https://eventos.utfpr.edu.br//sicite/sicite2020

Plataforma para submissão e correção automática de exercícios de programação

Platform for automatic submission and correction of programming exercises

João Augusto Grobe Castilho
joaocastilho@alunos.utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

Erikson Freitas de Morais
emorais@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

Iara da Cunha Ribeiro da Silva
iarasilva@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Ponta Grossa, Paraná, Brasil

RESUMO

Este trabalho visa apresentar o processo de desenvolvimento de uma ferramenta para permitir a submissão de algoritmos e automatizar a correção dos códigos fonte de acordo com uma bateria de testes. A ferramenta permite possibilidade de expansão para diversas linguagens de programação, e com suporte inicial em Matlab ou Octave, testado em turmas da disciplina de Cálculo Numérico. Assim facilitando trabalho do professor, otimizando seu tempo, e simplificando a comunicação de feedback para os alunos. No decorrer do trabalho serão descritos os módulos que compõe a plataforma desenvolvida, bem como as ferramentas utilizadas para seu desenvolvimento. A aplicação se mostrou positiva em alguns aspectos, porém em outros apresenta indícios de mudanças e melhorias sugeridos para trabalhos futuros.

PALAVRAS-CHAVE: Tecnologia educacional. GNU Octave (Programa de computador). MATLAB (Programa de computador)

ABSTRACT

The goal of this paper is to present the process of developing a tool that allows the submission of algorithms and automates the correction of source codes according to a series of tests. This tool allows the possibility of expansion to several programming languages, and it has initial support to Matlab or Octave, tested in Numerical Calculus classes. Thus facilitating the work of the professor, optimizing their time, and simplifying the communication of feedback to students. In this paper the modules that we used to develop platform will be described, as well as the tools used for its development. The application has been positive in some aspects, but in others it shows signs of changes and improvements suggested for future work.

KEYWORDS: Educational technology. GNU Octave (Software). MATLAB (Software).

Recebido: 19 ago. 2020.
Aprovado: 01 out. 2020.

INTRODUÇÃO

O Cálculo Numérico é uma disciplina ofertada nos cursos de graduação de engenharias e ciência da computação, mescla a área de matemática e da computação que estuda, analisa e implementa algoritmos numéricos para resolver problemas modelados matematicamente (CARRANZA, 2014). Uma das linguagens mais populares utilizadas nessa disciplina é o Matlab ou Octave (ATKISON, 2020).

Os docentes que ministram disciplinas que envolvem programação de computadores utilizam plataformas que auxiliam o gerenciamento de exercícios computacionais, bem como as submissões de códigos-fonte desenvolvidas pelos discentes, como o Moodle ou sistemas internos da própria universidade. Entretanto, poucas dessas plataformas oferecem ferramentas para a correção de exercícios computacionais.

As ferramentas de submissão e correção existentes, geralmente são desenvolvidas para o uso interno das universidades, como o SuSy, desenvolvida pela UNICAMP (UNICAMP, 2017).

Diante disso, foi desenvolvido uma ferramenta chamada Octave Correction Tool (SILVA, 2020) que apenas corrige as rotinas implementadas em Matlab ou Octave pelos alunos. Essa ferramenta automatizou a correção efetuada pelo professor e diminuiu o tempo de entrega das atividades computacionais corrigidas.

Nessa ferramenta citada anteriormente, o professor era responsável por fazer o download de cada uma das submissões dos seus alunos, separá-los em uma pasta de acordo com suas respectivas turmas e cadastrar testes através de um arquivo de texto estruturado ou pela aplicação em linha de comando. Esse processo gerava dificuldades para gerenciar turmas e exercícios, além de não prever a possibilidade de um código malígno ser enviado pelo aluno, podendo prejudicar o computador do mesmo.

O OCSS (Octave Correction Submission System) é um sistema de submissão de atividades computacionais com ênfase na correção e no feedback rápido para o aluno, além de uma correção imparcial, otimizando o tempo do professor para outras rotinas em sala de aula.

