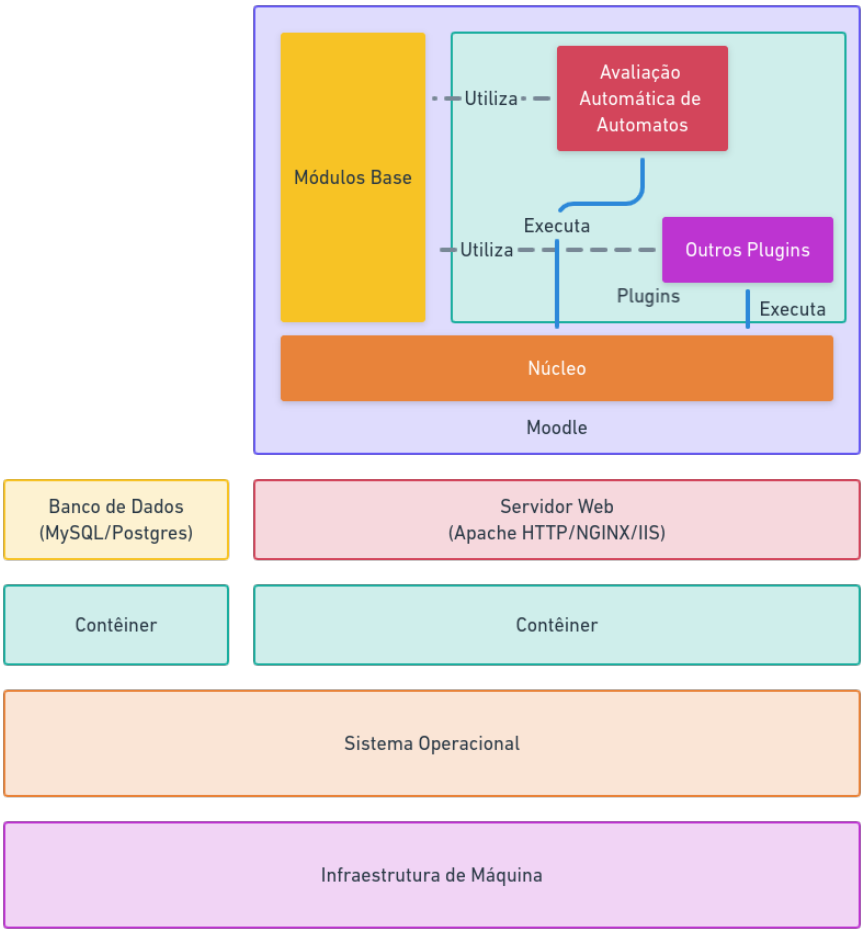
definidos pelo professor.

- O sistema deve atribuir uma nota relativa a quantidade de testes concluídos com sucesso pelo aluno.

4.1 DEFININDO O AMBIENTE DE DESENVOLVIMENTO

Para a avaliação automática das questões sobre Autômatos, é necessário desenvolver um módulo carregado dinamicamente pelo Moodle, isto é, um *plugin*. Tal *plugin* é carregado por outro módulo, já incluso, que gerencia e executa os plugins da plataforma. Na Figura 11 , é ilustrado onde estará localizada a contribuição deste projeto.

Figura 11 – Visão de alto nível da contribuição com este plugin



Fonte: Próprio

4.1.1 Padronização de Código

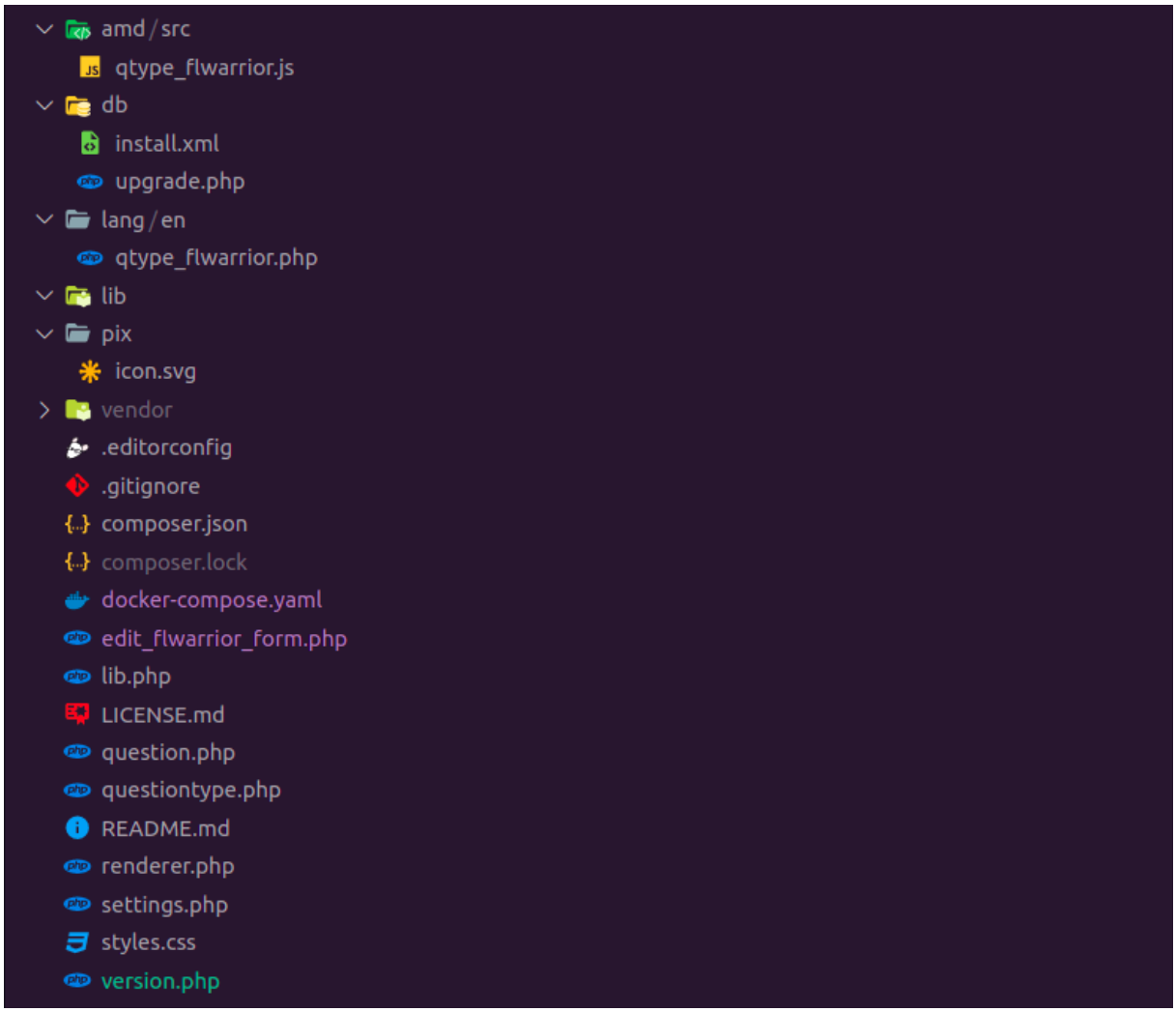
Seguindo o guia de desenvolvimento de plugins para o Moodle, é obrigatório seguir o padrão de nomenclatura e organização de código denominada *frankeinstyle*.

Para isso, é necessário importar o *Code Sniffer*, um utilitário para verificar o padrão de escrita de código. Seguindo o próprio guia, é possível concluir a instalação dele em editores de código como *VSCode* ou *PHPStorm* (MOODLE, 2020a).

4.1.2 Utilizando o Plugin Esqueleto

Podemos utilizar um plugin esqueleto disponibilizado pelo Moodle para iniciarmos o desenvolvimento do plugins, modificando as nomenclaturas para combinar com o tipo de plugin (qtype) e seu nome (flwarrior) conforme vemos na Figura 12. Os demais arquivos são agnósticos ao módulo pois são configurações que não interferem na execução ou arquivos previamente definidos no manual do Moodle.

Figura 12 – Pasta raiz do projeto contendo os arquivos básicos



Fonte: Próprio

4.1.3 Inicializando o Moodle

Para executarmos o plugin, é necessário ter o servidor web (Apache HTTP) executando o Moodle junto ao banco de dados, que neste caso é o MariaDB.

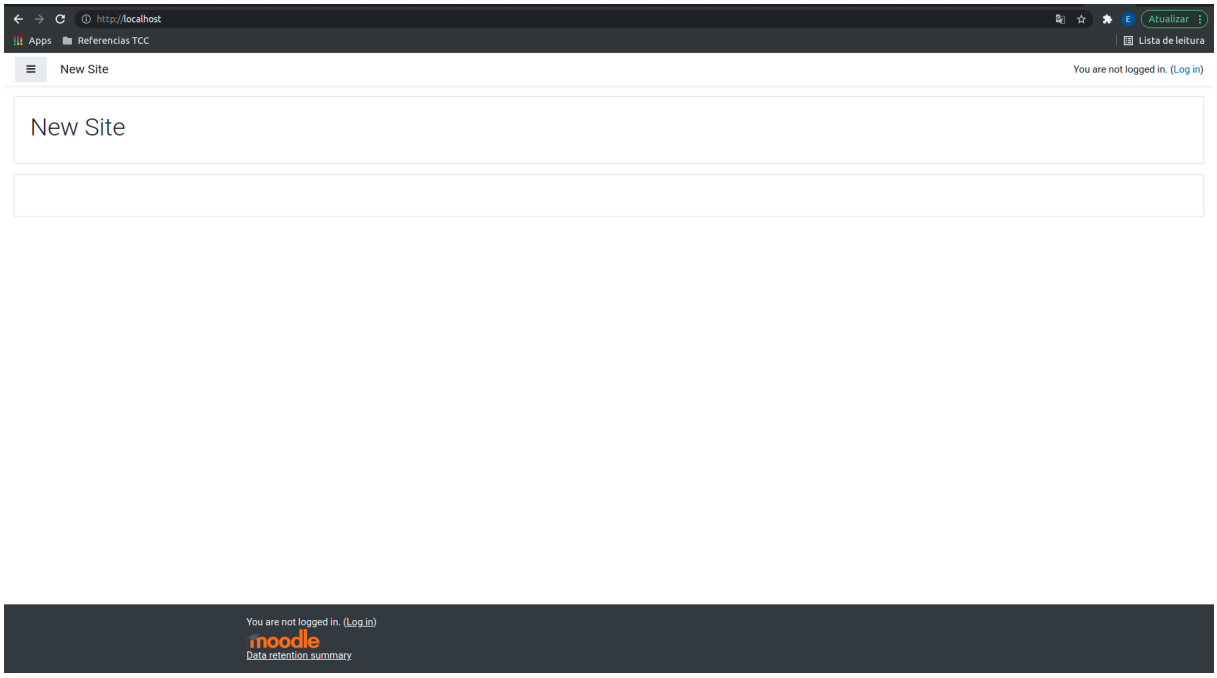
Para isso, serão criados dois contêineres, um para o banco de dados e outro para o servidor web e o Moodle. Conseguimos inicializá-los utilizando o arquivo mostrado na Figura B.1.1. Neste arquivo definimos quais imagens os contêineres irão executar (seção *image*). Definimos também as variáveis de ambiente (seção *environment*), mapeamento de portas (seção *ports*) e persistência de arquivos (seção *volumes*).

E executando o seguinte comando:

```
docker-compose -f "docker-compose.yml" up -d --build
```

Assim, como vemos na Figura 13, temos o Moodle sendo acessível a partir do navegador através do link *http://localhost:80/*

Figura 13 – Página inicial em uma nova instancia Moodle



Fonte: Próprio

4.1.4 Definindo o Banco de Dados

A fim de armazenar o testes definidos pelo professor em cada questão, precisamos adicionar uma tabela no banco de dados para persistir tal informação. Podemos modelar um teste em uma classe PHP e seu espelhamento em uma tabela XMLDB como visto na Figura 14.

Figura 14 – Abstração de teste para um autômato em PHP e em XMLDB

(a) Classe PHP

```php
class fl_machine_test
{
    public ?int $id;
    public string $word;
    public bool $should_match;
    public int $max_iterations;
    public ?int $question_id;
}
```

(b) Tabela em XMLDB

```xml
<TABLE NAME="qtype_flwarrior_tests" COMMENT="Stores question tests">
    <FIELDS>
      <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="true"/>
      <FIELD NAME="question_id" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="false"/>
      <FIELD NAME="word" TYPE="text" NOTNULL="true" SEQUENCE="false"/>
      <FIELD NAME="should_match" TYPE="int" LENGTH="1" NOTNULL="true" DEFAULT="1"
SEQUENCE="false"/>
      <FIELD NAME="max_iterations" TYPE="int" LENGTH="15" NOTNULL="true" DEFAULT="1000"
SEQUENCE="false"/>
    </FIELDS>
    <KEYS>
      <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
      <KEY NAME="question_id" TYPE="foreign" FIELDS="question_id" REFTABLE="question"
REFFIELDS="id"/>
    </KEYS>
</TABLE>
```

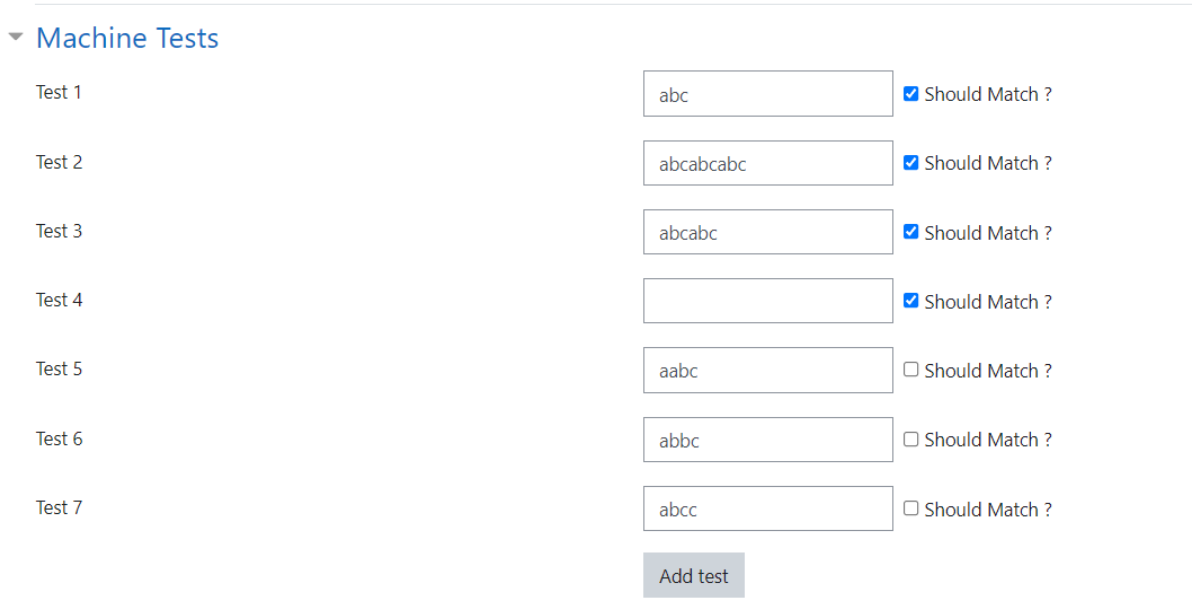
Fonte: Próprio

4.1.5 Desenvolvimento da Visão do Professor

4.1.5.1 Formulário de edição de questões

Inicialmente, para configurar um formulário de questão disponível para utilização pelo professor, é necessário alterar o arquivo *edit_flwarrior_form.php*. Este arquivo é responsável por construir a interface vista pelo docente ao criar uma nova questão. No caso deste trabalho, estendeu-se o formulário padrão a fim de adicionar a seção de *Machine Tests* (Testes de Máquina), como visto na Figura 15.

Figura 15 – Seção de definição de testes de máquina (no caso, uma máquina *M*, onde $L(M) = (abc)^n, n \geq 0$), presente ao criar ou editar uma questão



Fonte: Próprio

Para a inserção do segmento, é necessário sobrecarregar o método *definition_inner($mform)*, responsável pela adição de parâmetros extras na definição da questão. O parâmetro *$mform* possui o tipo *MoodleQuickForm* e é derivado da biblioteca *QuickForm*, apresentando métodos para a adição de listas, entradas, texto e botões.

A classe *question_edit_form* também apresenta o método *repeat_elements*, o qual permite o gerenciamento automático de listas dinâmicas, sem necessidade de uso de *Javascript* na interface gráfica.

Utilizando a biblioteca *QuickForm* e o método *repeat_elements*, foi adicionado o campo *machine-test-no* ao formulário. Este campo representa um teste a ser executado na máquina submetida pelo aluno e é composto pelos seguintes parâmetros:

1. ***machine-test-id-no***: Responsável por armazenar o identificador do teste no banco de dados.

2. ***machine-test-word-no***: Este parâmetro contém a sentença a qual será posta como entrada da máquina durante o teste.

3. ***machine-test-should-match-no***: Indica se a entrada da máquina deverá ser aceita ou não. Tanto a aceitação da máquina quanto este valor precisam ter o mesmo valor para a resposta ser avaliada como correta.

4.1.5.2 Persistência dos parâmetros da questão

Quando o editor da questão persiste as alterações, o método *save_question_options($question)* da classe *question_type* é invocado. Ao sobrecarregar este método é possível persistir alterações que não fazem parte do modelo padrão de questões, isto é, os parâmetros adicionais no formulário.

