

Relatório Parcial de Iniciação Tecnológica

Aluno: Fernanda Sartori Beltrame

R.A.: 22116031-0

Sistema de Correção Automática de Código-Fonte

Período: Setembro/2017 a Agosto/2018 (12 Meses)

Instituição: Centro Universitário da FEI

Orientador: Guilherme Wachs Lopes

São Bernardo do Campo, SP

Fevereiro de 2018

Resumo

A área de ensino a distância tem ganhado muita atenção por parte dos docentes. Um dos principais motivos por essa atenção é que o perfil dos alunos tem mudado de forma acentuada. Os alunos agora passam horas na Internet a partir de qualquer dispositivo, acessando, por exemplo, sistemas de correção de exercícios, a fim de explorar seus conhecimentos. Na área da Computação, muitos desses sistemas são usados até mesmo para correção automática de códigos-fonte. Contudo, um dos principais impedimentos do uso dessas ferramentas está no fato de que a interface para elaboração e uso do professor em sua disciplina não é eficiente. Muitas vezes é necessário que o professor entre em contato com os desenvolvedores da ferramenta para disponibilizar novos exercícios, tornando a ferramenta impraticável. Nesse projeto, é proposto um sistema de correção automática de códigos-fonte de tal forma que o professor possa elaborar seu próprio corretor de maneira prática a partir da comparação entre a saída do sistema construído pelo aluno e a saída esperada.

1 Introdução

Nos últimos anos, uma demanda crescente por métodos de ensino inovadores tem sido criada por diversos motivos. Alguns desses motivos são: gerações de pessoas diferentes, maior relação aluno/professor, alunos já exercendo uma atividade no mercado, falta de tempo do aluno, dificuldade de locomoção em grandes centros urbanos, entre muitos outros.

Alguns desses motivos alavancaram o uso de novas mídias em diversas instituições. Exemplos dessas mídias são: YouTube.com; Twitter; os chamados Learning Management Systems (LMS); entre muitos outros. Contudo, a utilização de cada mídia deve ser feita com muita cautela uma vez que diferentes disciplinas/cursos exigem diferentes formas de uso.

No caso específico da Ciência da Computação, há diversas disciplinas que envolvem o ensino e/ou contato com programação de computadores. Exemplos dessas disciplinas podem ser: desenvolvimento de algoritmos, que utiliza uma linguagem inicial de programação para ensinar lógica; estruturas de dados, que envolve lógica com estruturas fundamentais para elaboração de programas; compiladores, que utiliza programação para in-

interpretar linguagens através de autômatos; inteligência artificial, que utiliza programação para algoritmos de aprendizagem de máquina; entre outras disciplinas importantes para a formação do aluno em Ciência da Computação.

Quando se fala em disciplinas como as citadas acima, há um componente importante a ser observado que está relacionado com o *Feedback* do professor ao aluno. Muitas dessas disciplinas utilizam horas de laboratório para propor o uso da teoria em alguma situação prática. Nessas aulas, o professor é uma peça chave, uma vez que ele deve ter contato individual com os alunos para que todos possam saber onde e porque sua programação está impedindo o êxito do exercício.

Na maioria dessas aulas, é comum ver o professor “visitar” a estação de cada aluno para que ele possa avaliar a situação de cada um. Esse processo é excelente tanto para o professor quanto ao aluno, uma vez que o aluno terá disponível uma ajuda especializada para resolver o exercício e, por outro lado, o professor irá compreender quais são as dúvidas mais frequentes.

Contudo, algumas dessas “visitas” tomam um tempo maior que o esperado e isso pode atrasar o andamento da aula. Além disso, sempre há um conjunto de dúvidas que é frequentemente perguntado ao professor. Um exemplo típico e específico de disciplinas que envolvem programação é o esquecimento por parte do aluno em colocar “;” sempre que uma instrução é escrita na linguagem C. Apesar do aluno saber sobre a regra do “;” é natural esquecê-lo durante a programação. Assim, o professor acaba se tornando um recurso mal utilizado.

Recentemente muitos sistemas computacionais acabaram surgindo para tentar resolver esse problema. Um exemplo é o site <http://www.urionlinejudge.com>, que apresenta uma lista de problemas computacionais classificados em categorias e faz a correção automática para cada aluno [1]. Esses tipos de sistemas apresentam 3 principais vantagens em relação ao sistema de ensino tradicional. A primeira delas é que o aluno tem o retorno imediato sobre a corretude do seu programa. A segunda é que ele pode usar o sistema mesmo não estando na faculdade. E a última é que seu desempenho pode ser medido com o passar do curso.

Apesar desses sistemas de correção automática apresentarem esses benefícios, ainda não se tem até o momento um sistema como esse focado na usabilidade do professor em disponibilizar exercícios e materiais tal como o Moodle. Esse é um importante ponto

negativo que faz com que a implantação desses sistemas se torne inviável. Assim, nesse projeto é proposto um sistema de correção automática de programas computacionais que seja focado não só na utilização do aluno, mas principalmente, na utilização do professor. Esse sistema permitirá hospedagem de arquivos, cálculo de notas e configuração do sistema de correção para cada exercício.

1.1 Objetivo do Projeto

Modelar e implementar um sistema de correção automática de programas focado na utilização do professor. Esse sistema deverá ser implementado de tal forma que novas funcionalidades sejam facilmente incorporadas.

2 Sistemas de Gestão de Aprendizagem (SGA)

Os Sistemas de Gestão da Aprendizagem (*Learning Management Systems*) são popularmente conhecidos como plataformas de e-Learning. O principal objetivo de um SGA é prover conteúdo e ferramentas para medir o conhecimento dos alunos. Este tipo de sistema é uma sub-categoria dos Ambientes Virtuais de Aprendizagem.