MATERIAIS E MÉTODOS

Este trabalho abrange o desenvolvimento de alguns dos módulos que compõem a plataforma OCSS. Outros módulos estão sendo desenvolvidos por outros integrantes do projeto. Os módulos desenvolvidos pelo autor são responsáveis por criar um ambiente virtual onde o professor possa gerenciar turmas, disciplinas e matérias, além de ter informações sobre o desempenho dos seus alunos; E para os alunos, é possível enviar e acompanhar atividades e correções das atividades submetidas anteriormente.

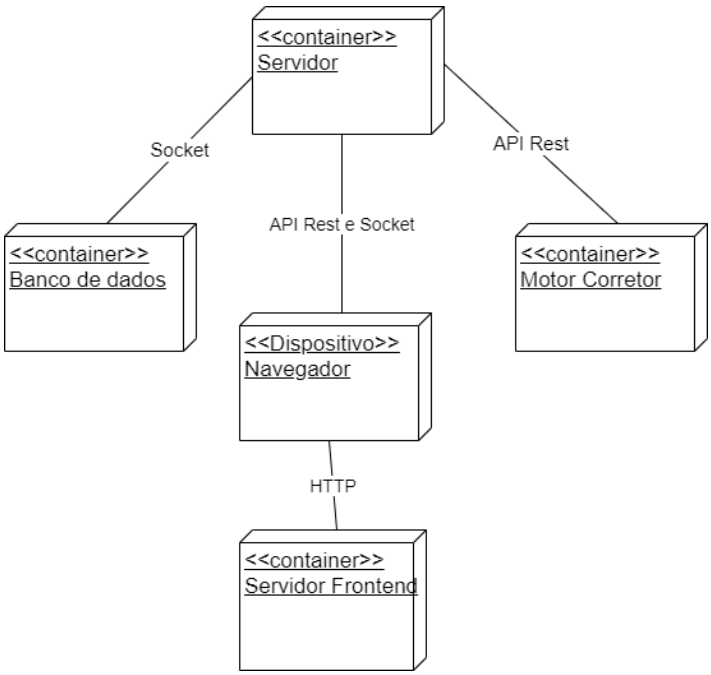
A plataforma completa prevê um módulo responsável por avaliar a submissão feita pelo aluno. Os detalhes do módulo citado estão fora do escopo deste relatório, porém o mesmo conta com detalhes de como a integração foi feita a fim de efetuar a avaliação.

Um container Linux é um conjunto de um ou mais processos organizados isoladamente do sistema. Todos os arquivos necessários para executá-los são

fornecidos por uma imagem distinta (RED HAT, 2020). Ao executar softwares em um container, os processos têm isolamento em nível de rede, memória e disco.

O desenvolvimento da aplicação contou com um ambiente baseado em containers através do Docker. Com essa divisão em containers são evitados conflitos entre versões de dependências dos aplicativos necessários para a aplicação, pois as dependências são empacotadas juntamente com a aplicação na imagem do container (MICROSOFT, 2018), um ambiente isolado e configurável para novos contribuidores do projeto, ajudando com uma melhor experiência de desenvolvimento (RED HAT, 2020). Cada container do diagrama de implantação, apresentado na Figura 1 é descrito subsequentemente.

Figura 1 – Diagrama de implantação



Fonte: Autoria própria

No contexto deste trabalho usaremos os termos cliente e servidor. O servidor é um container responsável por fornecer uma interface para acesso de leitura e escrita aos dados do sistema para um cliente. Esse cliente pode ser um navegador, aplicativo ou qualquer ferramenta que consiga realizar requisições HTTP ao servidor.

O servidor é responsável pela maior parte de todas as regras do sistema desenvolvido. Nele são encontradas as classes controladoras de cada um dos modelos da aplicação, conexão com o container de banco de dados e conexão com a máquina responsável por fazer a execução e comparações das submissões feitas pelos alunos.

API é um conjunto de definições e protocolos usados no desenvolvimento e na integração de software de aplicações. API é um acrônimo em inglês que significa interface de programação de aplicações (RED HAT, 2020).