Para isto, utiliza-se a variável global *$DB*, cuja interface é definida na documentação, permitindo o desenvolvedor interagir com o banco de dados do Moodle e outras tabelas definidas pelo plugin via XMLDB.

Como complemento, na classe *question_type* são definidos os métodos de deleção da questão, permitindo a exclusão da questão, seus testes e arquivos associados, no caso de submissão de arquivos.

Também é implementado o método de recuperação destes parâmetros em *get_question_options($question)*, onde utilizando novamente a interface de acesso ao banco de dados, é possível recuperar os testes utilizando o identificador da questão.

4.1.6 Desenvolvimento da Visão do Aluno

4.1.6.1 Formulário de resposta de questão

Para a construção do formulário que o aluno responde, é preciso definir um arquivo *renderer.php*, cuja função é de apresentar tanto o texto da questão quanto os

campos de entrada da resposta. Isso se dá através do método *formulation_and_controls($qa, $options)*. Este método deve retornar um texto referente ao código HTML da página. Embora o Moodle apresente a classe *html_writer*, que permite construir páginas HTML através de chamada de método, contudo existe uma alternativa mais fácil através do *HEREDOC*. O *HEREDOC* é um modo de tratamento de strings integrado ao PHP que permite ao desenvolvedor a inserção de código HTML diretamente no código em PHP, sendo interpretado como um texto, sendo assim bem mais amigável para pessoas experientes com HTML.

Como o foco deste trabalho é a avaliação automática, utilizou-se como entrada o envio de um arquivo no formato JFLAP 4. Esse formato é simples e pode ser convertido a partir de outros formatos de descrição de autômatos. Para aceitar um arquivo, usou-se a classe *form_filemanager*, presente no Moodle.

A fim do Moodle detectar alterações no formulário e salvar o arquivo precisamos criar um identificador de rascunho através da função *$qa->prepare_response_files_draft_itemid($name, $context_id)*, onde os parâmetros indicam, respectivamente, o nome do atributo no formulário e o identificador do contexto da questão. O uso desta função pode ser melhor compreendido na função *files_input* do Apêndice B.2.1. O parâmetro *$name* também é necessário estar no retorno da função *question_type::response_file_areas()*.

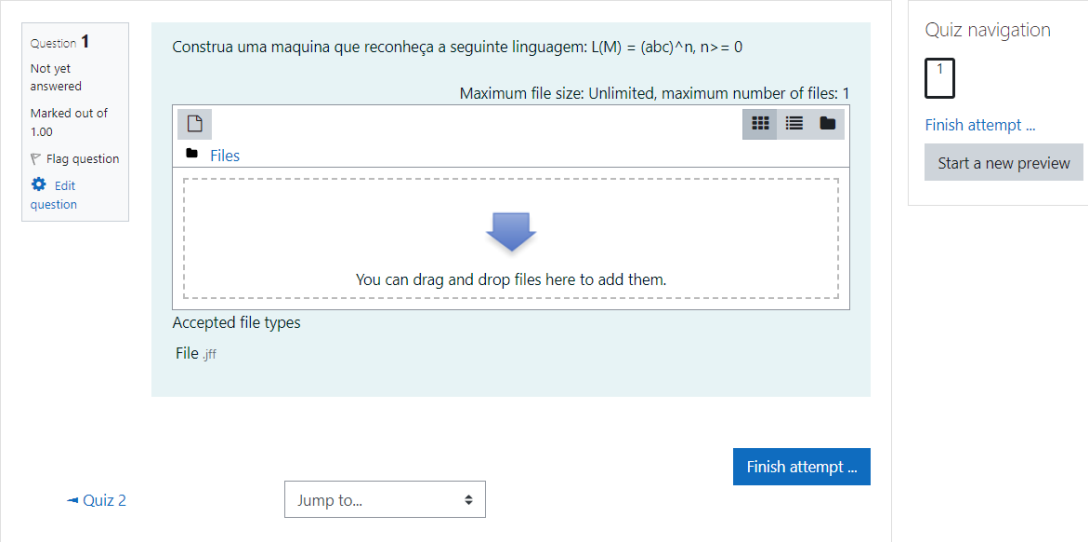
4.1.7 Persistência dos valores respondidos pelo aluno

Como foi utilizado o sistema de arquivos, o Moodle automaticamente se responsabiliza por armazenar e carregar os arquivos enviados pelo aluno. Contudo, é necessário implementar uma série de funções para gerenciar o comportamento da questão. A primeira dela é verificar se a questão foi alterada (através da função *question_graded_automatically::is_same_response*) e precisa ser reescrita no banco. Para isso, verifica-se o conteúdo dos arquivos enviados onde, sendo diferentes, será chamada a rotina de armazenamento do arquivo. Isso economiza recursos e permite que o servidor tenha mais vazão de comandos.

Outra função importante a ser implementada é a *question_graded_automatically::is_complete_response*, onde deve-se verificar se a resposta do aluno é adequada para ser enviada para avaliação. Essa função foi sobrecarregada e verifica se é possível interpretar uma máquina a partir do arquivo enviado. Em caso de sucesso, habilita-se o envio final da questão para correção.

A implementação das funções citadas nestas seção podem ser encontradas no Apêndice B.2.2. Como resultado, tem-se na Figura 16 a interface gráfica da questão para o estudante.

Figura 16 – Formulário de resposta visualizado pelo aluno



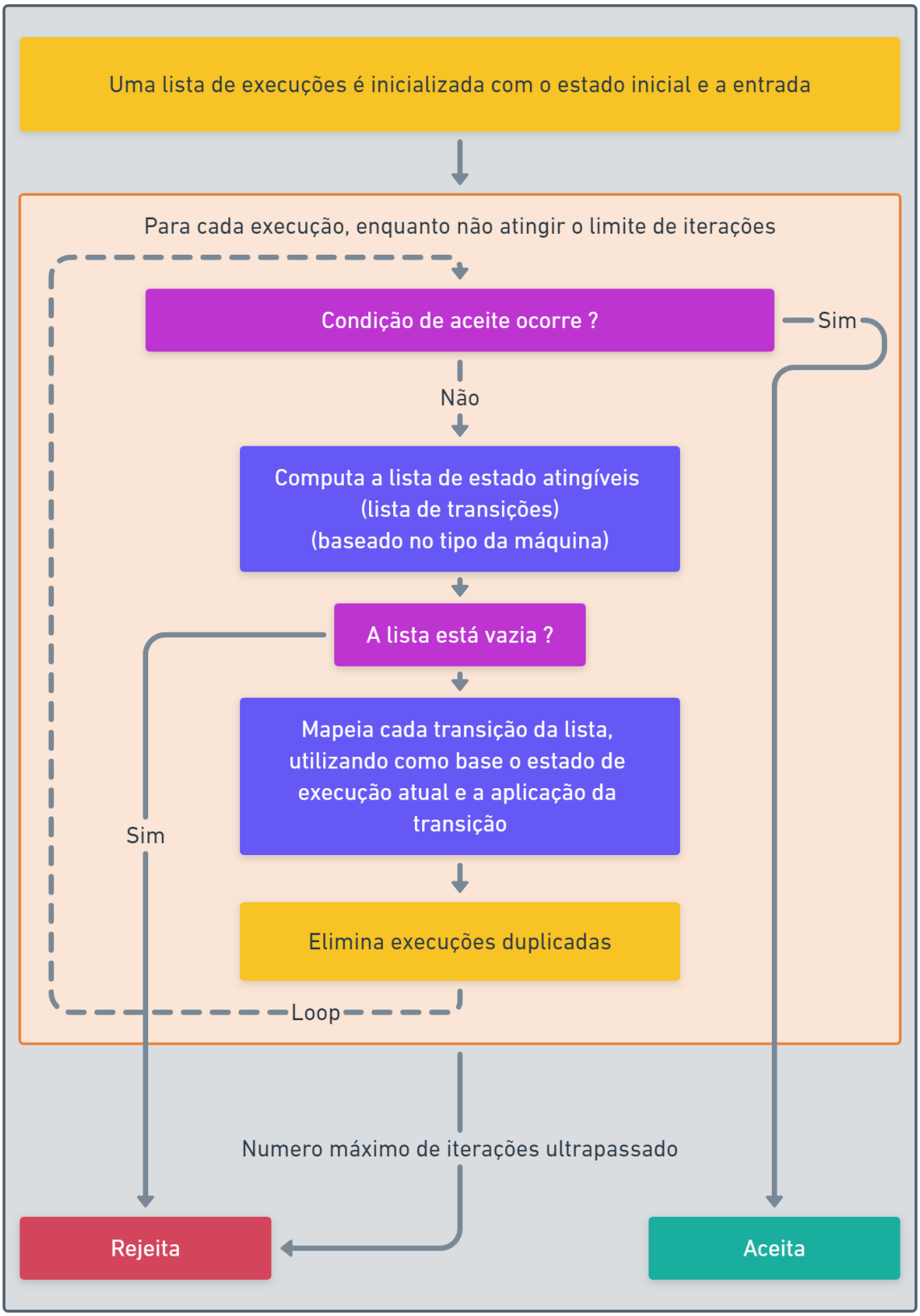
Fonte: Próprio

4.1.8 Desenvolvimento da Avaliação Automática

Após a submissão da resposta, o motor interno do Moodle executa o método *question_graded_automatically::grade_response($response)*. Neste método, é recebido como parâmetro um objeto contendo os campos enviados no formulário do aluno. Dentro deste objeto está presente o campo *machine*. Com ele, conseguimos obter o conteúdo do arquivo em formato de texto. A partir deste textos, é possível utilizar um analisador para interpretar o conteúdo e instanciar classes que representam uma máquina. Neste caso foi desenvolvida a classe *JFFParser*, cujo objetivo é fazer a análise de arquivos de tipo JFLAP 4 e retornar uma abstração de máquina, sendo do tipo *fl_machine*.

Caso haja sucesso na interpretação da máquina, executam-se os testes um a um, com o algoritmo de execução visto no Apêndice B.2.2 e no Apêndice B.3.6. A visão em alto nível deste algoritmo é vista na Figura 17. Para a execução da máquina, utilizamos um algoritmo de busca em largura na árvore de computação. Partimos de uma lista de "*threads*"de execução (uma estrutura contendo o estado da máquina, entrada restante, e fita ou pilha caso necessário), contendo inicialmente o estado inicial e a sentença testada. Então é verificada a condição de aceitação. Caso não esteja em condição de aceite a máquina tentará mapear a lista de "*threads*", aplicando as possíveis transições, num limite máximo de iterações para evitar que o sistema entre em *loop* ou tome muitos recursos do servidor.

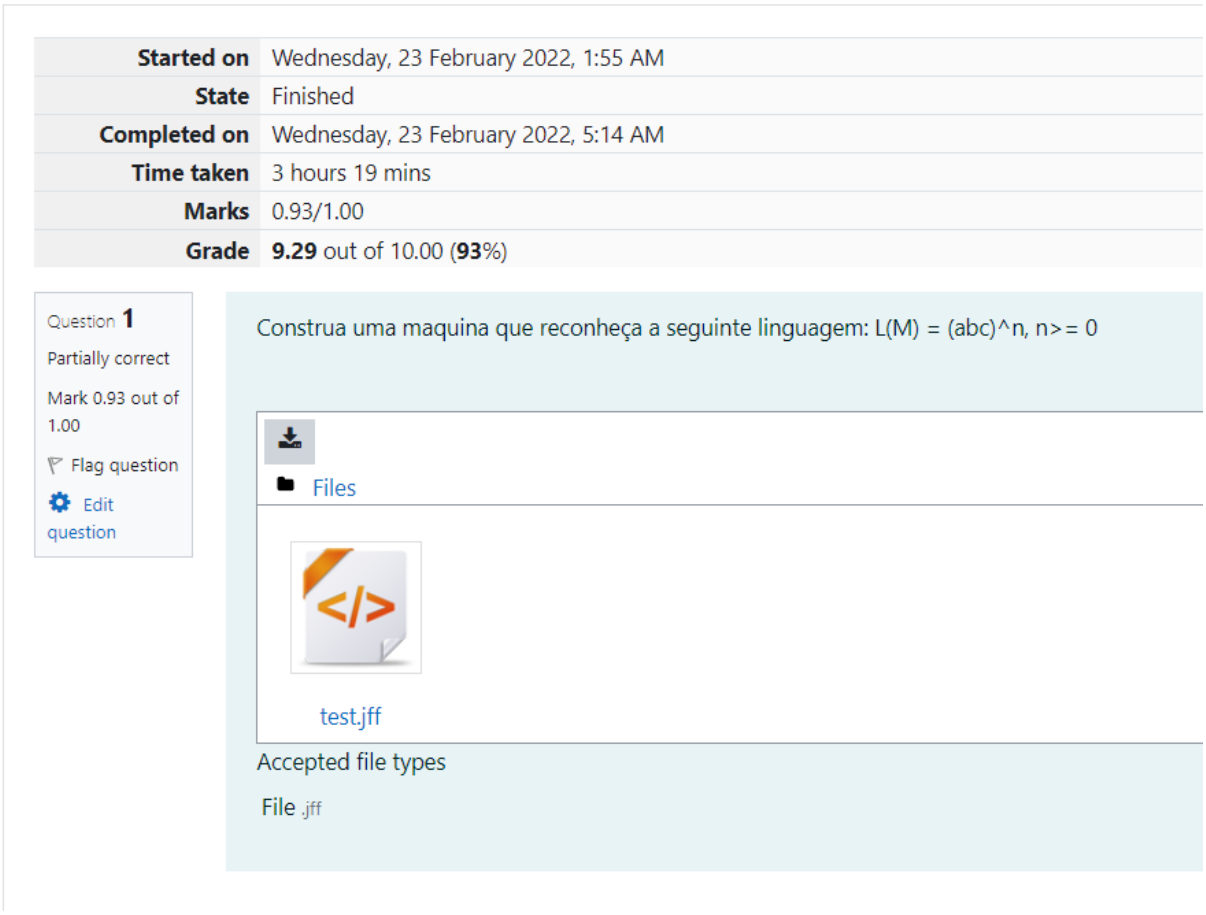
Figura 17 – Visão geral do algoritmo de execução de autômatos



Fonte: Próprio

Contendo o número de testes concluídos com sucesso, pode-se atribuir uma nota final à resposta, sendo ela retornada ao final da função *grade_response*. A nota é então visualizada pelo aluno na tela de visualização do *quiz*, como visto na Figura 17.

Figura 18 – Visualização do aluno da nota atribuída à questão pela avaliação automática, onde obteve-se uma nota parcial



Fonte: Próprio

Esta nota fica então disponível ao aluno para futura visualizações, podendo ele efetuar o *download* do arquivo para análise, isto é, conseguir reutilizar no JFLAP o arquivo enviado no *quiz*.

5 GUIA DE USO

5.1 GUIA PARA INTEGRAÇÃO AO MOODLE

Nesse documento, é apresentado os passos necessários para a adição de um plugin ao Moodle. Para o desenvolvimento de tal informativo, usou-se como base o Guia de Instalação de *Plugins* do Moodle Moodle (2021b). O guia pode ser encontrado no Apêndice C.1.

5.2 GUIA PARA DISCENTES E DOCENTES

Este pequeno manual exemplifica, através de texto e imagens, os passos que os docentes precisam fazer para:

1. Criar um novo *quiz*;
2. Adicionar uma questão do tipo FLWarrior, para avaliação automática de Autômatos;
3. Definir os testes de máquina;
4. Publicar o *quiz*.

Para a seção destinada aos alunos, o guia apresenta as seguintes etapas:

1. Iniciar a resposta a um *quiz*;
2. Exportar uma máquina do JFLAP para um arquivo JFLAP 4;
3. Anexar a máquina à resposta da questão;
4. Submeter a questão para avaliação.

O manual foi desenvolvido utilizando a linguagem Markdown e através da ferramenta Grip[1], gerou-se um documento no formato PDF. Este documento está presente nos Apêndices **??** e C.2.

[1] Disponível em: https://pypi.org/project/grip/

6 CONSIDERAÇÕES FINAIS

6.1 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvido um maior entendimento sobre o processo de criação e execução de *plugins* na plataforma Moodle. Embora outros *plugins* existam e haja uma vasta lista de documentos descrevendo as classes do AVA, não existem guia explicativos que descrevam o processo completo e o fluxo de integração dessas partes. Portanto, ao definir os métodos e classes necessárias para questões de terceiros, atingiu-se o objetivo de compreender a programação do Moodle e a extensão de seu comportamento.

Conforme visto na seção 4, houve sucesso no objetivo de construção de um tipo de questão que automaticamente avalia autômatos. Como explicado na seção 2, não é possível implementar um método formal de avaliação, sendo necessário utilizar métodos que outras ferramentas utilizam, como visto no seção 3.

Em relação ao guias, uma vez mapeado todo o fluxo de comportamento da questão, cumpriram-se os objetivos de construção de manuais explicando tanto a instalação no Moodle, quanto o seu uso no dia a dia por professores e alunos. Houve sucesso também no objetivo de publicação deste *plugin* em um repositório aberto[1], ficando disponível para que a comunidade o utilize e quiçá implementem novas funcionalidades.

6.2 TRABALHOS FUTUROS

Embora cumpridos os objetivos, um teste com professores e alunos não pode ser feito, pois, agravados pela alteração do calendário acadêmico na pandemia de 2021, ocorreram vários choques de horários, dificultando uma implementação. Portanto, em um passo seguinte, seria útil a realização de um teste público, coletando entrevistas dos alunos e professores, para uma melhoria na experiência de usuário.