Os ambientes virtuais de aprendizagem utilizam nomenclaturas que fazem analogias ao ambiente real. Dessa forma, esses ambientes são constituídos por salas, cursos, material didático, alunos e professores. Contudo, o grande diferencial é que o contato aluno-professor acontece também fora da sala de aula física. Isso significa que o aluno terá mais tempo para estudar e se dedicar às disciplinas. Diversos SGA tem sido desenvolvidos para a área de educação superior para facilitar o processo de aprendizagem. Alguns desses sistemas serão abordados nas próximas seções.

2.1 Blackboard Learning System

Blackboard é considerado um dos mais populares sistemas de aprendizagem web para a educação superior. Um dos principais diferenciais dessa plataforma é que ela é fácil de ser utilizada por professores e alunos. Há ceca de 39.000 professores em mais de 1.350 instituições de ensino que juntas entregam mais de 147.000 cursos pela web para mais de 10 milhões de alunos em 80 países.

Essa plataforma integra muitos recursos de comunicação, tais como imagens, vídeos, audios, de tal forma que torne o entendimento da disciplina mais tangível para o aluno. Os alunos também podem usar a ferramenta para entregar exercícios e criar materiais para enriquecer o conteúdo da matéria. Os professores têm a sua disposição ferramentas de avaliação, monitoramento, interação com o aluno e sistema de notas. Todos esses recursos são protegidos por senhas para que proteger a propriedade intelectual dos professores e privacidade do aluno.[2]

2.2 Moodle Learning System

O *Moodle* é uma plataforma de ensino livre e de código-aberto baseado no modelo construtivo social de pedagogia. O design do *Moodle* efatiza a criação de interação colaborativa e ambientes de ensino online focada no aluno [3, 4].

O *Moodle* é conhecido como um software “criado através da participação ao invés de publicações sucessivas”. Isso significa que a comunidade como um todo acabou interferindo em seu processo de criação. Talvez esse seja a sua maior vantagem e o que permitiu com que tivesse seu foco no aluno.

O *Moodle* é muito similar ao *Blackboard* quando se fala em exposição de materiais didáticos na forma de hypermidia. Contudo, há dois pontos positivos que o *Moodle* apresenta sobre o *Blackboard*: comunidade de programadores e interoperabilidade entre sistemas. Uma vez que o *Moodle* é feito sobre código-aberto, isso permite com que diversos programadores ingressem ao projeto e criem novas funcionalidades, tais como: quiz, questionários, entregas de arquivos, entre outros tipos de recursos. Além disso, o moodle apresenta diversas formas de integrá-lo a outros sistemas.

Nesse projeto, utilizamos o *Moodle* como uma ferramenta base para o sistema de correção automática. Sua utilização será explicada com maiores detalhes na próxima seção.

3 Proposta

Este trabalho propõe um sistema de correção automática de programas computacionais. A ideia geral é que o aluno possa ter um *feedback* imediato (ou o mais rápido possível)

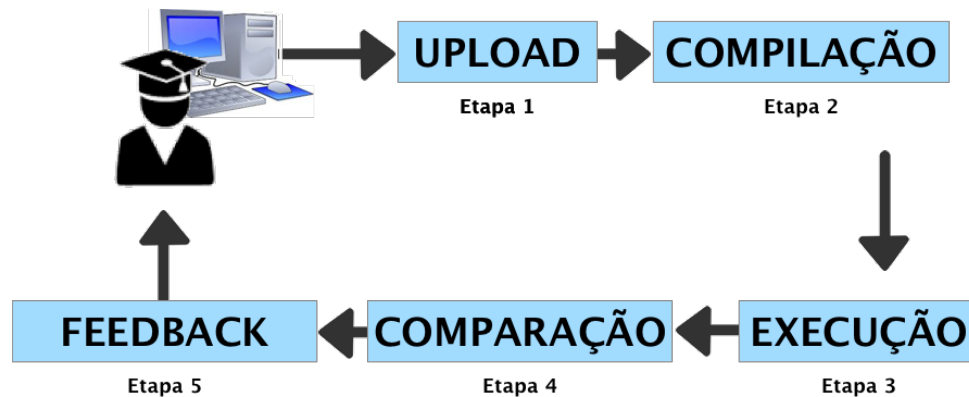


Fig. 1: Etapas do processo de correção automática

sobre a compilação e execução do seu programa tanto em aulas de laboratório quanto em atividades extra classe.

O processo de correção é feito em 5 etapas: Upload, Compilação, Execução, Comparação e Feedback. Essas etapas são ilustradas na Figura 1. A primeira etapa, chamada de *Upload*, é responsável pela interface de comunicação entre o código-fonte do exercício que o aluno resolveu e o sistema de correção. Esse processo irá receber um arquivo como entrada e carregá-lo no sistema. Informações sobre a identidade do aluno, hora do envio e código do exercício do sistema serão armazenados em um banco de dados no servidor.

A segunda etapa, chamada de *Compilação*, é responsável pela compilação do código-fonte em um programa executável. É sabido que algumas linguagem de programação, como *Python*, não necessitam de compilação. Nesse caso, esta etapa será ignorada.

Ainda nessa etapa, caso ocorra um erro de compilação o aluno será imediatamente informado sobre seu erro. Essa mensagem constará das mesmas informações fornecidas pelo compilador, tais como: arquivo, número da linha, número da coluna e tipo do erro gerado. Esse primeiro retorno é muito importante para o aluno e economiza tempo do professor em aulas de laboratório.