Para o desenvolvimento da plataforma é utilizado uma API REST (RESTFULAPI, 2020), que expõe um conjunto de rotas para o cadastro e a obtenção de informações do banco de dados, tratando as requisições devidamente para validar os dados e controlar o acesso impedindo, por exemplo, que um aluno tenha acesso a informações de outros alunos. Também são encontradas rotas para a submissão de arquivos (resoluções enviadas pelos alunos), que são salvos em um subdiretório do servidor.

Esse servidor é desenvolvido em JavaScript e Node.js, utilizando o framework Express, que fornece um conjunto robusto de ferramentas para o desenvolvimento de APIs e servidores.

Uma rotina importante do servidor é gerenciar uma fila de correções. A correção não pode ser efetuada enquanto a requisição de submissão do aluno é feita, pois essa rotina pode levar certo tempo, possibilitando a conexão expirar e o usuário ficar sem feedback. Por conta disso, quando uma nova submissão é enviada pelo aluno, uma instrução de correção é inserida em uma fila de execuções, passada para o Motor Corretor (Item 2.5), e um *socket* é aberto para a comunicação entre o servidor e cliente.

A comunicação da API REST usada na plataforma se baseia em protocolos HTTP, desta forma, a comunicação é sempre inicializada pelo lado do cliente. Por conta disso, o servidor precisa expor um servidor websocket, que se trata de uma comunicação TCP, onde dados podem ser transmitidos a qualquer momento pelo servidor ou pelo cliente.

Durante a etapa de autenticação o servidor gera um *token*, identificando o usuário e sua função (aluno ou professor). Esse token é um Json Web Token, que é uma forma do servidor realizar uma assinatura digital que pode ser visualizada pelo usuário, mas não pode ser alterada. Esse *token* deve ser usado pelo frontend para efetuar as futuras requisições que requerem autenticação.

O frontend é a parte em que é executada do lado do cliente, ou seja, pelo navegador do usuário. Essas páginas tem acesso à API REST através de requisições HTTP, exposta pelo backend para fazer a requisição e submissão dos dados de acordo com cada página.

O fronted foi desenvolvido utilizando o framework Vuejs, uma ferramenta popular para o desenvolvimento de interfaces (GITHUB, 2020), componentes e páginas web responsivas, ou seja, que se adaptam em qualquer tipo de dispositivo, e com uso da biblioteca Vuetify (VUETIFY, 2020) é possível utilizar os componentes, botões e ícones do Material Design, padrão adotado por grandes empresas e desenvolvido pela Google (MATERIAL, 2020).

Esse container expõe uma SPA (*Single Page Application*) na porta 80, porta padrão para sites. Essa página tem scripts para carregar o conteúdo dinamicamente através das requisições da API, preenchendo a página com informações do aluno/professor, exercícios e entre outros dados.

Através do token de autorização, comentado na seção anterior, o frontend tem como renderizar conteúdo específico para o cargo do usuário, e se essa informação for alterada, o servidor consegue bloquear as requisições devido ao token inválido.

O conteúdo gerado para o professor tem sessões para a administração de matérias e exercícios. Dentro da administração de exercícios é possível configurar testes públicos e privados. Os testes públicos tem os parâmetros de entrada e saída visíveis para o aluno, permitindo que ele faça testes locais com os mesmos valores. Já os testes privados são fechados para o aluno, pois ele não tem acesso aos parâmetros de entrada e de saída. Após a submissão do exercício, execução e avaliação pelo servidor, o aluno tem o feedback de todos os testes.

O banco de dados escolhido foi o MongoDB. O principal motivo da sua escolha foi o uso de Documentos para armazenamento de dados. Documentos MongoDB (MONGODB, 2020) são compostos por pares de chave e valor em documentos BSON. BSON é uma representação binária de documentos JSON (JavaScript Object Notation), embora contenha mais tipos de dados.

Por conta do MongoDB utilizar uma notação parecida com um objeto JavaScript é possível utilizar as mesmas abstrações da linguagem de programação no banco de dados.

Para o armazenamento de arquivos, como exercícios submetidos pelos alunos, o servidor armazena localmente e salva o diretório do arquivo no banco de dados.

Em alguns pontos desta sessão serão citados a ferramenta de escore. Tal ferramenta compõe o que está sendo desenvolvida por outro integrante do projeto, por este motivo, detalhes de sua implantação não serão citados neste artigo.