Outro aspecto também relacionado a experiência de usuário seria a adição de suporte a outros formatos de arquivos que descrevem máquinas, como o presente no FLWarrior. Vinculado a isto, pode-se colocar como visível os testes que deram sucesso ou falha na resposta do aluno (com visibilidade configurada pelo professor). Também seria útil a inclusão de um sistema gerador de sentenças para o professor utilizar.

Outra estratégia para a melhoria da usabilidade seria o desenvolvimento de uma interface gráfica acoplada à questão, gerando um arquivo compatível que possa ser submetido.

Além disso, o algoritmo de execução de máquinas pode ser bastante otimizado se feito uma revisão de complexidade do algoritmo ou até mesmo pela implementação do algoritmo em *WebAssembly* (padrão binário executável em navegadores de internet).

---

[1] Disponível em: https://github.com/EnzoAlbornoz/qtype_flwarrior

REFERÊNCIAS

ASCOM, Ministério das Comunicações. **Pesquisa mostra que 82,7% dos domicílios brasileiros têm acesso à internet**. 2021. Disponível em: <https://www.gov.br/mcom/pt-br/noticias/2021/abril/pesquisa-mostra-que-82-7-dos-domicilios-brasileiros-tem-acesso-a-internet>. Acesso em: 10 out. 2021. Citado na p. 13.

BAUER, Dominique. **Formulas question type for Moodle**. 2021. Disponível em: <https://github.com/dbauer-ets/moodle-qtype_formulas>. Acesso em: 10 out. 2021. Citado na p. 22.

BRI, Diana *et al.* A study of virtual learning environments. **WSEAS Transactions on Advances in Engineering Education**, Citeseer, v. 6, n. 1, p. 33–43, 2009. Citado na p. 16.

CGI. 2021. Disponível em: <https://pt.wikipedia.org/wiki/CGI>. Acesso em: 10 out. 2021.

CHOMSKY, N. Three models for the description of language. **IEEE Transactions on Information Theory**, Institute of Electrical e Electronics Engineers (IEEE), v. 2, n. 3, p. 113–124, set. 1956. DOI: 10.1109/tit.1956.1056813. Disponível em: <https://doi.org/10.1109/tit.1956.1056813>. Citado na p. 24.

CONVERSE, Tim; PARK, Joyce. **PHP: a bíblia**. [*S.l.*]: Gulf Professional Publishing, 2003. Acesso em: 10 out. 2021. Citado na p. 20.

DOCKER. **Docker overview**. 2021. Disponível em: <https://docs.docker.com/get-started/overview/>. Acesso em: 10 out. 2021. Citado na p. 19.

DOCKER. **What is a Container?** 2021. Disponível em: <https://www.docker.com/resources/what-container>. Acesso em: 10 out. 2021. Citado na p. 18.

ELMASRI, Ramez; NAVATHE, Shamkant B; PINHEIRO, Marília Guimarães *et al.* Sistemas de banco de dados. Pearson Addison Wesley São Paulo, 2005. Acesso em: 10 out. 2021. Citado na p. 20.

FIELDING, Roy T.; KAISER, Gail. The Apache HTTP server project. **IEEE Internet Computing**, IEEE, v. 1, n. 4, p. 88–90, 1997. Acesso em: 10 out. 2021. Citado na p. 20.

FINKELSTEIN, J. **A graphical representation of the sets of languages included in the Chomsky hierarchy.** 2010. Disponível em: <https://en.wikipedia.org/wiki/File:Chomsky-hierarchy.svg>. Citado na p. 24.

GNU. **A Quick Guide to GPLv3 - GNU Project - Free Software Foundation**. 2021. Disponível em: <https://www.gnu.org/licenses/quick-guide-gplv3.html>. Acesso em: 10 out. 2021. Citado na p. 17.

GOIÁS, Insituto Federal de. **Ambiente virtual de Ensino e Aprendizagem (avea)**. 2018. Disponível em: <http://guiaead.ifg.edu.br/wiki/index.php/Ambiente_Virtual_de_Ensino_e_Aprendizagem_(AVEA)>. Acesso em: 10 out. 2021. Citado na p. 13.

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. **Introduction to automata theory, languages, and computation, 2nd Edition.** [*S.l.*]: Addison-Wesley-Longman, 2001. P. i–xiv, 1–521. (Addison-Wesley series in computer science). ISBN 978-0-201-44124-6. Citado nas pp. 23, 25, 26, 29.

LOBB, Richard; HUNT, Tim. **Trampgeek/moodle-qtype_coderunner: CodeRunner: A Moodle quiz question type that runs student-submitted program code in a sandbox to check if it satisfies a given set of tests.** Disponível em: <https://github.com/trampgeek/moodle-qtype_coderunner>. Acesso em: 6 dez. 2021. Citado nas pp. 32, 33.

MILLIGAN, Colin. The role of virtual learning environments in the online delivery of staff development. Report 2: Delivering Staff and Professional Development Using Virtual Learning Environments, 1999. Citado na p. 16.

MOODLE. **Developer Documentation**. 2021. Disponível em: <https://docs.moodle.org/dev/Main_Page>. Acesso em: 10 out. 2021.

MOODLE. **Frankenstyle**. 2020. Disponível em: <https://docs.moodle.org/dev/Frankenstyle>. Acesso em: 10 out. 2021. Citado na p. 37.

MOODLE. **Installing Plugins**. 2021. Disponível em: <https://docs.moodle.org/310/en/Installing_plugins>. Acesso em: 10 out. 2021. Citado na p. 45.

MOODLE. **Plugins types**. 2021. Disponível em: <https://docs.moodle.org/dev/Plugin_types>. Acesso em: 10 out. 2021.

MOODLE. **Question types**. 2020. Disponível em: <https://docs.moodle.org/dev/Question_types>. Acesso em: 10 out. 2021.

MOODLE. **XMLDB introduction**. 2020. Disponível em: <https://docs.moodle.org/dev/XMLDB_introduction>. Acesso em: 10 out. 2021. Citado nas pp. 20, 21.

MOUAT, Adrian. **Using Docker: Developing and Deploying Software with Containers**. [*S.l.*]: " O'Reilly Media, Inc.", 2015. Acesso em: 10 out. 2021. Citado na p. 18.

PHP. [*S.l.*]: Wikimedia Foundation, 2021. Disponível em: <https://pt.wikipedia.org/wiki/PHP>. Acesso em: 10 out. 2021.

SIPSER, Michael. **Introduction to the Theory of Computation**. Third. Boston, MA: Course Technology, 2013. ISBN 113318779X. Citado nas pp. 25–29.

SOBRE o Moodle. 2021. Disponível em: <https://docs.moodle.org/all/pt_br/Sobre_o_Moodle>. Acesso em: 10 out. 2021.

TOP-DOWN Design (Introduction to Statistical Computing). 2012. Disponível em: <http://bactra.org/weblog/950.html>. Acesso em: 10 out. 2021.

Apêndices

APÊNDICE A – ARTIGO

Avaliação automática de questões sobre Autômatos na plataforma Moodle

Enzo C. Albornoz[1]

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

enzo.c.albornoz@grad.ufsc.br

Abstract. *The purpose of this project is to develop a system for automatic evaluation of questions about machines and formal languages in the Moodle virtual teaching and learning environment. For this, a PHP plugin will be developed and integrated with Moodle. The machine model will follow a model derived from JFLAP, but adapted for integration with MySQL database. The main application of this project will be to improve the model for questions that address this subject while decreasing the time to obtain corrections.*

Resumo. *O intuito deste projeto é desenvolver um sistema de avaliação automática de questões sobre Máquinas e Linguagens Formais no ambiente virtual de ensino e aprendizagem educacional Moodle. Para isso, será desenvolvido um plugin em PHP que será integrado ao Moodle. O modelo de máquinas seguirá como base um modelo derivado do JFLAP, porém adaptado para integração a banco de dados MySQL. A principal aplicação deste projeto será melhorar o modelo de questões que abordam este assunto enquanto diminui o tempo para obter as correções.*

1. Introdução

Os Ambientes Virtuais de Aprendizagem (AVAs) são softwares que, disponibilizados na internet, agregam ferramentas para a criação, tutoria e gestão de atividades que normalmente se apresentam na forma de cursos. Sendo constituídos a partir do uso de diferentes mídias e linguagens, a intenção é proporcionar não só a disponibilização de conteúdos, mas principalmente plena interatividade entre pessoas e grupos, viabilizando, por consequência, a construção do conhecimento [da Silva 2016].

O Moodle (*Modular Object-Oriented Dynamic Learning Environment*), lançado em 2002, é um dos AVAs mais conhecidos mundialmente. Dentre seus diferenciais está a sua licença GPL 3, publicando o código fonte da plataforma e permitindo que o Moodle possa ser utilizado gratuitamente. Também está presente no Moodle um sistema de *plugins* que permite a expansão de seu comportamento, adequando-se a novas necessidades. Tais diferenciais permitiram que milhares de instituições ao redor do mundo utilizassem tal ferramenta [Moodle 2021].

Por ser uma ferramenta tão abrangente e generalista, é notório que o Moodle não consiga abordar com perfeição todos os seus casos de uso. Muitos educadores, por lecionarem disciplinas com conteúdo complexo, não conseguem alcançar grande agilidade na correção de atividades.

Isso acontece pois o modelo de atividades usado por grande parte dos professores não consegue ser devidamente aplicada ao sistema de questões com avaliação automática do Moodle, tendo de serem manualmente corrigidas pelo docente e portanto, atrasando a correção. Isto se complica ainda mais no caso do ensino de teoria dos autômatos, visto que a correção de uma máquina pode rapidamente necessitar de um cálculo exponencial de estados de máquina. Este atraso acaba por influenciar negativamente o ensino, pois muitas vezes o aluno não consegue esclarecer a tempo suas dúvidas sobre uma correção ou assunto abordado, levando à piora na absorção do conteúdo.

2. Objetivos

2.1. Geral

Elaborar e disponibilizar um plugin para o ambiente virtual de aprendizagem Moodle que permita a avaliação automática de questões sobre Autômatos, no escopo de reconhecimento de linguagens, com a finalidade de permitir agilidade nas correções e no retorno dos professores aos os alunos.

2.2. Específicos

- Compreender a programação do Moodle e como expandir seu comportamento através de plugins;
- Implementar um plugin para avaliação automática de Autômatos (Finitos, Pilha e Turing) através do reconhecimento de linguagens baseado em testes de aceitação;
- Desenvolver um guia para a integração do plugin com Moodle UFSC;
- Desenvolver um guia de utilização do plugin tanto para alunos quanto para professores;
- Publicar o plugin em um repositório aberto para que a comunidade acadêmica o utilize.

3. Trabalhos Correlatos

Como qualquer projeto referente à adição de novas funcionalidades a ferramentas de terceiros, é necessário buscar documentação, exemplos e outros trabalhos envolvendo o que se busca implementar.

Como este trabalho visa implementar a avaliação automática de questões sobre autômatos, têm-se dois principais focos: avaliação de autômatos e avaliação automática no Moodle. Atualmente não existem ferramentas que implementam tal funcionalidade. Portanto, foram buscadas referências em cada um dos focos.

3.1. JFLAP

JFLAP (*Java Formal Languages and Automata Package*) é uma ferramenta educacional desenvolvida por Susan H. Rodge e publicada em 1996. Previamente, Rodge desenvolvera muitas ferramentas relacionada ao a teoria dos autômatos, sendo alguma delas o NPDA em 1992, o FLAP (*Formal Languages and Automata Package*) em 1993 [Rodger 2005].

O JFLAP possui uma interface simples e fácil de utilizar , contudo, devido à idade do programa, pouco reativa em comparação à softwares modernos. Outros problemas estão relacionados a usabilidade, tendo que o usuário, ao querer trocar de tipo de máquina, ter que fechar a tela do programa e voltar para o menu inicial.

Além de permitir a criação de autômatos finitos, de pilha e de Turing, ambos determinísticos e não determinísticos, o JFLAP também permite a simulação destas máquinas. Através de uma árvore de execução na parte de baixo da tela é possível ver a execução da máquina, com sua entrada, pilha (se existir) e fita (se existir).

Por ser uma ferramenta *standalone*, o JFLAP está sujeito apenas às restrições que a linguagem e o sistema operacional impõem, podendo otimizar e arquitetar devidamente o comportamento do programa. Isso permite uma sólida orientação a objeto em seu código fonte, além da alta performance. Uma característica interessante para este trabalho é a capacidade do JFLAP conseguir exportar um arquivo XML contendo a descrição da máquina. Isso nos permite interoperar com os arquivos de definição da ferramenta, conseguindo importar e executar as máquinas.

3.2. FLWarrior

Ferramenta desenvolvida por Enzo Coelho Albornoz e Arthur Philippi Bianco em 2021, inspirada no JFLAP, que permite a definição, execução e conversão entre máquinas finitas, gramáticas regulares e expressões regulares. Possui também operações de minimização e determinização de autômatos finitos, além da visualização da máquina. Uma das grandes vantagens dessa ferramenta é a interface simples, performática e reativa [Albornoz and Bianco 2021].

Uma outra vantagem é a exportação e importação de arquivos de texto que descrevem autômatos, aos moldes do JFLAP, permitindo a interoperabilidade com outros sistemas. Diferentemente do JFLAP, o FLWarrior é uma ferramenta executada pelo navegador, sendo distribuído automaticamente pela internet, sem necessidade de download ou configuração do computador.

Outra contribuição grande desse trabalho está na definição de um modelo de tratamento de Autômatos, no quesito de execução e aceitação de sentenças. Isto é, o conhecimento gerado por essa ferramenta define os meios necessários para executarmos um modelo genérico de máquina no servidor do Moodle.

3.3. Comparativo entre ferramentas

Após analisar estas ferramentas, podemos traçar um comparativo entre elas. A principal diferença é que o plugin desenvolvido neste trabalho tem o grande diferencial de estar integrado ao Moodle, além disso, ele não refaz todos os mecanismos das outras ferramentas, mas utiliza definições prévias e reutilizando mecanismos internos das outras, como a interoperabilidade com o JFLAP através dos arquivos JFLAP 4.

Ferramenta	**JFLAP**	**FLWarrior**	**QType-FLWarrior (Este Trabalho)**
Autômatos Finitos	✓	✓	✓
Autômatos de Pilha	✓	✗	✓
Máquinas de Turing	✓	✗	✓
Expressões Regulares	✓	✓	✗
Gramáticas	✓	Regulares	✗
Executado na Web	✗	✓	✓
Integrado ao Moodle	✗	✗	✓

3.4. Code Runner

Code Runner é um plugin para o Moodle o qual permite a avaliação automática de códigos fonte. Desenvolvido por Richard Lobb da Universidade de Canterbury e Tim Hunt da *The Open University*, esta ferramenta vem sendo utilizada a mais de meia década em múltiplas disciplinas envolvendo programação na universidade de Canterbury. Segundo dados do autor da biblioteca, mais de 1800 cursos ao redor do mundo já utilizam o *Code Runner* em suas atividades [Lobb 2021].

O modo de avaliação automática é baseada em testes. O professor, ao definir as questões, insere uma série da validações a serem executadas com o código do aluno. Após isso, quando o aluno insere sua resposta, o código é enviado via rede para uma ambiente virtualizado de execução de código, onde os testes são feitos e sua pontuação é recebida de volta no ambiente Moodle. Essa arquitetura é extremamente complexa. Embora tenha uma base de código imensa, sua performance é notória, sendo esse mais um ponto importante para a utilização dessa ferramenta.

4. Desenvolvimento de um Plugin para Autômatos

Para a avaliação automática das questões sobre Autômatos, é necessário desenvolver um módulo carregado dinamicamente pelo Moodle, isto é, um *plugin*. Tal *plugin* é carregado por outro módulo, já incluso, que gerencia e executa os plugins da plataforma. Na Figura 1 , é ilustrado onde estará localizada a contribuição deste projeto.