A terceira etapa, chamada de *Execução*, irá executar a aplicação do aluno em um *container*; isto é, em um mecanismo de máquina virtual. Nesse processo é fundamental fazer a execução de maneira “isolada” para que o servidor não sofra influências de códigos maliciosos. Para fazer a execução desse programa, o professor deverá cadastrar previamente um arquivo de entrada de dados de tal forma que possa ser enviado do programa

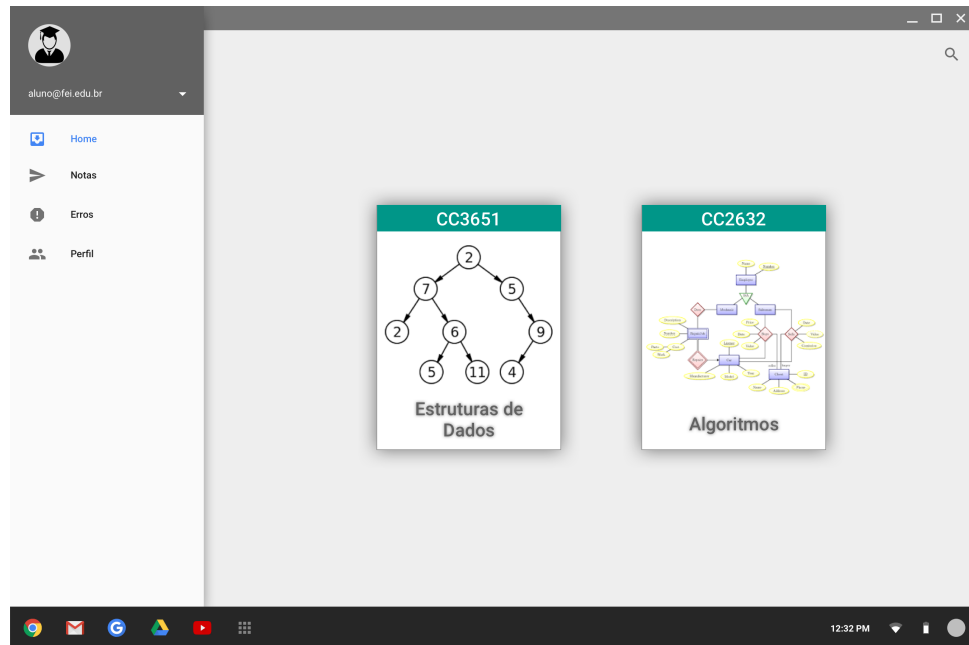


Fig. 2: *Mock-up* da tela de disciplinas

do aluno. O resultado dessa etapa será um arquivo de saída que foi gerado pelo arquivo executável.

A quarta etapa, chamada de *Comparação*, irá comparar a saída gerada pelo arquivo executável com a saída esperada do programa. Caso alguma linha do arquivo de saída não corresponda ao esperado será mostrada uma mensagem de erro informando o número da primeira linha que não correspondeu ao esperado.

A última etapa, chamada de *Feedback*, é responsável por organizar todas as informações de erro e exibir ao aluno. O aluno poderá consultar todos as correções anteriores bem como cada código enviado ao sistema.

3.1 Interface Gráfica

A interface gráfica do aluno será composta por 3 principais telas. A primeira será a tela de disciplinas que irá conter as disciplinas cadastradas no sistema que o aluno cursa (Figura 2). A segunda tela irá mostrar os exercícios por disciplina (Figura 3). E a última tela irá mostrar os envios do aluno de cada exercício (Figura 4).