Por conta da sensibilidade de executar um código desconhecido, submetido por um aluno, esse código não deve ser executado no mesmo container do servidor por questões de segurança. Por isso, um container é preparado contendo apenas os programas e arquivos necessários para a execução e comparação. Dessa forma, o código sempre é executado em um ambiente seguro e isolado, garantindo que a execução da resolução não terá acesso a submissões de outros alunos ou da máquina hospedeira.

Além do isolamento, o container pode ser configurado para rodar em uma máquina diferente do servidor, podendo ter especificações para a otimização do processamento ou I/O. Independentemente de onde esse container rode, ele irá executar em uma thread independente do servidor, não bloqueando a chegada de novas requisições.

Considerando que o sistema de correções automáticas poderá ser usado para corrigir rotinas de diferentes linguagens de programação, cada linguagem de programação deve ter seu próprio motor corretor e o programa corretor responsável por executar o código fonte e realizar o escore da atividade de acordo com a saída gerada pelo código e a saída esperada pelo teste. O motor deve estar configurado com os softwares para efetuar a compilação (se necessário) e a execução do código do aluno.

Ao iniciar esse container, parâmetros de configurações para o programa corretor devem ser informados. Nesses parâmetros são definidos qual a operação se quer fazer:

Executar: solicita execução de um código de acordo com parâmetros de entrada.

Os parâmetros para a operação são:

a) diretório do código;

b) diretório de um arquivo descrevendo os parâmetros de entrada;

c) diretório onde serão salvas as saídas.

Comparar: solicita a comparação do arquivo de saída do aluno com a saída do teste, informada pelo professor. O programa de escore é responsável por atribuir uma nota de 0 até 10 de acordo com a semelhança entre os dois arquivos.

Os parâmetros adicionais para a operação são:

d) Diretório do arquivo de saídas do aluno;

e) Diretório do arquivo de saídas do teste.

Ao final de qualquer operação o programa faz uma chamada a API REST informando que a operação foi efetuada com sucesso/falha e enviando os dados referentes a operação. Com isso o servidor sabe o estado da correção e consegue dar um feedback para o aluno.

RESULTADOS E DISCUSSÕES

A plataforma se mostrou efetiva em termos das rotinas de gerenciamento de exercícios e matérias, mas o desempenho da correção e nota estão completamente atrelados a ferramenta de escore, que por sua vez está atrelada ao motor corretor. O professor tem capacidade para fazer correções manuais e alterar as notas atribuídas pela ferramenta de escore, tarefa que pode ser feita caso a mesma apresente alguma anomalia e gere uma nota não condizente com a submissão do aluno.

Na disciplina de cálculo numérico, problemas envolvendo matrizes são recorrentes, o que dificulta ainda mais o desenvolvimento de uma ferramenta de escore acurada. Por exemplo, em uma matriz de ordem grande, um erro pequeno em cada um dos fatores pode revelar uma enorme diferença no resultado final.

Ainda sobre matrizes, a plataforma não consegue exibir de forma amigável matrizes no cadastro de testes. Em Matlab ou Octave matrizes podem ser descritas através de uma cadeia de números e separadores de linha e coluna e isso gera confusão no momento de visualização através da plataforma. Isso poderia ser melhorado adicionando suporte a visualização de matrizes e vetores, de forma parecida como o Octave implementa.

Outro ponto a discutir é em relação a resposta de cada teste. Atualmente, o professor é obrigado a colocar a saída esperada para o teste de acordo com a entrada fornecida, isso faz com que strings de tamanhos variáveis sejam salvas no banco de dados. Os testes com um output muito grande, uma matriz de grande dimensão, por exemplo, pode gerar uma string muito longa para ser salva no banco de dados, tornando a execução mais lenta que o esperado. Uma possível alternativa para isso seria o armazenamento de um código fonte gabarito para a resolução do exercício, o que gera mais uso de processamento, pois esse output é necessário ser executado para a comparação das submissões dos alunos.