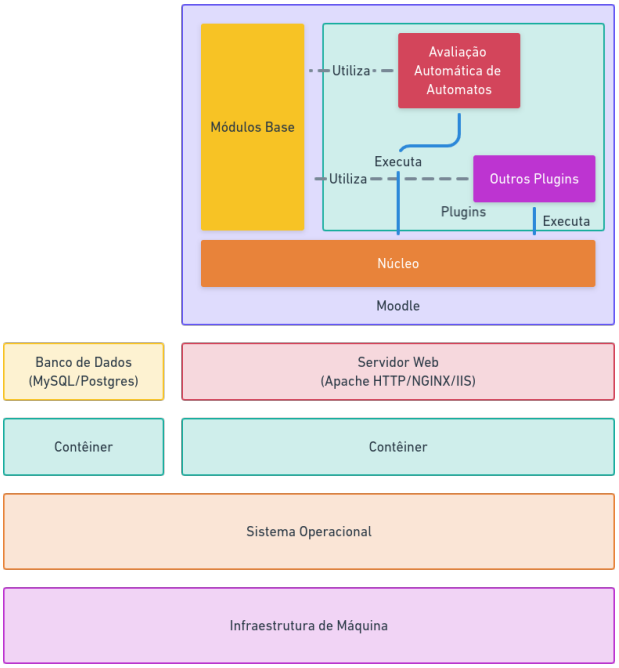


Figura 1. Visão de alto nível da contribuição com este plugin

4.1. Estrutura do *Plugin*

A abordagem de solução adotada envolve a possibilidade de interoperabilidade do JFLAP com o método de correção por teste do *Code Runner*. Para isso o professor poderá inserir uma série de palavras, indicando se deverão ser rejeitadas ou não pela máquina. O aluno, para responder a questão, poderá anexar uma máquina exportada diretamente do JFLAP. O plugin de avaliação automático interpretará a resposta do aluno e executará a maquina com as entradas disponibilizadas pelo professor. A pontuação do aluno na questão será computado pelo número de testes positivos. As máquinas suportadas neste trabalho serão os autômatos finitos, de pilha e de Turing, tanto em suas versões determinísticas quanto nas não determinísticas. Não haverá suporte ao caso de múltiplas pilhas ou múltiplas fitas. O Fluxo de correção previsto é definido na Figura 2

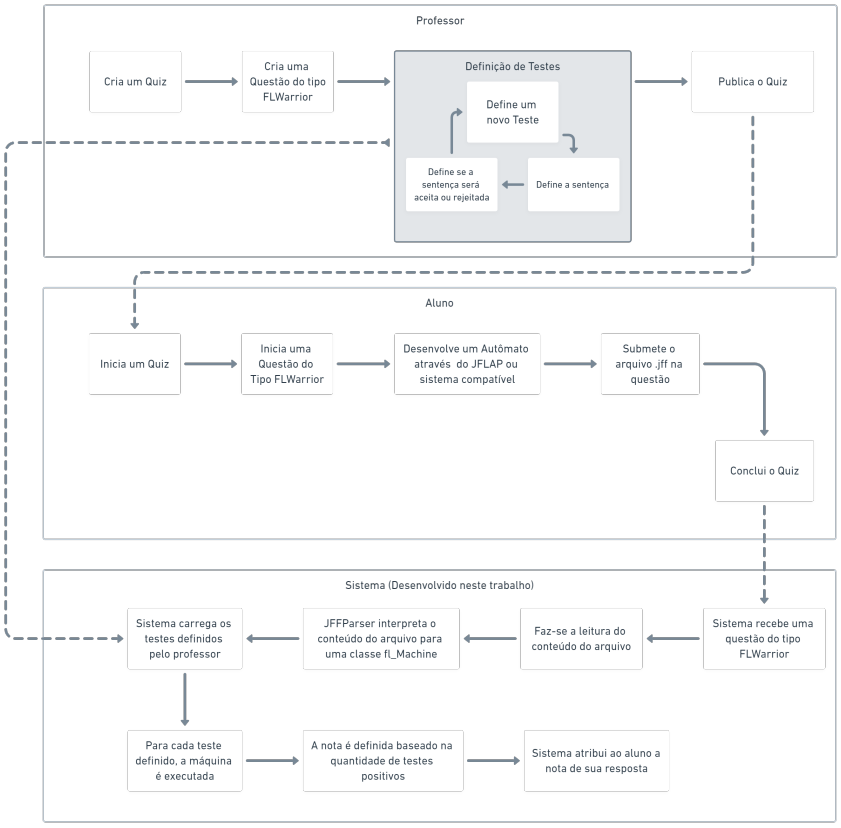


Figura 2. Visão de alto nível do processo de avaliação automático

4.2. Visão do Professor

Inicialmente, para configurar um formulário de questão disponível para utilização pelo professor, é necessário alterar o arquivo responsável por construir a interface vista pelo docente ao criar uma nova questão. No caso deste trabalho, estendeu-se o formulário padrão a fim de adicionar a seção de *Machine Tests* (Testes de Máquina), como visto na Figura 3.

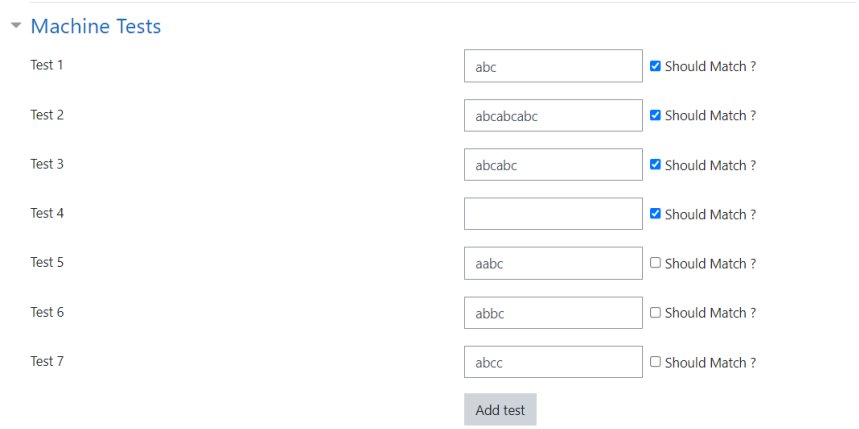


Figura 3. Seção de definição de testes de máquina (no caso, uma máquina *M*, onde $L(M) = (abc)^n, n \geq 0$), presente ao criar ou editar uma questão

Como complemento são definidos os métodos de deleção da questão, permitindo a exclusão da questão, seus testes e arquivos associados, no caso de submissão de arquivos. Também é implementado o método de recuperação destes parâmetros, onde utilizando a interface de acesso ao banco de dados, é possível recuperar os testes utilizando o identificador da questão.

4.3. Visão do Aluno

Para a construção do formulário que o aluno responde, é preciso definir um arquivo cuja função é de apresentar tanto o texto da questão quanto os campos de entrada da resposta. Como o foco deste trabalho é a avaliação automática, utilizou-se como entrada o envio de um arquivo no formato JFLAP 4. Esse formato é simples e pode ser convertido a partir de outros formatos de descrição de autômatos. Para aceitar um arquivo, usou-se a classe *form_filemanager*, presente no Moodle.

Como foi utilizado o sistema de arquivos, o Moodle automaticamente se responsabiliza por armazenar e carregar os arquivos enviados pelo aluno. Contudo, é necessário implementar uma série de funções para gerenciar o comportamento da questão. A primeira dela é verificar se a questão foi alterada e precisa ser reescrita no banco. Para isso, verifica-se o conteúdo dos arquivos enviados onde, sendo diferentes, será chamada a rotina de armazenamento do arquivo. Isso economiza recursos e permite que o servidor tenha mais vazão de comandos.

Como resultado, tem-se na Figura 4 a interface gráfica da questão para o estudante.

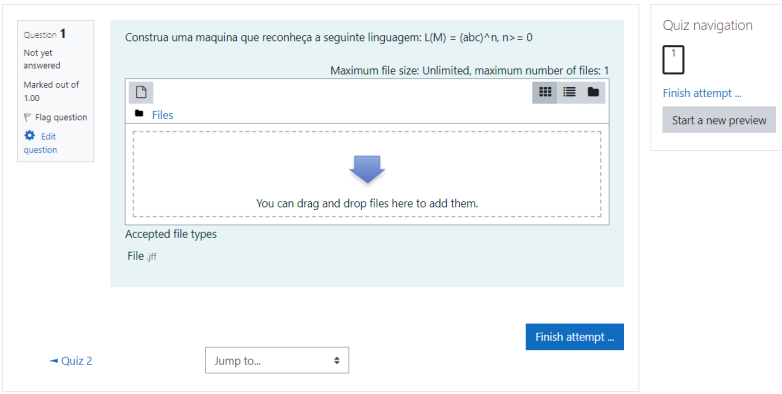


Figura 4. Formulário de resposta visualizado pelo aluno

4.4. Avaliação Automática

Após a submissão da resposta, o motor interno do Moodle executa um método onde é recebido como parâmetro um objeto contendo os campos enviados no formulário do aluno. Com ele, conseguimos obter o conteúdo do arquivo em formato de texto. A partir destes textos, é possível utilizar um analisador para interpretar o conteúdo e instanciar classes que representam uma máquina. Neste caso foi desenvolvida a classe *JFFParser*, cujo objetivo é fazer a análise de arquivos de tipo JFLAP 4 e retornar uma abstração de máquina, sendo do tipo *fl_machine*.

Caso haja sucesso na interpretação da máquina, executam-se os testes um a um. A visão em alto nível deste algoritmo é vista na Figura 5. Para a execução da máquina, utilizamos um algoritmo de busca em largura na árvore de computação. Partimos de uma lista de ”*threads*”de execução (uma estrutura contendo o estado da máquina, entrada restante, e fita ou pilha caso necessário), contendo inicialmente o estado inicial e a sentença testada. Então é verificada a condição de aceitação. Caso não esteja em condição de aceite a máquina tentará mapear a lista de ”*threads*”, aplicando as possíveis transições, num limite máximo de iterações para evitar que o sistema entre em *loop* ou tome muitos recursos do servidor.

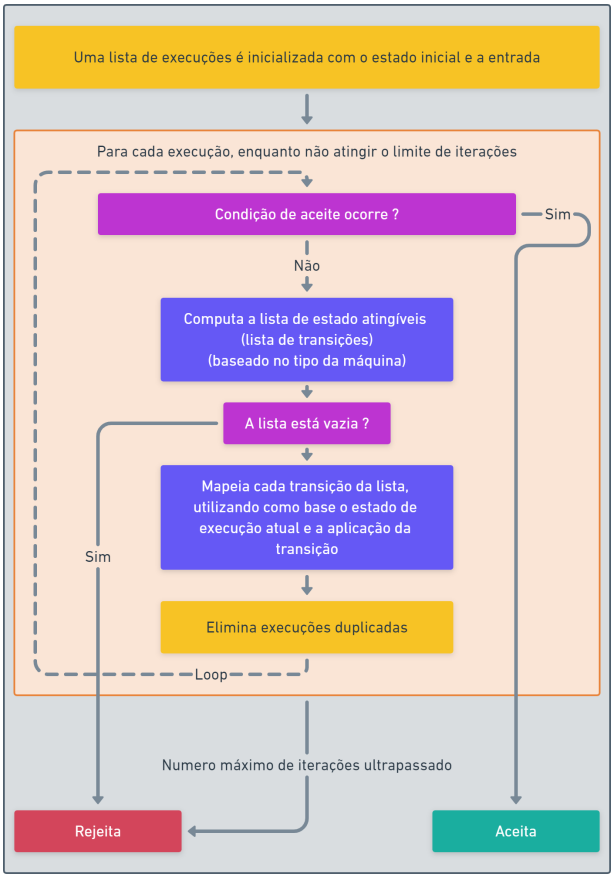


Figura 5. Visão geral do algoritmo de execução de autômatos

Contendo o número de testes concluídos com sucesso, pode-se atribuir uma nota final à resposta. A nota é então visualizada pelo aluno na tela de visualização do *quiz*, como visto na Figura 5.

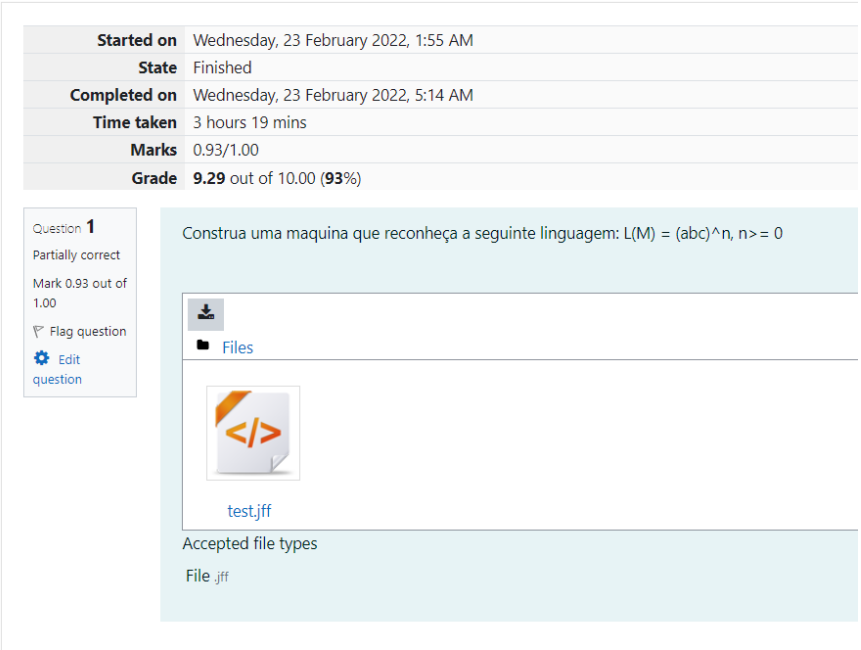


Figura 6. Visualização do aluno da nota atribuída à questão pela avaliação automática, onde obteve-se uma nota parcial

Esta nota fica então disponível ao aluno para futura visualizações, podendo ele efetuar o *download* do arquivo para análise, isto é, conseguir reutilizar no JFLAP o arquivo enviado no *quiz*.

5. Instalação do *Plugin*

1. Download do *Plugin*
   (a) Para efetuar o download do plugin, proceda ao repositório do projeto em `https://github.com/EnzoAlbornoz/qtype_flwarrior`
   (b) Selecione a opção "*Code*"e em seguida "*Download Zip*"
2. Instalação via Gerenciador de *Plugins*
   (a) Para efetuar a instalação deste plugin o administrador do sistema deve acessar seção de "Administração do Site"[1]
   (b) Após isso, selecione a aba "*Plugins*"
   (c) Avance para "*Install plugins*"(ou sua versão localizada, em caso da instância estiver sob alguma outra linguagem). Uma nova tela para a submissão de arquivos aparecerá.
   (d) Com isso basta anexar o arquivo, que fora feito o download previamente, na seção "*Install plugin from ZIP file*"

[1] A versão do Moodle necessária para o prosseguimento desta operação é a 3.11.

(e) Após isso basta acionar o botão "*Install plugin from the Zip file*". Caso ocorra algum erro, verifique as permissões do usuário. Em caso de sucesso, um pequeno relatório de alterações aparecerá.

6. Guia de uso pelo Professor

1. O professor deverá criar uma nova atividade do tipo *Quiz*[2]
2. Preencha normalmente a estrutura do *quiz*
3. Ao criar uma nova questão, defina o tipo de questão como sendo do tipo *FLWarrior*
4. No segmento de *Machine Tests*, defina os casos de teste para a linguagem requerida na questão, definindo a sentença a ser verificada e se ela deve ser aceita ou não pela máquina
5. Após isso, basta publicar o *quiz* como qualquer outro dentro do Moodle.

7. Guia de uso pelo Aluno

1. Como aluno, acesse o *quiz* o qual deseja responder
2. Vá até a questão correspondente do tipo *FLWarrior*
3. Agora o aluno deve submeter o arquivo de máquina[3]. Clique em anexar e selecione o arquivo ou arraste-o para dentro da caixa indicada. O arquivo deverá aparecer onde estava a caixa.
4. Basta agora continuar o *quiz*, respondendo outras questões e ao final, submeta todas as respostas
5. Uma tela contendo a nota e a revisão, se configurado pelo professor no *quiz*, deverá aparecer.

8. Considerações Finais

Neste trabalho foi desenvolvido um maior entendimento sobre o processo de criação e execução de *plugins* na plataforma Moodle. Embora outros *plugins* existam e haja uma vasta lista de documentos descrevendo as classes do AVA, não existem guia explicativos que descrevam o processo completo e o fluxo de integração dessas partes. Portanto, ao definir os métodos e classes necessárias para questões de terceiros, atingiu-se o objetivo de compreender a programação do Moodle e a extensão de seu comportamento. Houve sucesso no objetivo de construção de um tipo de questão que automaticamente avalia autômatos. Em relação ao guias, uma vez mapeado todo o fluxo de comportamento da questão, cumpriram-se os objetivos de construção de manuais explicando tanto a instalação no Moodle, quanto o seu uso no dia a dia por professores e alunos. Houve sucesso também no objetivo de publicação deste *plugin* em um repositório aberto[4], ficando disponível para que a comunidade o utilize e quiçá implementem novas funcionalidades.

[2]O professor deverá ter permissão de edição do curso.

[3]O aluno pode decidir construir o arquivo manualmente ou utilizar alguns construtor compatível com o formato JFLAP 4

[4]Disponível em: https://github.com/EnzoAlbornoz/qtype_flwarrior

9. Trabalhos Futuros

Embora cumpridos os objetivos, um teste com professores e alunos não pode ser feito, pois, agravados pela alteração do calendário acadêmico na pandemia de 2021, ocorreram vários choques de horários, dificultando uma implementação. Portanto, em um passo seguinte, seria útil a realização de um teste público, coletando entrevistas dos alunos e professores, para uma melhoria na experiência de usuário.

Outro aspecto também relacionado a experiência de usuário seria a adição de suporte a outros formatos de arquivos que descrevem máquinas, como o presente no FLWarrior. Vinculado a isto, pode-se colocar como visível os testes que deram sucesso ou falha na resposta do aluno (com visibilidade configurada pelo professor). Também seria útil a inclusão de um sistema gerador de sentenças para o professor utilizar.

Outra estratégia para a melhoria da usabilidade seria o desenvolvimento de uma interface gráfica acoplada à questão, gerando um arquivo compatível que possa ser submetido.

Além disso, o algoritmo de execução de máquinas pode ser bastante otimizado se feito uma revisão de complexidade do algoritmo ou até mesmo pela implementação do algoritmo em *WebAssembly* (padrão binário executável em navegadores de internet).

Referências

Albornoz, E. C. and Bianco, A. P. (2021). Enzoalbornoz/flwarrior: Tool for creating and editing formal languages. `https://github.com/EnzoAlbornoz/flwarrior`. [online; acesso em 06-Dezembro-2021].

da Silva, R. S. (2016). *Moodle 3 para gestores, autores e tutores*. Novatec, 1st edition.

Lobb, Richard e Hunt, T. (2021). Trampgeek/moodle-qtype_coderunner: Coderunner: A moodle quiz question type that runs student-submitted program code in a sandbox to check if it satisfies a given set of tests. `https://github.com/trampgeek/moodle-qtype_coderunner`. [online; acesso em 06-Dezembro-2021].