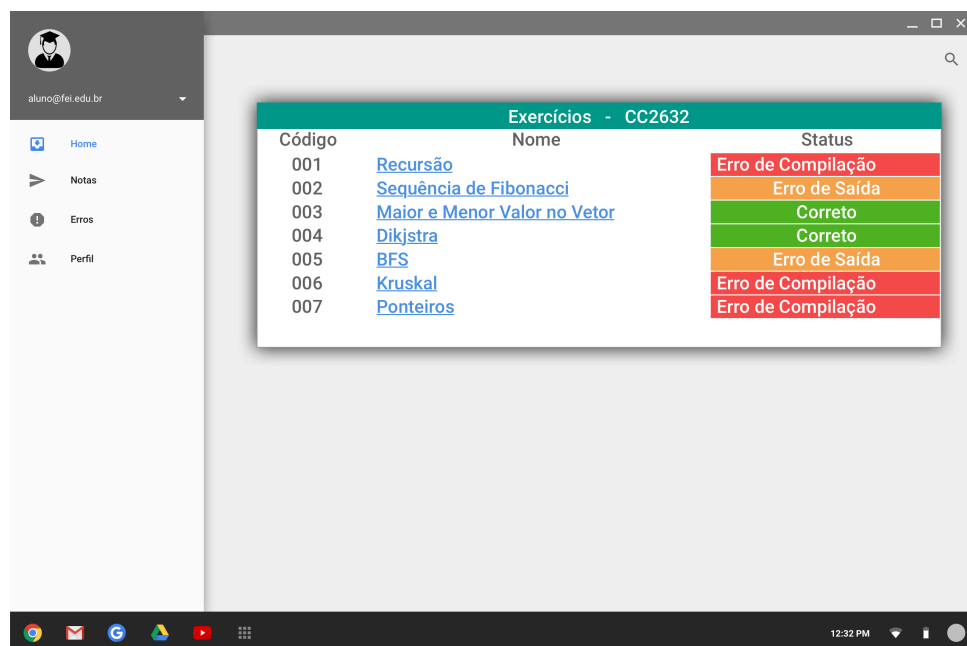


Fig. 3: Mock-up da tela de exercícios

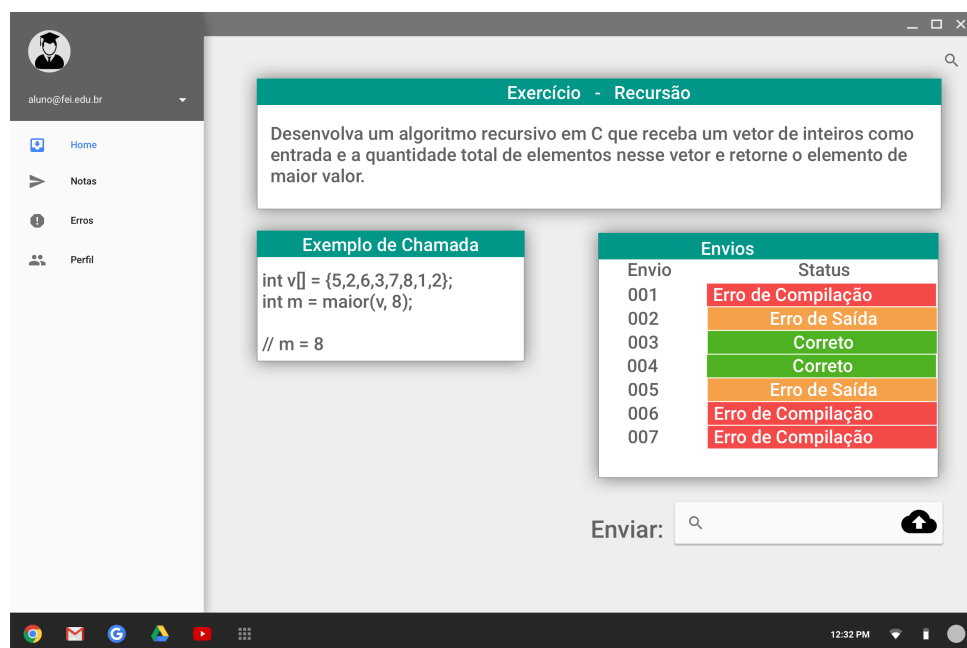


Fig. 4: Mock-up da tela de envios

4 Resultados Parciais

Com o objetivo de se ter uma identidade visual para o sistema, foi proposto o nome "Escape++" e o logo da Figura 6.



Fig. 5: Logotipo do Escape++.

Uma vez que há intenção de se aplicar esse sistema durante as aulas de programação (principalmente C/C++), o logotipo foi pensando seguindo as mesmas ideias visuais estabelecidas para essas linguagens.

4.1 Desenvolvimento das Telas

O desenvolvimento das telas dos alunos foi feito tendo como base a sequência de passos da Figura 1. O objetivo aqui foi fazer uma interface simples e intuitiva seguindo os mesmos moldes do Moodle.

1. Menu lateral:

No Menu Lateral, o usuário tem a possibilidade de “sair”, ir para a página das disciplinas em que está cadastrado, caso for aluno, ou caso professor, ir para seu menu inicial, ir para a página de agradecimentos ou acessar o seu perfil.

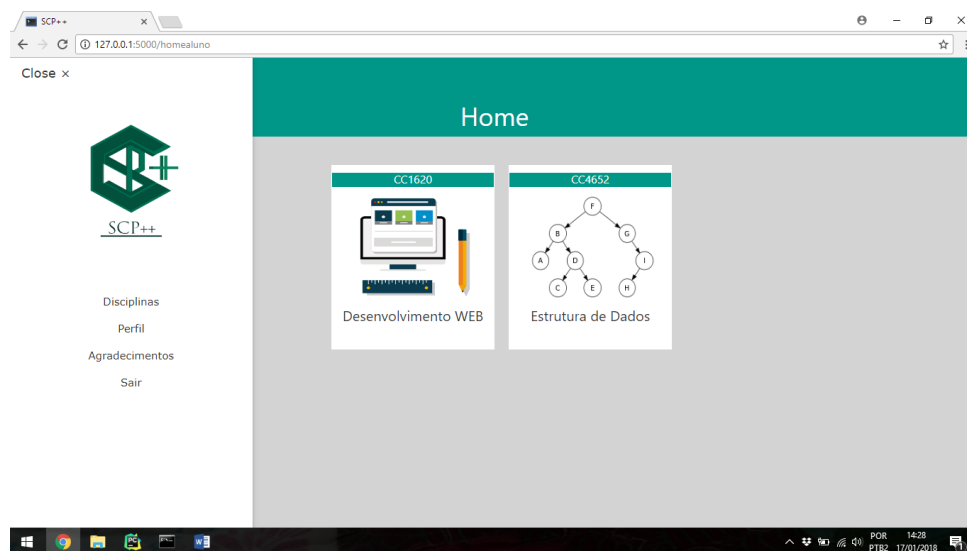


Fig. 6: Menu lateral.

2. Perfil:

O usuário terá acesso ao nome de usuário, seu e-mail e sua senha, podendo alterar, se desejar, sua senha.