A importância de testes privados é um ponto chave na correção do exercício. Se todos os testes forem públicos aos alunos, o mesmo poderia implementar um código extremamente específico para resolver apenas aqueles casos apresentados. Com testes privados, o aluno é forçado a escrever suas rotinas de forma genérica, que atenda casos não apresentados pelos testes públicos, que serão usados apenas como guia para chegar em uma solução para o problema.

O uso de containers para a correção permite que a plataforma consiga escalar em termos de capacidade de processamento e memória para os testes, além de permitir o acoplamento de outras linguagens sem muitas alterações na plataforma. Um ponto fraco do sistema é em relação a fila de correção. Uma possível forma de melhorar isso seria o uso de um banco de dados em memória, como o Redis, aplicação de alto desempenho que pode ser usada para gerenciar filas e que permite a inclusão de observadores, aguardando novas correções na fila para executar as correções, sendo possível ter diversas correções em paralelo.

Além disso, configurações de escalabilidade, como por exemplo, quantos containers de correção podem estar ativos simultaneamente poderiam ser disponibilizadas para melhor o ajuste entre o tempo de espera do aluno na fila de correção e o custo do poder de processamento.

CONCLUSÕES

Dentre as inúmeras plataformas de submissões de exercícios, poucas têm suporte a execução e correção do teste com feedback ao aluno. A plataforma conseguiu alcançar esse objetivo, criando um ambiente em que o professor pode gerenciar matérias e exercícios e o aluno consegue se cadastrar em disciplinas e submeter resoluções para os exercícios. Em relação a nota atribuída para a submissão, dependemos totalmente de um módulo externo, desenvolvido por outro integrante do projeto, capaz de fazer a avaliação da submissão de acordo com testes previamente informados.

Com uma ferramenta de escore acurada em conjunto a plataforma, é possível fornecer um feedback ao aluno de forma rápida, confiável e imparcial, tirando a carga da correção manual do professor. Para chegar a isso, a modularização da plataforma em containers facilita essa etapa, visto que esse módulo pode ser alterado facilmente, desde que as entradas e saídas estejam padronizadas entre as ferramentas de score.

REFERÊNCIAS

ATKISON, K. E. Numerical a sis. Encyclopedia Britannica, Chicago. Disponível em https://www.britannica.com/science/numerical-analysis. Acesso em: 8 de julho de 2020.

CARRANZA, R. R. et al. Numerical Methods: An Online Course. AASRI Procedia, Volume 8, 2014.

GITHUB, Front-end frameworks popularity. **Github**. Disponível em: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190. Acesso em: 23 de julho de 2020.

MATERIAL Design. Material Introduction. **Material Design**. Disponível em https://material.io/design/introduction. Acesso em: 23 de julho de 2020.

MICROSOFT, Introdução aos contêineres e ao Docker. **Microsoft**, 2018. Disponível em https://docs.microsoft.com/pt-br/dotnet/architecture/microservices/container-docker-introduction/. Acesso em: 22 de julho de 2020.

MONGODB manual. Introduction to **MongoDB**. Disponível em https://docs.mongodb.com/manual/core/document/ Acesso em: 8 de julho de 2020.

RED HAT, O que é API?. **Red Hat**, 2020. Disponível em https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces. Acesso em: 22 de julho de 2020.

RED HAT, O que é um container Linux?. **Red Hat**, 2020. Disponível em https://www.redhat.com/pt-br/topics/containers/whats-a-linux-container. Acesso em: 22 de julho de 2020.

RESTFULAPI, What is REST. **restfulapi**, 2020. Disponível em https://restfulapi.net/. Acesso em: 22 de julho de 2020.

SILVA, W. C. et al. Octave Correction Tool: uma Ferramenta de Correção de Rotinas Computacionais para a Disciplina de Cálculo Numérico. Revista Eletrônica de Iniciação Científica em Computação, v. 18, n. 2, 2020.

UNICAMP, Sistema SuSy 9.10 — IC/UNICAMP. **UNICAMP**, 2017. Disponível em https://www.ic.unicamp.br/~susy/. Acesso em 28 de julho de 2020.

VUETIFY, What's the difference?. **Vuetify**. Disponível em: https://vuetifyjs.com/en/introduction/why-vuetify/. Acesso em: 23 de julho de 2020.