Moodle (2021). Sobre o moodle. `https://docs.moodle.org/all/pt_br/Sobre_o_Moodle`. [online; acesso em 21-Outubro-2021].

Rodger, S. H. (2005). Jflap. `https://www.jflap.org/`. [online; acesso em 06-Dezembro-2021].

APÊNDICE B – ARQUIVOS CONTENDO OS CÓDIGOS FONTE

B.1 ARQUIVOS DE CONFIGURAÇÃO

B.1.1 Declaração dos Contêineres - docker-compose.yaml

```yaml
version: "3.7"

services:
    # Configure Moodle MariaDB Database
    moodledb:
        # Use MariaDB
        image: bitnami/mariadb:10.5
        container_name: moodledb
        # Restart Only on Fail
        restart: on-failure
        # Configure Environment
        environment:
            MARIADB_ROOT_PASSWORD: root
            MARIADB_USER: admin
            MARIADB_PASSWORD: admin
            MARIADB_DATABASE: moodle
        # Configure Persistance
        volumes:
            - "moodle-db:/bitnami/mariadb"
        networks:
            - "moodle-net"
        ports:
            - "3306:3306"
    # Configure Moodle
    moodle:
        depends_on: ["moodledb"]
        # Use Bitnami Moodle
        image: bitnami/moodle:3.11.4
        container_name: moodle
        # Restart Only on Fail
        restart: on-failure
        # Configure Environment
        environment:
            MOODLE_USERNAME: admin
            MOODLE_PASSWORD: supersecure
            MOODLE_EMAIL: enzo.c.albornoz@grad.ufsc.br
            MOODLE_DATABASE_HOST: moodledb
```

```
38              MOODLE_DATABASE_PORT_NUMBER: 3306
39              MOODLE_DATABASE_USER: root
40              MOODLE_DATABASE_PASSWORD: root
41              MOODLE_DATABASE_NAME: moodle
42          # Configure Mount Points
43          volumes:
44              # Persis Moodle Data
45              - "moodle-data:/bitnami"
46              # Mount All Files on this Folder - Comment this line when moodle DB is
                ↪  empty
47              - ".:/bitnami/moodle/question/type/flwarrior"
48          networks:
49              - "moodle-net"
50          ports:
51              - "8080:8080"
52              - "8443:8443"
53
54      # Compile JS Files
55      grunt:
56          depends_on: ["moodle"]
57          # Use Grunt Docker
58          image: node:14-bullseye
59          container_name: moodle_grunt
60          # Define Initial Task
61          entrypoint: ["/bin/bash", "-c", "apt update && apt install watchman && npm
            ↪  install && npx grunt watch"]
62          # Restart Only on Fail
63          restart: on-failure
64          # Configure Volumes
65          volumes:
66              # Load Moodle Data
67              - "moodle-data:/bitnami"
68              # Mount All Files on this Folder
69              - ".:/bitnami/moodle/question/type/flwarrior"
70          # Define Workspace Folder
71          working_dir: "/bitnami/moodle"
72          networks:
73              - "moodle-net"
74  networks:
75      moodle-net:
76          driver: "bridge"
77  volumes:
```

```
78      moodle-db:
79      moodle-data:
```

B.1.2 Declaração do Banco de Dados XMLDB - *install.xml*

```
1  <XMLDB PATH="question/type/flwarrior/db" VERSION="20210628" COMMENT="XMLDB file for
   ↪  Moodle question/type/flwarrior">
2    <TABLES>
3      <TABLE NAME="qtype_flwarrior_tests" COMMENT="Stores question tests">
4        <FIELDS>
5          <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" SEQUENCE="true"/>
6          <FIELD NAME="question_id" TYPE="int" LENGTH="10" NOTNULL="true"
           ↪  SEQUENCE="false"/>
7          <FIELD NAME="word" TYPE="text" NOTNULL="true" SEQUENCE="false"/>
8          <FIELD NAME="should_match" TYPE="int" LENGTH="1" NOTNULL="true" DEFAULT="1"
           ↪  SEQUENCE="false"/>
9          <FIELD NAME="max_iterations" TYPE="int" LENGTH="15" NOTNULL="true"
           ↪  DEFAULT="1000" SEQUENCE="false"/>
10       </FIELDS>
11       <KEYS>
12         <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
13         <KEY NAME="question_id" TYPE="foreign" FIELDS="question_id"
           ↪  REFTABLE="question" REFFIELDS="id"/>
14       </KEYS>
15     </TABLE>
16   </TABLES>
17 </XMLDB>
```

B.2 ARQUIVOS FONTE PARA O MOODLE

B.2.1 Renderização da questão para o aluno - *renderer.php*

```
1  <?php
2  // This file is part of Moodle - http://moodle.org/
3  //
4  // Moodle is free software: you can redistribute it and/or modify
5  // it under the terms of the GNU General Public License as published by
6  // the Free Software Foundation, either version 3 of the License, or
7  // (at your option) any later version.
8  //
9  // Moodle is distributed in the hope that it will be useful,
10 // but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
11 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12 // GNU General Public License for more details.
13 //
14 // You should have received a copy of the GNU General Public License
15 // along with Moodle.  If not, see <http://www.gnu.org/licenses/>.
16
17 /**
18  * flwarrior question renderer class.
19  *
20  * @package   qtype_flwarrior
21  * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
22  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
23  */
24
25 defined('MOODLE_INTERNAL') || die();
26
27 /**
28  * Generates the output for flwarrior questions.
29  *
30  * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
31  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
32  */
33 class qtype_flwarrior_renderer extends qtype_renderer {
34
35     public function formulation_and_controls(
36             question_attempt $qa,
37             question_display_options $options
38     ) {
39         global $PAGE;
40
41         /* @var qtype_flwarrior_question Fetch Question Data */
42         $question = $qa->get_question();
43
44         /* Fetch Question Text */
45         $question_text = $question->format_questiontext($qa);
46         /* Create Elements */
47         $file_input = $this->files_input($qa, 1, $options);
48         $result = <<<HTML
49             <div id="question-header" class="qtext">
50                 <span>{$question_text}</span>
51             </div>
52             <div id="question-form" class="ablock">
```

```
53                 <div class="attachments">
54                     {$file_input}
55                 </div>
56             </div>
57         HTML;
58         // Add JS
59         return $result;
60     }
61 
62     public function specific_feedback(question_attempt $qa) {
63         // TODO.
64         return '';
65     }
66 
67     public function correct_response(question_attempt $qa) {
68         // TODO.
69         return '';
70     }
71 
72     // From qtype_essay
73     public function files_input(
74         question_attempt $qa,
75         int $num_allowed,
76         question_display_options $options
77     ): string {
78         $filetypeslist = ".jff";
79         global $CFG, $COURSE;
80         require_once($CFG->dirroot . '/lib/form/filemanager.php');
81 
82         $pickeroptions = new stdClass();
83         $pickeroptions->mainfile = null;
84         $pickeroptions->maxfiles = $num_allowed;
85         $pickeroptions->itemid = $qa->prepare_response_files_draft_itemid(
86             'machine', $options->context->id);
87         $pickeroptions->context = $options->context;
88         $pickeroptions->return_types = FILE_INTERNAL | FILE_CONTROLLED_LINK;
89 
90         $pickeroptions->itemid = $qa->prepare_response_files_draft_itemid(
91             'machine', $options->context->id);
92         $pickeroptions->accepted_types = $filetypeslist;
93 
94         $fm = new form_filemanager($pickeroptions);
```

```
95          $fm->options->maxbytes = get_user_max_upload_file_size(
96              $this->page->context,
97              $CFG->maxbytes,
98              $COURSE->maxbytes,
99              $qa->get_question()->maxbytes
100         );
101         $filesrenderer = $this->page->get_renderer('core', 'files');
102
103         $text = '';
104         if (!empty($filetypeslist)) {
105             $text = html_writer::tag(
106                 'p',
107                 get_string('accepted_file_types', 'qtype_flwarrior')
108             );
109             $filetypesutil = new \core_form\filetypes_util();
110             $filetypes = $filetypeslist;
111            $filetypedescriptions = $filetypesutil->describe_file_types($filetypes);
112            $text .= $this->render_from_template('core_form/filetypes-descriptions',
                ↪  $filetypedescriptions);
113         }
114
115         $output = html_writer::start_tag('fieldset');
116         $output .= html_writer::tag(
117             'legend',
118             get_string('answer_files', 'qtype_flwarrior'),
119             ['class' => 'sr-only']
120         );
121         $output .= $filesrenderer->render($fm);
122         $output .= html_writer::empty_tag('input', [
123             'type' => 'hidden',
124             'name' => $qa->get_qt_field_name('machine'),
125             'value' => $pickeroptions->itemid,
126         ]);
127         $output .= $text;
128         $output .= html_writer::end_tag('fieldset');
129
130         return $output;
131     }
132 }
```

B.2.2 Definição da questão dentro do Moodle - *question.php*

```php
<?php
// This file is part of Moodle - http://moodle.org/
//
// Moodle is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Moodle is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Moodle.  If not, see <http://www.gnu.org/licenses/>.

/**
 * Fl Warrior question definition class.
 *
 * @package    qtype_flwarrior
 * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
 * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */


defined('MOODLE_INTERNAL') || die();

global $CFG;
require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_machine_test.php');
require_once($CFG->dirroot . '/question/type/flwarrior/lib/interop/JFFParser.php');
require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_machine.php');

/**
 * Represents a FL Warrior question.
 *
 * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
 * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */
class qtype_flwarrior_question extends question_graded_automatically
{

```

```
42      // Teacher Defined Variables ===========================================
43      /* @type fl_machine_test[] Test List */
44      public array $machine_tests;
45
46      public function get_expected_data(): array
47      {
48          return array('machine' => question_attempt::PARAM_FILES);
49      }
50
51      public function apply_attempt_state(question_attempt_step $step)
52      {
53      }
54
55      /**
56       * @throws coding_exception
57       */
58      public function summarise_response(array $response)
59      {
60          if (isset($response['machine']) && $response['machine']) {
61              $machine_files = $response['machine']->get_files();
62              if (array_key_exists(0, $machine_files)) {
63                  $file = $machine_files[0];
64                  return get_string(
65                      'attached_files',
66                      'qtype_flwarrior',
67                      implode(
68                          ', ',
69                          $file->get_filename() . ' (' .
                            ↪  display_size($file->get_filesize()) . ')'
70                      )
71                  );
72              }
73          }
74          return '';
75      }
76
77      public function is_complete_response(array $response): bool
78      {
79          // Check the filetypes.
80          /** @type question_file_saver $machine_field */
81          $machine_field = $response['machine'];
82          $filetypes_util = new \core_form\filetypes_util();
```

```
83          $allow_list = $filetypes_util->normalize_file_types(".jff");
84          // Check Machine is set
85          if (!(isset($machine_field) && $machine_field)) {
86              return false;
87          }
88          // Check Machine has at least 1 machine
89          $machine_files = array_values($machine_field->get_files());
90
91          if (!array_key_exists(0, $machine_files)) {
92              return false;
93          }
94          // Check Machine file type
95          $file = $machine_files[0];
96
97          return $filetypes_util->is_allowed_file_type($file->get_filename(),
            ↪  $allow_list);
98      }
99
100     public function validate_response(array $response)
101     {
102
103     }
104
105     public function get_validation_error(array $response)
106     {
107         return '';
108     }
109
110     public function is_same_response(array $prevresponse, array $newresponse)
111     {
112         /** @var question_response_files $prev_response_machine */
113         $prev_response_machine = $prevresponse['machine'];
114         /** @var question_response_files $new_response_machine */
115         $new_response_machine = $newresponse['machine'];
116
117         if ($prev_response_machine == null && $new_response_machine == null) {
118             return true;
119         }
120
121         if (
122             ($prev_response_machine == null && $new_response_machine != null) ||
123             ($prev_response_machine != null && $new_response_machine == null)
```

```
124         ) {
125             return false;
126         }
127 
128         /** @var stored_file[] $prev_files */
129         $prev_files = array_values($prev_response_machine->get_files());
130         /** @var stored_file[] $new_files */
131         $new_files = array_values($new_response_machine->get_files());
132 
133         if (count($prev_files) != count($new_files)) {
134             return false;
135         }
136 
137         // Check Content Hash
138         for ($idx = 0; $idx < count($prev_files); $idx++) {
139             if ($prev_files[$idx]->get_contenthash() !=
                ↪  $new_files[$idx]->get_contenthash()) {
140                 return false;
141             }
142         }
143 
144         return true;
145     }
146 
147     public function get_correct_response()
148     {
149         return null;
150     }
151 
152     public function check_file_access(
153         $qa,
154         $options,
155         $component,
156         $filearea,
157         $args,
158         $forcedownload
159     )
160     {
161         if ($component == 'question' && $filearea == 'hint') {
162             return $this->check_hint_file_access($qa, $options, $args);
163 
164         } else {
```

```
165             return parent::check_file_access($qa, $options, $component, $filearea,
166                 $args, $forcedownload);
167         }
168     }
169
170     public function is_gradable_response(array $response): bool
171     {
172         return (
173             array_key_exists('machine', $response) &&
174             $response['machine'] instanceof question_response_files
175         );
176     }
177
178     public function grade_response(array $response): array
179     {
180         // Get Machine file
181         /** @var stored_file[] $machine_files */
182         $machine_files = array_values($response['machine']->get_files());
183         // Get Machine Content
184         $machine_file_content = $machine_files[0]->get_content();
185         // Parse Machine
186         $machine = JFFParser::parse_string($machine_file_content);
187         // If there is any error, just grade as 0
188         if ($machine == null) {
189             return array(0, question_state::graded_state_for_fraction(0));
190         }
191         // Compute fraction for each test
192         $test_fraction_bonus = 1 / count($this->machine_tests);
193         // Define initial fraction
194         $fraction = 0;
195         // Execute tests
196         foreach ($this->machine_tests as $raw_test) {
197             //Parse Test
198             $test = fl_machine_test::from_array($raw_test);
199             $matches = $machine->matches($test);
200             if ($matches) {
201                 $fraction += $test_fraction_bonus;
202             }
203         }
204         // Return Computed Fraction
205         return array($fraction,
        ↪  question_state::graded_state_for_fraction($fraction));
```

```
206     }
207
208     // Return a map from filename to file contents for all the attached files
209     // in the given response.
210     private function get_attached_files($response): array
211     {
212         $attachments = array();
213         if (array_key_exists('machine', $response) && $response['machine']) {
214             $files = $response['machine']->get_files();
215             foreach ($files as $file) {
216                 $attachments[$file->get_filename()] = $file->get_content();
217             }
218         }
219         return $attachments;
220     }
221 }
```

B.2.3 Gestão da questão dentro do Moodle - *questiontype.php*

```
1  <?php
2  // This file is part of Moodle - http://moodle.org/
3  //
4  // Moodle is free software: you can redistribute it and/or modify
5  // it under the terms of the GNU General Public License as published by
6  // the Free Software Foundation, either version 3 of the License, or
7  // (at your option) any later version.
8  //
9  // Moodle is distributed in the hope that it will be useful,
10 // but WITHOUT ANY WARRANTY; without even the implied warranty of
11 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12 // GNU General Public License for more details.
13 //
14 // You should have received a copy of the GNU General Public License
15 // along with Moodle.  If not, see <http://www.gnu.org/licenses/>.
16
17 /**
18  * Question type class for the flwarrior question type.
19  *
20  * @package   qtype_flwarrior
21  * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
22  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
23  */
```

```php

defined('MOODLE_INTERNAL') || die();

global $CFG;

require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_machine_test.php');
require_once($CFG->dirroot . '/question/type/flwarrior/lib/interop/JFFParser.php');
require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_machine.php');
require_once($CFG->dirroot . '/question/type/flwarrior/lib/utils.php');
require_once($CFG->libdir . '/questionlib.php');
require_once($CFG->dirroot . '/question/engine/lib.php');
require_once($CFG->dirroot . '/question/type/flwarrior/question.php');

/**
 * The flwarrior question type.
 *
 * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
 * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
 */
class qtype_flwarrior extends question_type
{

    public function response_file_areas()
    {
        return array('machine');
    }

    /**
     * Upsert a list of machine tests for a given question
     * @param qtype_flwarrior_question $question
     * @return void
     * @throws dml_exception
     */
    public function save_question_options($question)
    {
        global $DB;

        $this->save_hints($question);
        // Save Tests in Database
        foreach ($question->{'machine-test-{no}'} as $test_form) {
            $id = array_key_exists('machine-test-id-{no}', $test_form)
```

```
66                  ? $test_form['machine-test-id-{no}']
67                  : null;
68              $word = array_key_exists('machine-test-word-{no}', $test_form)
69                  ? $test_form['machine-test-word-{no}']
70                  : '';
71              $should_match = boolval(
72                  array_key_exists('machine-test-should-match-{no}', $test_form)
73                      ? $test_form['machine-test-should-match-{no}']
74                      : 0
75              );
76
77              $test = new fl_machine_test($id, $word, $should_match, 1000,
                ↪  $question->id);
78
79              if ($test->id != null && $DB->record_exists('qtype_flwarrior_tests',
                ↪  array('id' => $test->id))) {
80                  // Update
81                  $DB->update_record('qtype_flwarrior_tests', $test);
82              } else {
83                  /** @var int $inserted_id Create new entry */
84                  $inserted_id = $DB->insert_record('qtype_flwarrior_tests', $test);
85                  $test->id = $inserted_id;
86              }
87          }
88
89      }
90
91      /**
92       * Retrieves a list of machine tests that around bounded with the given question
    ↪  id
93       * @param qtype_flwarrior_question $question
94       * @return bool
95       */
96      public function get_question_options($question)
97      {
98          global $CFG, $DB, $OUTPUT;
99          if (parent::get_question_options($question)) {
100
101             // Fetch and Parse Tests from DB (indexed by id)
102             $tests = array_map(
103                 function ($db_test) {
104                     return fl_machine_test::from_db_entry_to_array($db_test);
```

```
105                 },
106                 $DB->get_records('qtype_flwarrior_tests', array('question_id' =>
                    ↪  $question->id))
107             );
108
109             // Insert tests into question object
110             $question->machine_tests = $tests ? [...$tests] : array();
111
112             return true;
113         }
114         return false;
115     }
116
117     /**
118      * @throws dml_exception
119      */
120     public function delete_question($questionid, $contextid)
121     {
122         global $DB;
123         $success = $DB->delete_records('qtype_flwarrior_tests', array('question_id'
                ↪  => $questionid));
124         if ($success) {
125             parent::delete_question();
126         }
127     }
128
129     public function get_random_guess_score($questiondata)
130     {
131         return 0;
132     }
133
134     public function get_possible_responses($questiondata)
135     {
136         return array();
137     }
138
139     protected function initialise_question_instance(question_definition $question,
        ↪  $questiondata) {
140         // Load parent data
141         parent::initialise_question_instance($question, $questiondata);
142         // Load tests
```

```
143         error_log("[initialise_question_instance]\n".print_r($questiondata->machin
            ↪  e_tests,
            ↪  true));
144         $question->machine_tests = $questiondata->machine_tests;
145     }
146
147     public function move_files($questionid, $oldcontextid, $newcontextid)
148     {
149         parent::move_files($questionid, $oldcontextid, $newcontextid);
150         $fs = get_file_storage();
151         $fs->move_area_files_to_new_context($oldcontextid,
152             $newcontextid, 'qtype_essay', 'graderinfo', $questionid);
153     }
154
155     protected function delete_files($questionid, $contextid)
156     {
157         parent::delete_files($questionid, $contextid);
158         $fs = get_file_storage();
159         $fs->delete_area_files($contextid, 'qtype_essay', 'graderinfo',
            ↪  $questionid);
160     }
161 }
```