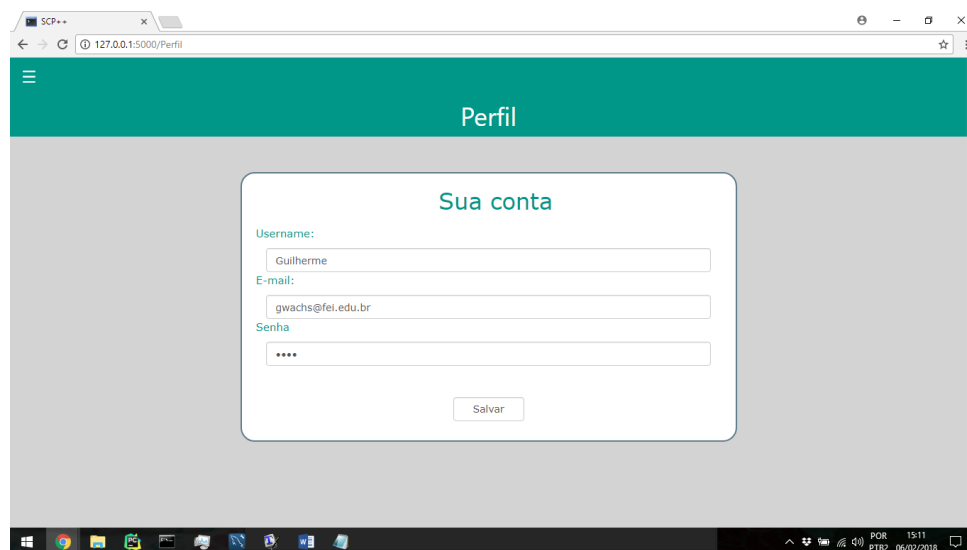


Fig. 7: Tela de acesso ao perfil.

3. Agradecimentos:

Essa página é destinada para agradecer pelo apoio na confecção desse trabalho.



Fig. 8: Agradecimentos.

4.1.1 Alunos

1. Tela de Login:

Os usuários serão encaminhados para essa página para realizar o login e utilizarem o projeto.

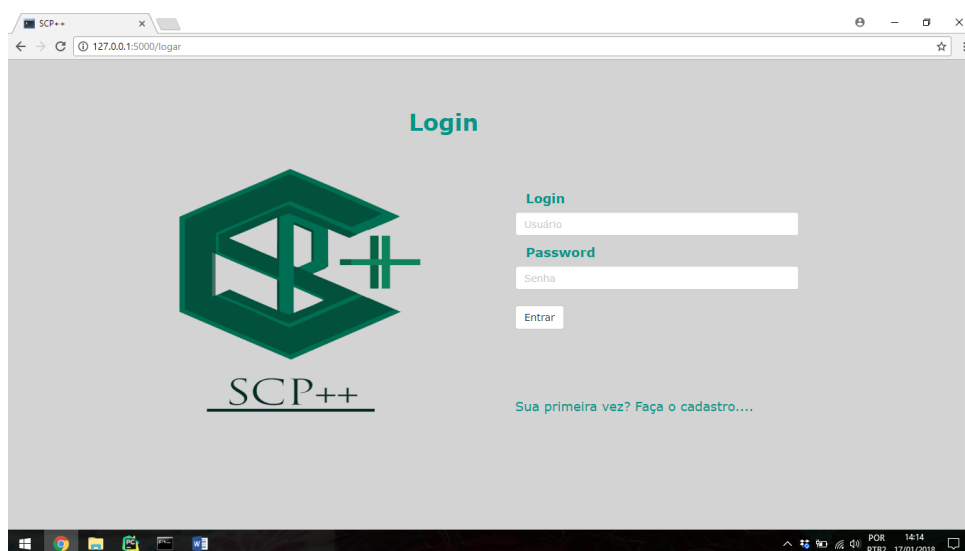


Fig. 9: Tela de Login.

2. Tela de Cadastro:

Na sessão 6.2 será explicado como procederá o encaminhamento dos alunos pelo Moodle. Como não haverá a necessidade de novos cadastros para os alunos, a página de cadastro será destinada ao cadastro de novos professores.

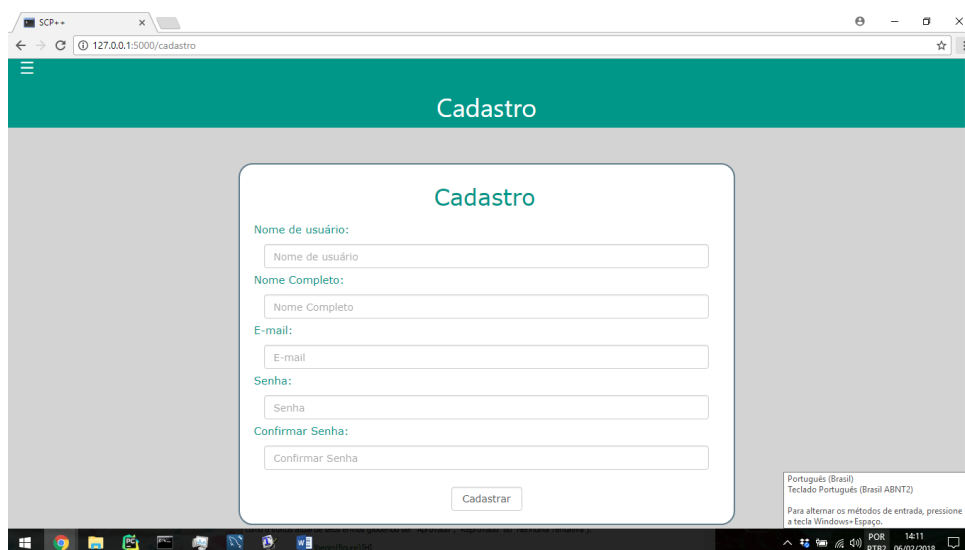


Fig. 10: Tela de Cadastro para novos professores.

3. Portal do Aluno (Home):

Aqui estarão disponíveis as disciplinas que o aluno está inscrito.