B.2.4 Formulário de edição da questão - *edit_flwarrior_form.php*

```
1  <?php
2  // This file is part of Moodle - http://moodle.org/
3  //
4  // Moodle is free software: you can redistribute it and/or modify
5  // it under the terms of the GNU General Public License as published by
6  // the Free Software Foundation, either version 3 of the License, or
7  // (at your option) any later version.
8  //
9  // Moodle is distributed in the hope that it will be useful,
10 // but WITHOUT ANY WARRANTY; without even the implied warranty of
11 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12 // GNU General Public License for more details.
13 //
14 // You should have received a copy of the GNU General Public License
15 // along with Moodle.  If not, see <http://www.gnu.org/licenses/>.
16
17 /**
```

```
18  * Defines the editing form for the flwarrior question type.
19  *
20  * @package   qtype_flwarrior
21  * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
22  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
23  */
24
25 defined('MOODLE_INTERNAL') || die();
26
27 /**
28  * flwarrior question editing form definition.
29  *
30  * ?Note: This file will define the form when the teacher is creating a question
31  *
32  * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
33  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
34  */
35 class qtype_flwarrior_edit_form extends question_edit_form
36 {
37
38     const NUM_TESTCASES_START = 5;  // Num empty test cases with new questions.
39     const NUM_TESTCASES_ADD = 1;    // Extra empty test cases to add.
40
41     public function qtype(): string
42     {
43         return 'flwarrior';
44     }
45
46     /**
47      * @param MoodleQuickForm $mform
48      * @throws coding_exception
49      */
50     protected function definition_inner($mform)
51     {
52         global $COURSE, $CFG, $DB, $PAGE;
53         /* Add Machine Tests Section */
54         $mform->addElement('header', 'machine-tests', get_string(
55             'question_heading_machine_tests', 'qtype_flwarrior'
56         ));
57
58         /* @type qtype_flwarrior_question $question Repeat for each test */
59         $question = $this->question;
```

```
60          /* Define Repeating Groups*/
61          $tests_in_question = count($question->machine_tests);
62          $repeating_num = max($tests_in_question, 1);
63          $repeating_groups = array();
64          $repeated_options = array();
65
66          /* Define what going to be grouped in repeated elements */
67          $repeat_array = array();
68          /* @type HTML_QuickForm_text $test_word */
69          $test_word = $mform->createElement('text', 'machine-test-word-{no}', '');
70          /* @type HTML_QuickForm_checkbox $test_should_match */
71          $test_should_match = $mform->createElement(
72              'advcheckbox',
73              'machine-test-should-match-{no}',
74              null,
75              'Should Match ?',
76              null,
77              array(0, 1)
78          );
79          $test_should_match->setValue(true);
80          /* @type HTML_QuickForm_hidden $test_id */
81          $test_id = $mform->createElement('hidden', 'machine-test-id-{no}');
82
83          /* Add them to group */
84          $repeat_array[] = $test_word;
85          $repeat_array[] = $test_should_match;
86          $repeat_array[] = $test_id;
87
88          /* Add elements to group*/
89          /* @type HTML_QuickForm_group $repeatGroup */
90          $repeatGroup = $mform->createElement('group', 'machine-test-{no}', 'Test
            ↪  {no}');
91          $repeatGroup->setElements($repeat_array);
92
93          /* Define Group Options*/
94          $repeated_options['machine-test-{no}[machine-test-word-{no}]']['type'] =
            ↪  PARAM_TEXT;
95          $repeated_options['machine-test-{no}[machine-test-id-{no}]']['type'] =
            ↪  PARAM_INT;
96          $repeated_options['machine-test-{no}[machine-test-should-match-{no}]']['ty
            ↪  pe'] =
            ↪  PARAM_BOOL;
```

```
97          /* Add group to form */
98          $repeating_groups[] = $repeatGroup;
99
100         /* Repeat elements */
101         $this->repeat_elements(
102             $repeating_groups,
103             $repeating_num,
104             $repeated_options,
105             'option_repeats',
106             'option_add_fields',
107             1,
108             'Add test',
109             true
110         );
111
112         /* For already existing tests, fill the values */
113         for ($idx = 0; $idx < $tests_in_question; $idx++) {
114             /* Transform array of elements into an object with the elements as
                ↪  values with their name as key */
115             $group_elements = flu_object_from_entries(
116                 array_map(
117                     function ($el) {
118                         return array($el->getName(), $el);
119                     },
120                     $mform->getElement('machine-test-{no}[' . $idx .
                        ↪  ']')->getElements()
121                 )
122             );
123             /* Update values */
124             $group_elements->{'machine-test-word-{no}'}->setValue($question->machi
                ↪  ne_tests[$idx]->word);
125             $group_elements->{'machine-test-should-match-{no}'}->setValue($questio
                ↪  n->machine_tests[$idx]->should_match);
126             $group_elements->{'machine-test-id-{no}'}->setValue($question->machine
                ↪  _tests[$idx]->id);
127         }
128
129     }
130
131     protected function data_preprocessing($question)
132     {
133         global $PAGE;
```

```
134         $question = parent::data_preprocessing($question);
135         $question = $this->data_preprocessing_hints($question);
136 
137         return $question;
138     }
139 }
```

B.3 ARQUIVOS FONTE DA BIBLIOTECA DE AUTÔMATOS

B.3.1 Camada de Interoperabilidade com o JFLAP - *JFFParser.php*

```
1  <?php
2 
3  global $CFG;
4 
5  require_once($CFG->dirroot . '/question/type/flwarrior/lib/utils.php');
6  require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_symbol.php');
7  require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_alphabet.php');
8  require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_state.php');
9  require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_transition.php');
10 require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_machine.php');
11 
12 class JFFParser
13 {
14     public static function parse_string(string $str)
15     {
16         // Instantiate Document
17         $xml = new DOMDocument();
18         // Parse XML
19         $xml->loadXML($str);
20         // Build Objects Based on JFF file format
21         // Read Machine Type
22         $xml_machine_type = $xml->getElementsByTagName("type")->item(0);
23         if ($xml_machine_type == null) {
24             return null;
25         }
26         /** @var string $machine_type */
27         $machine_type = $xml_machine_type->textContent;
28         // Read Machine States
29         $xml_states = $xml->getElementsByTagName("state");
30         $states = array_map(function (DOMElement $el) {
31             $is_init = $el->getElementsByTagName("initial")->length > 0;
```

```
32              $is_exit = $el->getElementsByTagName("final")->length > 0;
33              $id = $el->getAttribute("id");
34              return new fl_State($id, $is_init, $is_exit);
35          }, iterator_to_array($xml_states));
36          $states_obj = array_reduce($states, function ($acc, fl_State $state) {
37              $acc[$state->get_label()] = $state;
38              return $acc;
39          }, array());
40          // Read Symbols
41          $xml_symbols = array_merge(
42              iterator_to_array($xml->getElementsByTagName("read")),
43              iterator_to_array($xml->getElementsByTagName("write")),
44              iterator_to_array($xml->getElementsByTagName("pop")),
45              iterator_to_array($xml->getElementsByTagName("push")),
46          );
47          $symbols = array_map(function (DOMElement $el) {
48              if ($el->textContent) {
49                  return fl_Symbol::symbol_for($el->textContent);
50              }
51              return null;
52          }, $xml_symbols);
53          $symbols = array_unique(
54              array_filter(
55                  array_merge(
56                      array(fl_Symbol::EPSILON()),
57                      $symbols
58                  ),
59                  fn($el) => $el !== null
60              )
61          );
62          $alphabet = new fl_Alphabet($symbols);
63          // Read Machine Transitions
64          $xml_transitions = $xml->getElementsByTagName("transition");
65          $transitions = array_map(function (DOMElement $el) use ($machine_type,
            ↪  $states_obj) {
66              /** @var DOMElement $el_from */
67              $el_from = $el->getElementsByTagName("from")->item(0);
68              /** @var DOMElement $el_to */
69              $el_to = $el->getElementsByTagName("to")->item(0);
70              /** @var DOMElement|null $el_pop */
71              $el_pop = $el->getElementsByTagName("pop")->item(0);
72              /** @var DOMElement|null $el_pop */
```

```
73              $el_push = $el->getElementsByTagName("push")->item(0);
74              /** @var DOMElement $el_pop */
75              $el_read = $el->getElementsByTagName("read")->item(0);
76              /** @var DOMElement|null $el_pop */
77              $el_write = $el->getElementsByTagName("write")->item(0);
78              /** @var DOMElement|null $el_pop */
79              $el_move = $el->getElementsByTagName("move")->item(0);
80
81              $from = $states_obj[$el_from->textContent];
82              $to = $states_obj[$el_to->textContent];
83              $with_head_symbol = $el_read->textContent === ""
84                  ? fl_Symbol::EPSILON()
85                  : fl_Symbol::symbol_for($el_read->textContent);
86              $with_memory_symbol = $el_pop != null
87                  ? $el_pop->textContent === ""
88                      ? fl_Symbol::EPSILON()
89                      : fl_Symbol::symbol_for($el_pop->textContent)
90                  : null;
91              if ($machine_type === "turing") {
92                  $write_symbol = $el_write->textContent !== ""
93                      ? fl_Symbol::symbol_for($el_write->textContent)
94                      : fl_Symbol::EPSILON();
95              } else if ($machine_type === "pda") {
96                  $write_symbol = $el_push->textContent !== ""
97                      ? fl_Symbol::symbol_for($el_push->textContent)
98                      : fl_Symbol::EPSILON();
99              } else {
100                 $write_symbol = null;
101             }
102             $head_direction = $el_move !== null && $el_move->textContent !== ""
103                 ? JFFParser::compute_head_direction($el_move->textContent)
104                 : null;
105
106             return new fl_Transition($from, $to, $with_head_symbol,
                ↪  $with_memory_symbol, $write_symbol, $head_direction);
107         }, iterator_to_array($xml_transitions));
108         // Create Machine
109         $machine_type = JFFParser::compute_machine_type($machine_type);
110         if ($machine_type === null) {
111             return null;
112         }
113         return new fl_Machine($machine_type, $alphabet, $states, $transitions);
```

```
114     }
115
116     private static function compute_head_direction(string $move)
117     {
118         if ($move === "L") {
119             return fl_Transition::H_LEFT;
120         } else if ($move === "R") {
121             return fl_Transition::H_RIGHT;
122         }
123     }
124
125     private static function compute_machine_type(string $type): ?string
126     {
127         if ($type === "turing") {
128             return fl_Machine::TURING_MACHINE;
129         } else if ($type === "pda") {
130             return fl_Machine::PUSH_DOWN_MACHINE;
131         } else if ($type === "fa") {
132             return fl_Machine::FINITE_STATE_MACHINE;
133         }
134         return null;
135     }
136 }
```

B.3.2 Abstração de Símbolo - *fl_symbol.php*

```
1  <?php
2
3  class fl_Symbol
4  {
5      // Define attributes
6      private static ?fl_Symbol $EPSILON_CONST = null;
7      private static ?fl_Symbol $END_OF_STACK_CONST = null;
8      private ?int $id = null;
9      private string $symbol_raw;
10
11     // Define constructor
12     private bool $rep_epsilon = false;
13     private bool $rep_end_of_stack = false;
14
15     // Define Symbol public constructor
16
```

```
17      private function __construct(string $symbol_raw, bool $epsilon = false, bool
        ↪  $end_of_stack = false)
18      {
19          $this->symbol_raw = $symbol_raw;
20
21          if ($epsilon) {
22              $this->rep_epsilon = $epsilon;
23          }
24          if ($end_of_stack) {
25              $this->rep_end_of_stack = $end_of_stack;
26          }
27      }
28
29      // Define methods
30
31      public static function symbol_for($symbol_raw): fl_Symbol
32      {
33          return new fl_Symbol($symbol_raw);
34      }
35
36      public static function EPSILON(): fl_Symbol
37      {
38          if (!fl_Symbol::$EPSILON_CONST) {
39              fl_Symbol::$EPSILON_CONST = new fl_Symbol("", true, false);
40          }
41          return fl_Symbol::$EPSILON_CONST;
42      }
43
44      public static function END_OF_STACK(): fl_Symbol
45      {
46          if (!fl_Symbol::$END_OF_STACK_CONST) {
47              fl_Symbol::$END_OF_STACK_CONST = new fl_Symbol("$", true, false);
48          }
49          return fl_Symbol::$END_OF_STACK_CONST;
50      }
51
52      public function __toString()
53      {
54          return $this->symbol_raw . $this->rep_epsilon . $this->rep_end_of_stack;
55      }
56
57      // Define constants
```

```
58
59      public function get_id(): ?int
60      {
61          return $this->id;
62      }
63
64      public function set_id(int $id)
65      {
66          $this->id = $id;
67      }
68
69      public function is_equal_to(fl_Symbol $other): bool
70      {
71          if ($this->rep_epsilon) {
72              return $this->rep_epsilon == $other->rep_epsilon;
73          }
74          if ($this->rep_end_of_stack) {
75              return $this->rep_end_of_stack == $other->rep_end_of_stack;
76          }
77          return !strcmp($this->symbol_raw, $other->symbol_raw);
78      }
79
80      public function cmp(fl_Symbol $other)
81      {
82          if ($this->rep_epsilon) {
83              return $this->rep_epsilon == $other->rep_epsilon;
84          }
85          if ($this->rep_end_of_stack) {
86              return $this->rep_end_of_stack == $other->rep_end_of_stack;
87          }
88          return strcmp($this->symbol_raw, $other->symbol_raw);
89      }
90  }
```

B.3.3 Abstração de Alfabeto - *fl_alphabet.php*

```
1  <?php
2
3  global $CFG;
4  require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_symbol.php');
5
6  class fl_Alphabet
```

```
7  {
8      // Define Attributes
9      private ?int $id = null;
10     /* @type fl_Symbol[] $symbols */
11     private array $symbols = array();
12     private bool $symbols_sorted = false;
13
14     // Define Constructor
15     public function __construct(array $symbols = array())
16     {
17         $this->symbols = $symbols;
18     }
19
20     public static function new_empty()
21     {
22         return new fl_Alphabet(array());
23     }
24
25     // Define Methods
26
27     public function get_id(): ?int
28     {
29         return $this->id;
30     }
31
32     public function set_id(int $id)
33     {
34         $this->id = $id;
35     }
36
37     public function is_equal_to(fl_Alphabet $other): bool
38     {
39         if (count($other->symbols) != count($this->symbols)) {
40             return false;
41         }
42
43         $this->sort_symbols();
44         $other->sort_symbols();
45
46         for ($idx = 0; $idx < count($this->symbols); $idx++) {
47             if (!($this->symbols[$idx]->is_equal_to($other->symbols[$idx]))) {
48                 return false;
```

```
49              }
50          }
51          return true;
52      }
53
54      private function sort_symbols()
55      {
56          if (!$this->symbols_sorted) {
57              usort($this->symbols, function (fl_Symbol $a, fl_Symbol $b) {
58                  return $a->cmp($b);
59              });
60          }
61      }
62
63      /**
64       * @return fl_Symbol[]
65       */
66      public function get_symbols()
67      {
68          return $this->symbols;
69      }
70
71      public function add_symbol(fl_Symbol $symbol)
72      {
73          foreach ($this->symbols as $self_symbol) {
74              if ($self_symbol->is_equal_to($symbol)) {
75                  return;
76              }
77          }
78
79          $this->symbols[] = $symbol;
80          $this->symbols_sorted = true;
81      }
82
83      public function remove_symbol(fl_Symbol $symbol)
84      {
85          for ($idx = 0; $idx < count($this->symbols); $idx++) {
86              if ($this->symbols[$idx]->is_equal_to($symbol)) {
87                  $this->symbols = array_slice($this->symbols, $idx, 1);
88              }
89          }
90      }
```

```
91
92  }
```