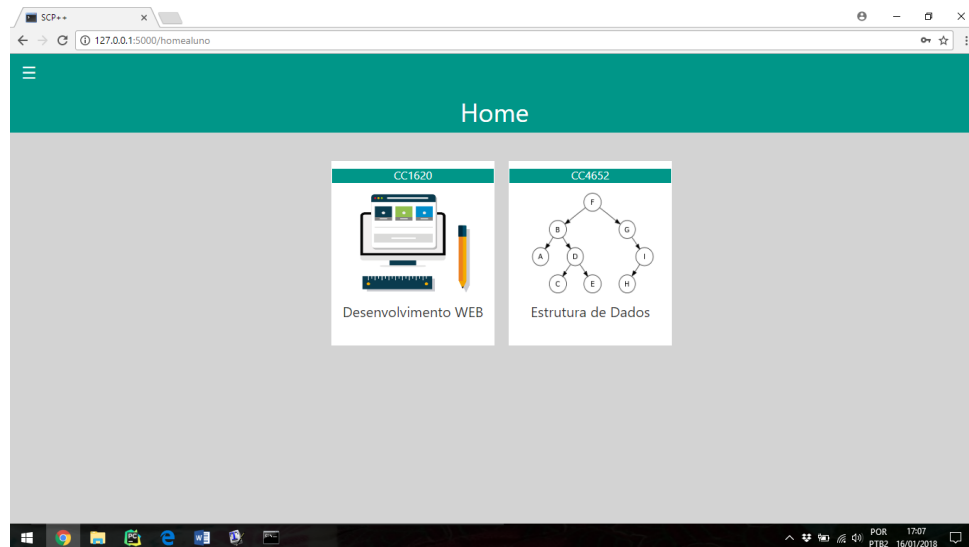


Fig. 11: Tela de Menu Principal do Portal do Aluno.

4. Menu da Disciplina (aluno):

Ao selecionar a disciplina que desejar, serão dispostos os laboratórios disponíveis para serem realizados, bem como o status atual de seus envios (podendo ser “Aprovado”, “Reprovado” ou “Nenhuma Tentativa”).



Fig. 12: Menu da disciplina.

5. Upload de arquivos:

Após selecionado qual laboratório deseja realizar, o aluno será encaminhado para a página de envios referente ao laboratório escolhido. Nessa tela, serão dispostos o enunciado do exercício (disponível em formato PDF), a opção de enviar novas submissões e os status de todas as submissões enviadas, apresentando mensagem informando se foi aprovada ou reprovada, e em caso de reprovação, exibindo a mensagem de erro descrita pelo compilador.

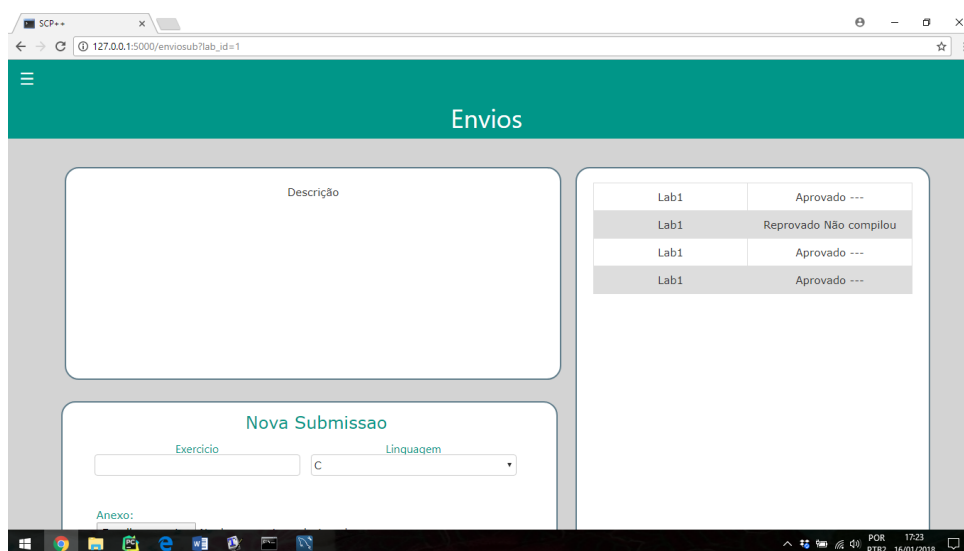


Fig. 13: Tela do exercício destinada ao envio de submissões.

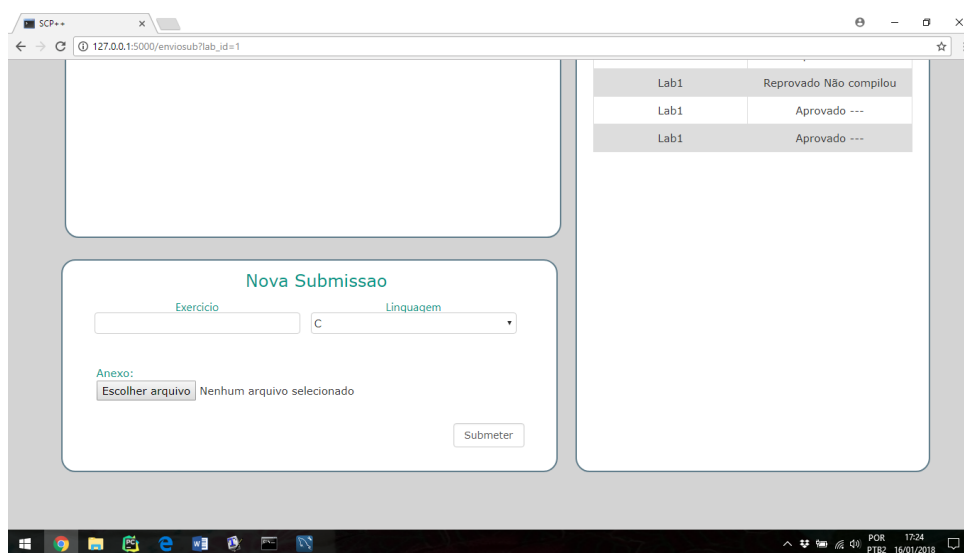


Fig. 14: Tela do exercício destinada ao envio de submissões.