B.3.4 Abstração de Estado - *fl_state.php*

```
1   <?php
2
3   class fl_State
4   {
5       // Define Attributes
6       private ?int $id;
7       private string $label;
8       private bool $is_entry;
9       private bool $is_exit;
10
11      // Define Constructor
12      public function __construct(string $label, bool $is_entry, bool $is_exit)
13      {
14          $this->label = $label;
15          $this->is_entry = $is_entry;
16          $this->is_exit = $is_exit;
17      }
18
19      // Define Functions
20
21      public function get_id()
22      {
23          return $this->id;
24      }
25
26      public function set_id(int $id)
27      {
28          $this->id = $id;
29      }
30
31      public function get_label()
32      {
33          return $this->label;
34      }
35
36      public function set_as_entry(bool $is_entry)
37      {
```

```
38          $this->is_entry = $is_entry;
39      }
40
41      public function set_as_exit(bool $is_exit)
42      {
43          $this->is_exit = $is_exit;
44      }
45
46      public function is_entry()
47      {
48          return $this->is_entry;
49      }
50
51      public function is_exit()
52      {
53          return $this->is_exit;
54      }
55
56  }
```

B.3.5 Abstração de Transição - *fl_transititon.php*

```
1   <?php
2
3   global $CFG;
4   require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_state.php');
5   require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_symbol.php');
6
7   class fl_Transition
8   {
9       // Define Constants
10      const H_LEFT = "L";
11      const H_RIGHT = "R";
12      // Define Attributes
13      private ?int $id;
14      private fl_State $from;
15      private fl_State $to;
16      private fl_Symbol $with_head_symbol;
17      private ?fl_Symbol $with_memory_symbol;
18      private ?fl_Symbol $write_symbol;
19      private ?string $head_direction;
20
```

```
21      // Define Constructor
22      public function __construct(
23          fl_State   $from,
24          fl_State   $to,
25          fl_Symbol  $with_head_symbol,
26          ?fl_Symbol $with_memory_symbol = null,
27          ?fl_Symbol $write_symbol = null,
28          ?string    $head_direction = null
29      )
30      {
31          $this->from = $from;
32          $this->to = $to;
33          $this->with_head_symbol = $with_head_symbol;
34          $this->with_memory_symbol = $with_memory_symbol;
35          $this->write_symbol = $write_symbol;
36          $this->head_direction = $head_direction;
37      }
38
39      public function get_state_from()
40      {
41          return $this->from;
42      }
43
44      public function get_state_to()
45      {
46          return $this->to;
47      }
48
49      public function get_with_head_symbol()
50      {
51          return $this->with_head_symbol;
52      }
53
54      public function get_with_memory_symbol()
55      {
56          return $this->with_memory_symbol;
57      }
58
59      public function get_write_symbol()
60      {
61          return $this->write_symbol;
62      }
```

```
63
64      public function get_head_direction()
65      {
66          return $this->head_direction;
67      }
68  }
```

B.3.6 Abastração de Máquina - *fl_machine.php*

```
1   <?php
2
3   global $CFG;
4   require_once($CFG->dirroot . '/question/type/flwarrior/lib/utils.php');
5   require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_machine_test.php');
6   require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_symbol.php');
7   require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_alphabet.php');
8   require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_state.php');
9   require_once($CFG->dirroot . '/question/type/flwarrior/lib/fl_transition.php');
10
11  class fl_Machine
12  {
13      // Define Constants
14      const FINITE_STATE_MACHINE = "fsm";
15      const PUSH_DOWN_MACHINE = "pdm";
16      const TURING_MACHINE = "tm";
17      // Define Attributes
18      private ?int $id;
19      private string $type;
20      private array $states;
21      private array $transitions;
22      private fl_Alphabet $alphabet;
23
24      // Define Constructor
25      public function __construct(string $type, fl_Alphabet $alphabet, $states =
        ↪  array(), $transitions = array())
26      {
27          $this->type = $type;
28          $this->alphabet = $alphabet;
29          $this->states = $states;
30          $this->transitions = $transitions;
31      }
32      // Define Helpers
```

```
33
34      /** @return fl_State[] */
35      public function get_exit_states(): array
36      {
37          return array_filter($this->states, fn(fl_State $state) =>
            ↪  $state->is_exit());
38      }
39
40      public function matches(fl_machine_test $test): bool
41      {
42          // Prepare Data
43          $max_iterations = $test->max_iterations;
44          $input_list = str_split($test->word);
45          $should_match = $test->should_match;
46
47          switch ($this->type) {
48              case "fsm":
49                  return $this->fsm_matches($input_list, $max_iterations,
                    ↪  $should_match);
50              case "pdm":
51                  return $this->pdm_matches($input_list, $max_iterations,
                    ↪  $should_match);
52              case "tm":
53                  return $this->tm_matches($input_list, $max_iterations,
                    ↪  $should_match);
54              default:
55                  return false;
56          }
57      }
58
59      // Define Execution Functions
60
61      private function fsm_matches(array $input, int $max_iterations, bool
        ↪  $should_match): bool
62      {
63          // Define execution threads
64          $threads = array(
65              array(
66                  "input" => $input,
67                  "state" => $this->get_initial_state()
68              )
69          );
```

```
70
71          // Iterate
72          for ($iter = 0; $iter < $max_iterations; $iter++) {
73              // Check Done
74              $done_thread_idx = flu_array_lambda_find_index($threads, fn($el) =>
                ↪  empty($el['input']) && $el['state']->is_exit());
75              if ($done_thread_idx >= 0) {
76                  return $should_match === true;
77              }
78              // From current execution threads, match what states can be reached
79              $computed_thread_states = array_map(function ($thread_state) {
80                  // Struct Thread State
81                  // \-> input = (string[]) pending input
82                  // \-> state = flState
83                  // Entry -> Thread State
84                  // Output -> Thread State[]
85
86                  // Verify transitions that match
87                  $filtered_transitions = array_filter(
88                      $this->transitions,
89                      function (fl_Transition $transition) use ($thread_state) {
90                          $matches_from = $transition->get_state_from() ===
                            ↪  $thread_state['state'];
91                          $matches_with = array_key_exists(0, $thread_state['input'])
                            ↪  && $transition->get_with_head_symbol()->is_equal_to(fl
                            ↪  _Symbol::symbol_for($thread_state['input'][0]));
92                          $matches_with_lambda = $transition->get_with_head_symbol()-
                            ↪  >is_equal_to(fl_Symbol::EPSILON());
93                          return $matches_from && ($matches_with ||
                            ↪  $matches_with_lambda);
94                      }
95                  );
96                  // Compute Thread state and return it
97                  return array_map(function (fl_Transition $transition) use
                    ↪  ($thread_state) {
98                      return array(
99                          "input" => $transition->get_with_head_symbol()->is_equal_to
                            ↪  (fl_Symbol::EPSILON())
100                             ? $thread_state['input']
101                             : array_slice($thread_state['input'], 1),
102                         "state" => $transition->get_state_to(),
103                     );
```

```
104                 }, $filtered_transitions);
105             }, $threads);
106             // Store flatten iteration threads
107             $threads =
                ↪  array_values(array_unique(array_merge(...$computed_thread_states),
                ↪  SORT_REGULAR));
108         }
109         // Max iterations reached
110         return $should_match === false;
111     }
112
113     /** @return fl_State|null */
114     public function get_initial_state(): ?fl_State
115     {
116         $idx = flu_array_lambda_find_index($this->states, fn(fl_State $state) =>
            ↪  $state->is_entry());
117         if ($idx < 0) {
118             return null;
119         }
120         return $this->states[$idx];
121     }
122
123     private function pdm_matches(array $input, int $max_iterations, bool
        ↪  $should_match): bool
124     {
125         // Define execution threads
126         $threads = array(
127             array(
128                 "input" => $input,
129                 "state" => $this->get_initial_state(),
130                 "stack" => array()
131             )
132         );
133         // Iterate
134         for ($iter = 0; $iter < $max_iterations; $iter++) {
135             // Check Done
136             $done_thread_idx = flu_array_lambda_find_index($threads, fn($el) =>
                ↪  empty($el['input']) && $el['state']->is_exit());
137             if ($done_thread_idx >= 0) {
138                 return $should_match === true;
139             }
140             // From current execution threads, match what states can be reached
```

```
141             $computed_thread_states = array_map(function ($thread_state) {
142                 // Struct Thread State
143                 // \-> input = (string[]) pending input
144                 // \-> state = flState
145                 // \-> stack = flSymbol[]
146                 // Entry -> Thread State
147                 // Output -> Thread State[]
148
149                 // Verify transitions that match
150                 $filtered_transitions = array_filter(
151                     $this->transitions,
152                     function (fl_Transition $transition) use ($thread_state) {
153                         $matches_from = $transition->get_state_from() ===
                            ↪  $thread_state['state'];
154                         $matches_with = array_key_exists(0, $thread_state['input'])
                            ↪  && $transition->get_with_head_symbol()->is_equal_to(fl
                            ↪  _Symbol::symbol_for($thread_state['input'][0]));
155                         $matches_with_lambda = $transition->get_with_head_symbol()-
                            ↪  >is_equal_to(fl_Symbol::EPSILON());
156
157                         $matches_stack = $transition->get_with_memory_symbol()->is_
                            ↪  equal_to(fl_Symbol::EPSILON())
                            ↪  ||
158                            (!empty($thread_state['stack']) && $transition->get_wit
                            ↪  h_memory_symbol()->is_equal_to($thread_state['stac
                            ↪  k'][count($thread_state['stack']) -
                            ↪  1]));
159                         return $matches_from && $matches_stack && ($matches_with ||
                            ↪  $matches_with_lambda);
160                     }
161                 );
162                 // Compute Thread state and return it
163                 return array_map(function (fl_Transition $transition) use
                    ↪  ($thread_state) {
164                     // Compute Stack
165                     /** @var fl_Symbol[] $stack */
166                     $stack = $thread_state['stack'];
167                     if (!$transition->get_with_memory_symbol()->is_equal_to(fl_Symb
                        ↪  ol::EPSILON()))
                        ↪  {
168                         array_pop($stack);
169                     }
```

```
170                 if (!$transition->get_write_symbol()->is_equal_to(fl_Symbol::EP
                    ↪ SILON()))
                    ↪ {
171                     $stack[] = $transition->get_write_symbol();
172                 }
173                 // Return Thread State
174                 return array(
175                     "input" => $transition->get_with_head_symbol()->is_equal_to
                        ↪ (fl_Symbol::EPSILON())
176                         ? $thread_state['input']
177                         : array_slice($thread_state['input'], 1),
178                     "state" => $transition->get_state_to(),
179                     "stack" => $stack,
180                 );
181             }, $filtered_transitions);
182         }, $threads);
183         // Store flatten iteration threads
184         $threads =
            ↪ array_values(array_unique(array_merge(...$computed_thread_states),
            ↪ SORT_REGULAR));
185     }
186     // Max iterations reached
187     return $should_match === false;
188 }
189
190 private function tm_matches(array $input, int $max_iterations, bool
    ↪ $should_match): bool
191 {
192     // Define execution threads
193     $threads = array(
194         array(
195             "memory" => array_map(fn($el) => fl_Symbol::symbol_for($el), $input),
196             "state" => $this->get_initial_state(),
197             "mem_idx" => 0
198         )
199     );
200     // Iterate
201     for ($iter = 0; $iter < $max_iterations; $iter++) {
202         // Check Done
203         $done_thread_idx = flu_array_lambda_find_index($threads, fn($el) =>
            ↪ $el['state']->is_exit());
204         if ($done_thread_idx >= 0) {
```

```
205                 return $should_match === true;
206             }
207             // From current execution threads, match what states can be reached
208             $computed_thread_states = array_map(function ($thread_state) {
209                 // Struct Thread State
210                 // \-> memory = (string[]) pending input
211                 // \-> state = flState
212                 // \-> mem_idx = flSymbol[]
213                 // Entry -> Thread State
214                 // Output -> Thread State[]
215
216                 // Verify transitions that match
217                 $filtered_transitions = array_filter(
218                     $this->transitions,
219                     function (fl_Transition $transition) use ($thread_state) {
220                         $matches_from = $transition->get_state_from() ===
                            ↪  $thread_state['state'];
221                         $matches_with = array_key_exists($thread_state["mem_idx"],
                            ↪  $thread_state['memory']) &&
                            ↪  $transition->get_with_head_symbol()->is_equal_to($thre
                            ↪  ad_state['memory'][$thread_state["mem_idx"]]);
222                        $matches_with_lambda = $transition->get_with_head_symbol()-
                            ↪  >is_equal_to(fl_Symbol::EPSILON());
223
224                         return $matches_from && ($matches_with ||
                            ↪  $matches_with_lambda);
225                     }
226                 );
227                 // Compute Thread state and return it
228                 return array_map(function (fl_Transition $transition) use
                    ↪  ($thread_state) {
229                     // Compute New Memory
230                     $memory = $thread_state['memory'];
231                     $mem_idx = $thread_state['mem_idx'];
232                     // Update Write Symbol
233                     $memory[$mem_idx] = $transition->get_write_symbol();
234                     // Move Head
235                     if ($mem_idx == 0 && $transition->get_head_direction() ==
                        ↪  fl_Transition::H_LEFT) {
236                         array_unshift($memory, fl_Symbol::EPSILON());
```

```
237                     } else if ($mem_idx == (count($thread_state['memory']) - 1) &&
                        ↪  $transition->get_head_direction() ==
                        ↪  fl_Transition::H_RIGHT) {
238                         $memory[] = fl_Symbol::EPSILON();
239                     } else if ($transition->get_head_direction() ==
                        ↪  fl_Transition::H_LEFT) {
240                         $mem_idx--;
241                     } else if ($transition->get_head_direction() ==
                        ↪  fl_Transition::H_RIGHT) {
242                         $mem_idx++;
243                     }
244                     // Return Thread State
245                     return array(
246                        "memory" => $transition->get_with_head_symbol()->is_equal_t
                           ↪  o(fl_Symbol::EPSILON())
247                               ? $thread_state['memory']
248                               : array_slice($thread_state['memory'], 1),
249                           "state" => $transition->get_state_to(),
250                           "mem_idx" => $mem_idx,
251                     );
252                 }, $filtered_transitions);
253             }, $threads);
254             // Store flatten iteration threads
255             $threads =
                ↪  array_values(array_unique(array_merge(...$computed_thread_states),
                ↪  SORT_REGULAR));
256         }
257         // Max iterations reached
258         return $should_match === false;
259     }
260 }
```

B.3.7 Abastração de Teste de Máquina - *fl_machine_test.php*

```
1  <?php
2
3  // This file is part of Moodle - http://moodle.org/
4  //
5  // Moodle is free software: you can redistribute it and/or modify
6  // it under the terms of the GNU General Public License as published by
7  // the Free Software Foundation, either version 3 of the License, or
8  // (at your option) any later version.
```

```
9  //
10 // Moodle is distributed in the hope that it will be useful,
11 // but WITHOUT ANY WARRANTY; without even the implied warranty of
12 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13 // GNU General Public License for more details.
14 //
15 // You should have received a copy of the GNU General Public License
16 // along with Moodle.  If not, see <http://www.gnu.org/licenses/>.
17
18 /**
19  * Question type class for the flwarrior question type.
20  *
21  * @package   qtype_flwarrior
22  * @copyright 2021 Enzo Coelho Albornoz <enzocoelhoalbornoz@gmail.com>
23  * @license   http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
24  */
25
26
27 defined('MOODLE_INTERNAL') || die();
28
29 global $CFG;
30
31 class fl_machine_test
32 {
33     public ?int $id;
34     public string $word;
35     public bool $should_match;
36     public int $max_iterations;
37     public ?int $question_id;
38
39     public function __construct(
40         ?int   $id,
41         string $word,
42         bool   $should_match = true,
43         int    $max_iterations = 1000,
44         ?int   $question_id = null
45     )
46     {
47         $this->id = $id;
48         $this->word = $word;
49         $this->should_match = $should_match;
50         $this->max_iterations = $max_iterations;
```

```
51          $this->question_id = $question_id;
52      }
53
54      public function to_array(): array
55      {
56          return array(
57              "id" => $this->id,
58              "word" => $this->word,
59              "should_match" => $this->should_match,
60              "max_iterations" => $this->max_iterations,
61              "question_id" => $this->question_id
62          );
63      }
64
65      static function from_db_entry($db_entry): fl_machine_test
66      {
67          return new fl_machine_test(
68              $db_entry->id,
69              $db_entry->word,
70              boolval($db_entry->should_match),
71              $db_entry->max_iterations,
72              $db_entry->question_id
73          );
74      }
75
76      static function from_array($entry): fl_machine_test
77      {
78          return new fl_machine_test(
79              $entry['id'],
80              $entry['word'],
81              boolval($entry['should_match']),
82              $entry['max_iterations'],
83              $entry['question_id']
84          );
85      }
86
87      static function from_db_entry_to_array($db_entry): array
88      {
89          return array(
90              "id" => $db_entry->id,
91              "word" => $db_entry->word,
92              "should_match" => boolval($db_entry->should_match),
```

```
93              "max_iterations" => $db_entry->max_iterations,
94              "question_id" => $db_entry->question_id
95          );
96      }
97  }
```

APÊNDICE C – MINI-GUIAS

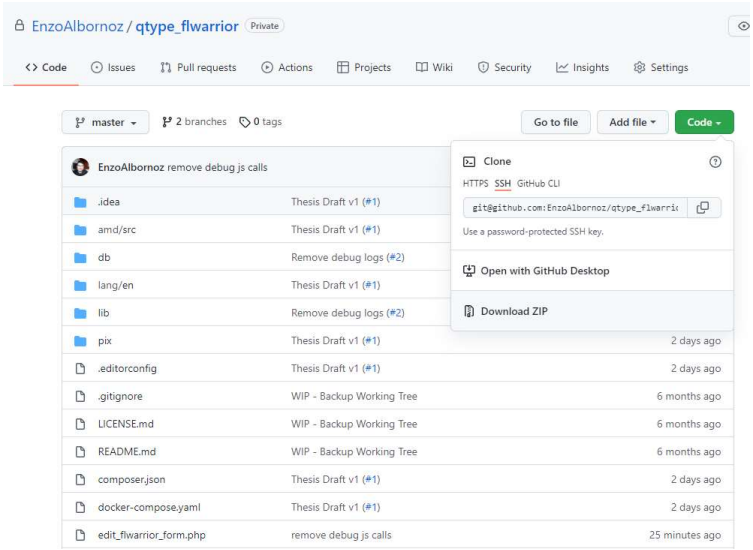
C.1 GUIA DE INSTALAÇÃO

Instalação do Plugin FLWarrior

1 - Download do Plugin

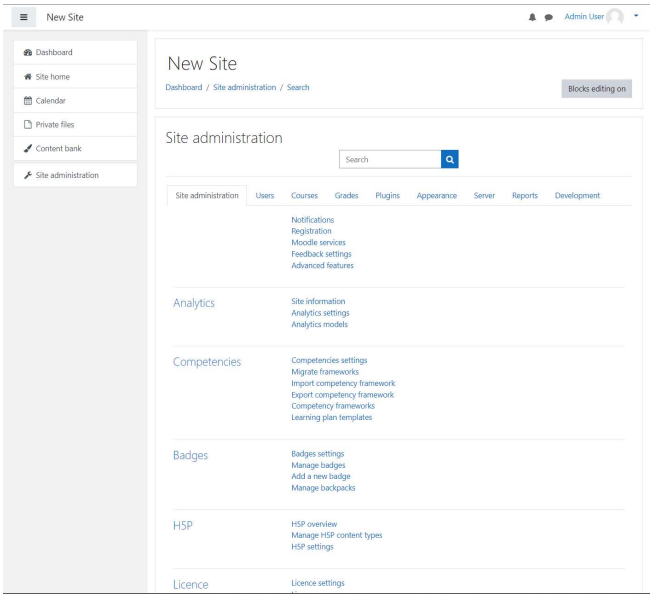
Para efetuar o download do plugin, proceda ao repositório do projeto em https://github.com/EnzoAlbornoz/qtype_flwarrior

Após isto, selecione a opção "Code" e em seguida "Download Zip", como visto na figura abaixo.

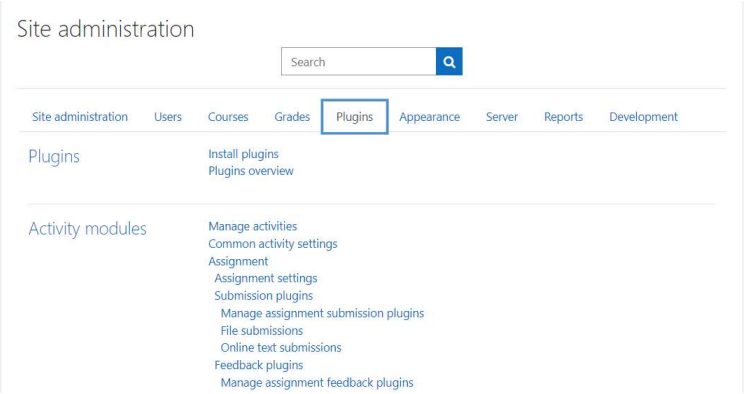


2 - Instalação via gerenciador de plugins

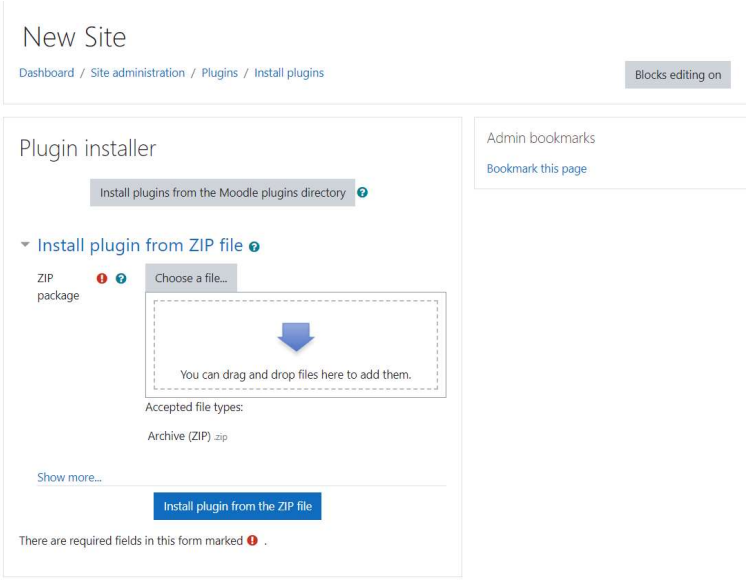
Para efetuar a instalação deste plugin o administrador do sistema deve acessar seção de "Administração do Site". A versão do Moodle necessária para o prosseguimento desta operação é a 3.11 .



Após isso, selecione a aba "Plugins".



Avançe para "Install plugins" (ou sua versão localizada, em caso da instância estiver sob alguma outra linguagem). Uma nova tela para a submissão de arquivos aparecerá.



Com isso basta anexar o arquivo, que fora feito o download previamente, na seção "Install plugin from ZIP file".

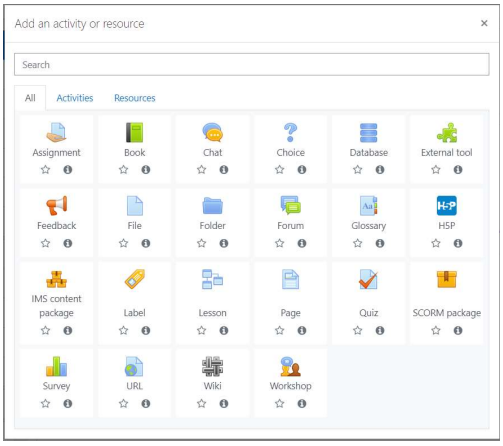
Após isso basta acionar o botão "Install plugin from the Zip file". Caso ocorra algum erro, verifique as permissões do usuário. Em caso de sucesso, um pequeno relátorio de alterações aparecerá.

Com isso, o plugin FLWarrior foi instalado ao Moodle.

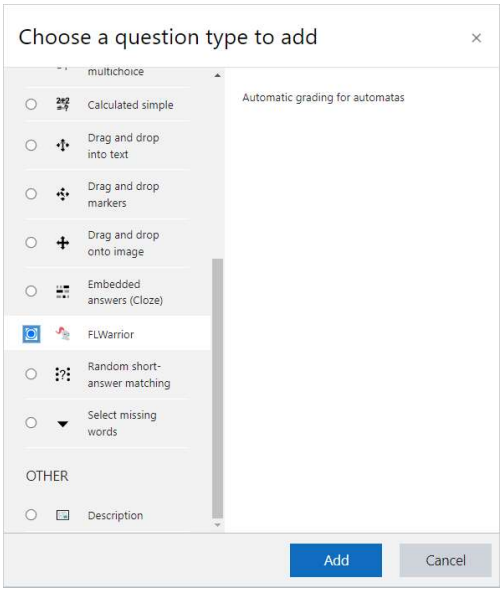
C.2 MANUAL DO PROFESSOR

Criando Questões do Plugin FLWarrior

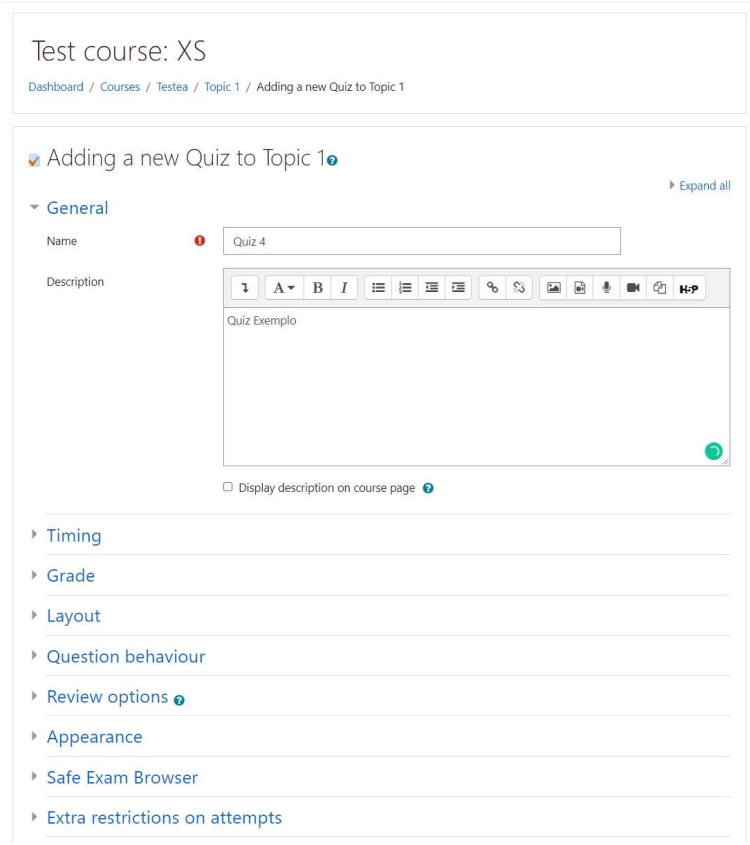
Para isso o professor deverá ter permissão de edição do curso. O professor deverá criar uma nova atividade do tipo **Quiz**.



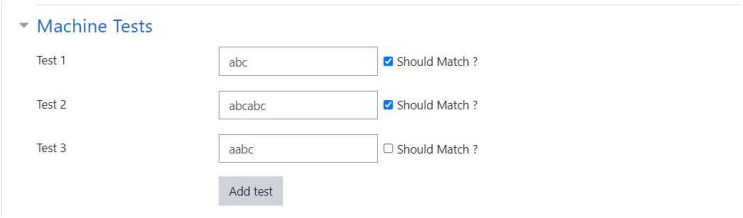
Preeencha normalmente a estrutura do quiz. Configurando



Ao criar uma nova questão, defina o tipo de questão como sendo do tipo FLWarrior.



No segmento de Machine Tests, defina os casos de teste para a linguagem requerida na questão, definindo a sentença a ser verificada e se ela deve ser aceita ou não pela máquina. Vale ressaltar que existe um limite máximo de iterações que a máquina pode fazer.



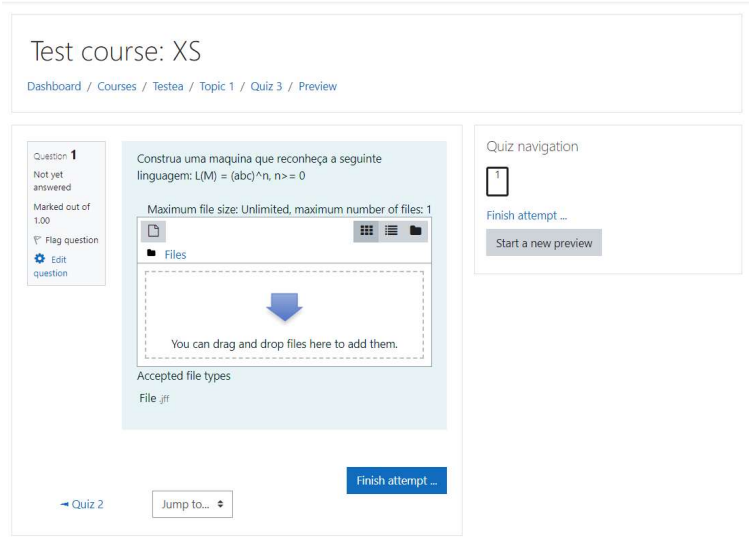
Após isso, basta publicar o quiz como qualquer outro dentro do Moodle.

C.3 MANUAL DO ALUNO

Respondendo Questões do Plugin FLWarrior

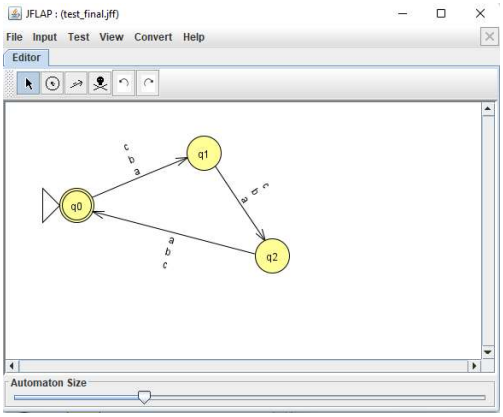
Etapa 1 - Iniciar Questão

Como aluno, acesse o quiz o qual deseja responder. Após isso, vá até a questão correspondente. A seguinte tela se apresentará.

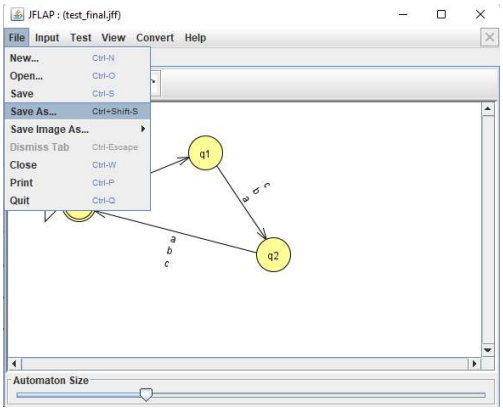


Etapa 2 - Responder a Questão

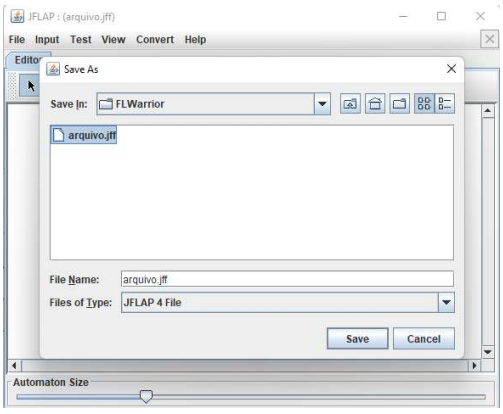
O aluno pode decidir construir o arquivo manualmente ou utilizar alguns construtor compatível com o formato JFLAP 4. Neste caso utilizou-se a ferramenta JFLAP.



Para exportar a máquina, vá em *File* > *Save As*

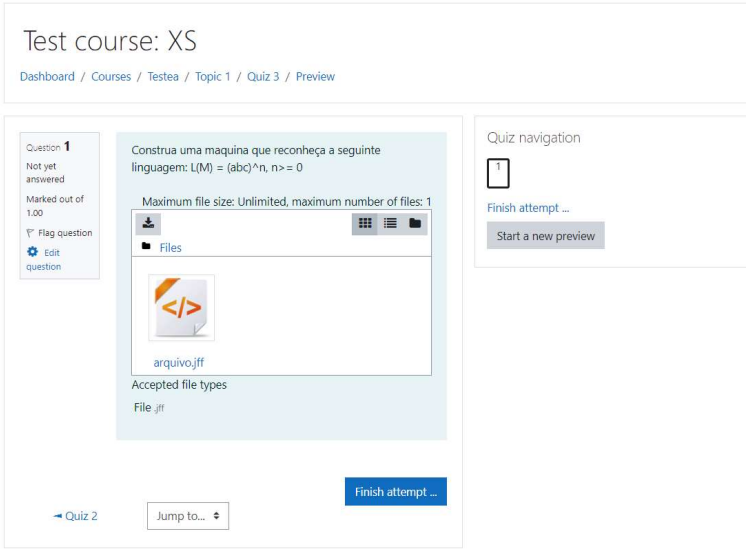


Após isso, selecione o local de exportação



Etapa 3 - Submeter a Questão

De volta a questão, clique em anexar e selecione o arquivo ou arraste-o para dentro da caixa indicada. O arquivo deverá aparecer onde estava a caixa.



Basta agora continuar o quiz, respondendo outras questões e ao final, submeta todas as respostas.

Etapa 4 - Revisão

Uma tela contendo a nota e a revisão, se configurado pelo professor no quiz, deverá aparecer.

Test course: XS

Started on	Thursday, 24 February 2022, 10:34 PM
State	Finished
Completed on	Thursday, 24 February 2022, 10:40 PM
Time taken	5 mins 56 secs
Marks	1/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1 out of 1.00

Flag question

Edit question

Construa uma maquina que reconheça a seguinte linguagem: L(M) = (abc)^n, n>= 0

Maximum file size: Unlimited, maximum number of files: 1

Files

arquivo.jff

Accepted file types

File .jff

Finish review

Quiz 2

Jump to...

Quiz navigation

1

Finish review

Start a new preview