4.1.2 Portal do Professor (Home)

Essa tela oferece três opções para o professor: escolher uma disciplina já cadastrada, criar nova disciplina e convidar novo professor.

Para disciplinas já criadas, são disponíveis três outros recursos: editar disciplina, excluir disciplina e torná-la disponível somente para usuários professores.

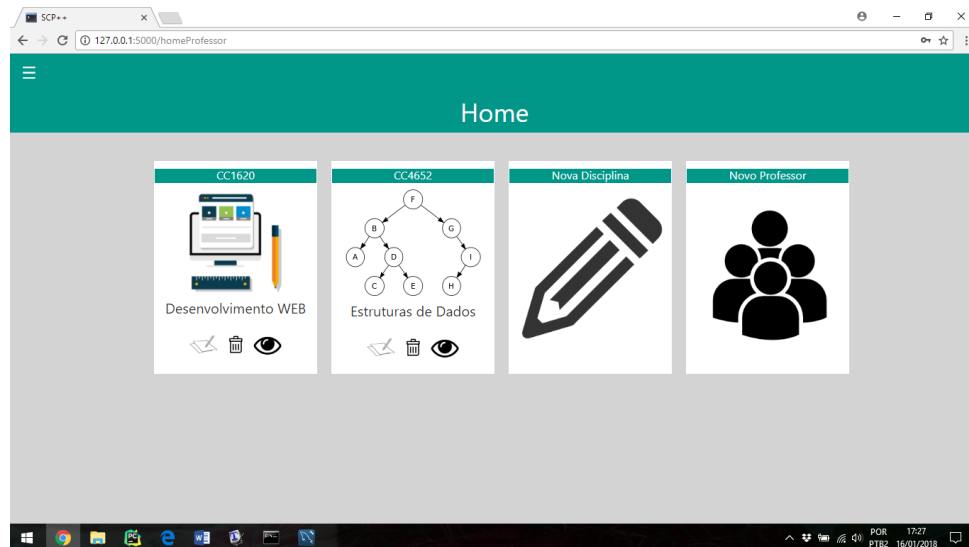


Fig. 15: Menu inicial do Portal do Professor.

6. Edição de Disciplina:

Nessa página, é oferecido o recurso de editar uma disciplina já existente, como comentado no tópico anterior. Através da busca no banco de dados, o código, o título e a chave da disciplina estão disponíveis na tela do usuário. É possível, editar o título referente à disciplina, sua chave de acesso e seu logo. Segue abaixo um exemplo da busca e da página de edição.



Fig. 16: Tela para edição de disciplinas já existentes.

7. Criação de Disciplina:

Aqui está disponível ao professor criar uma nova disciplina, podendo escolher o logo da disciplina, a chave de acesso e o código referente à disciplina.

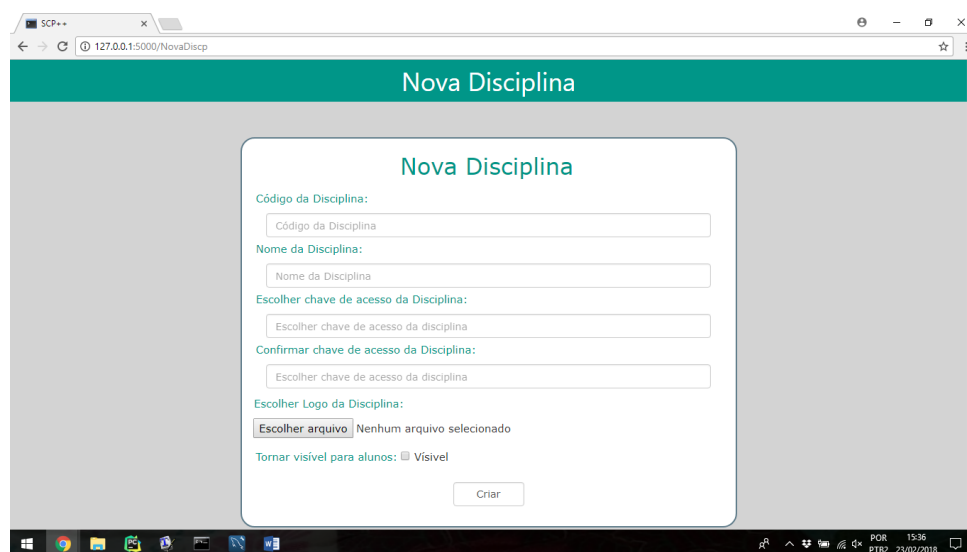


Fig. 17: Tela para criação de nova disciplina.

8. Novo professor:

Nessa página, o professor poderá convidar outro professor para fazer parte do Escape++, enviando-o uma mensagem.

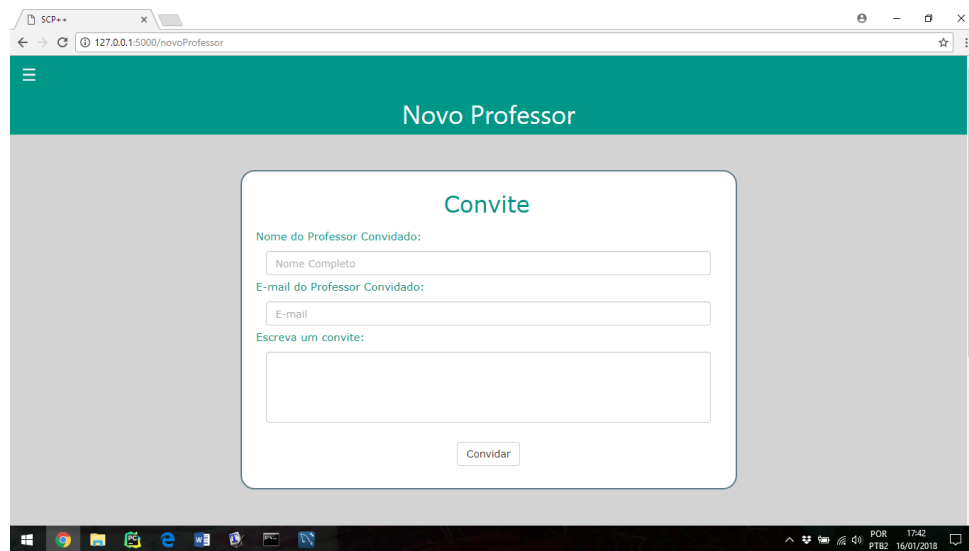


Fig. 18: Tela destinada para convidar um novo professor.

9. Menu da Disciplina:

Após o professor escolher uma das disciplinas já existentes, ele será encaminhado ao menu da disciplina, onde lhe estará acessível os laboratórios dessa disciplina e o relatório sobre a participação dos alunos.

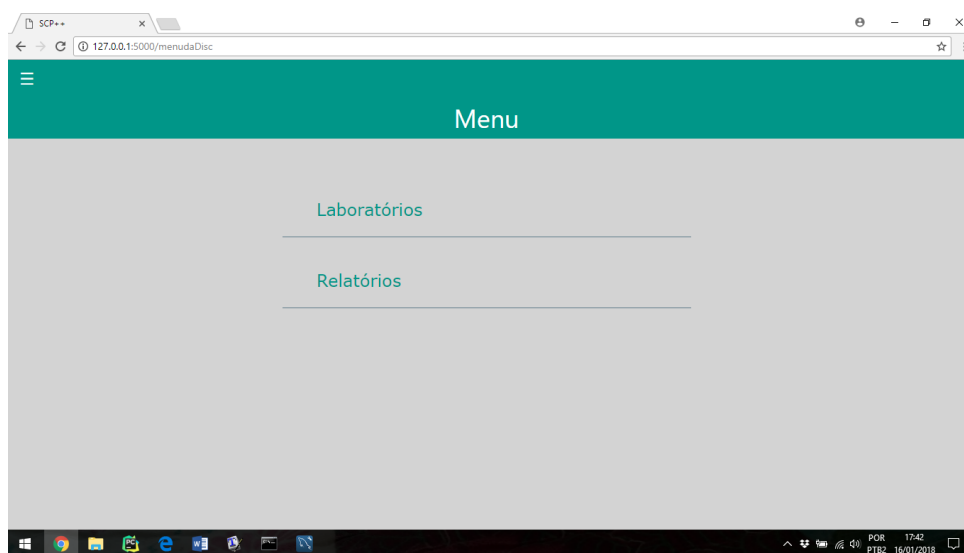


Fig. 19: Menu da disciplina para o Professor.

10. Laboratórios da Disciplina (professor):

O professor poderá criar um novo laboratório, editar, excluir ou tornar indisponível aos alunos um laboratório já existente.

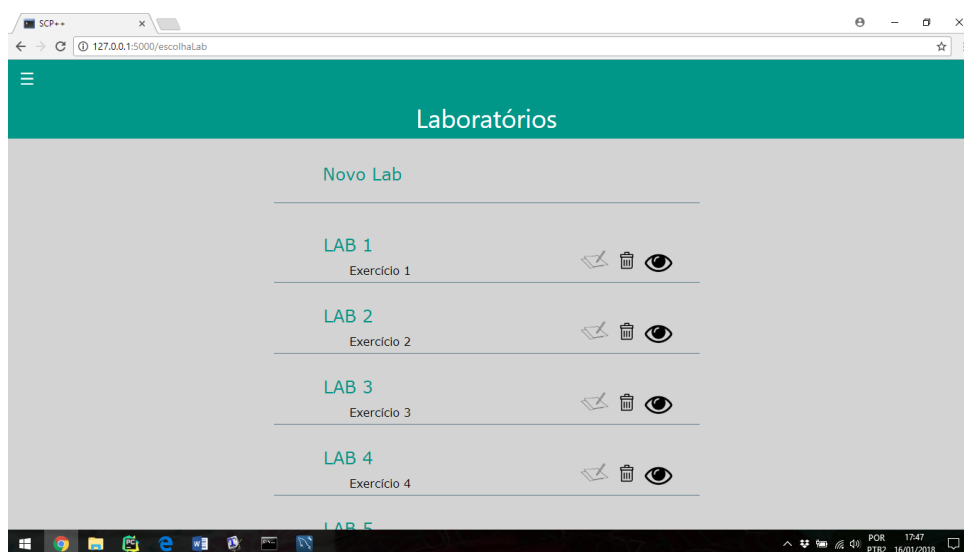


Fig. 20: Menu dos laboratórios.

11. Novo Laboratório:

O professor poderá escolher o título do laboratório, a data máxima de entrega e escolher um arquivo em anexo com os dados do exercício a ser realizado pelos alunos.

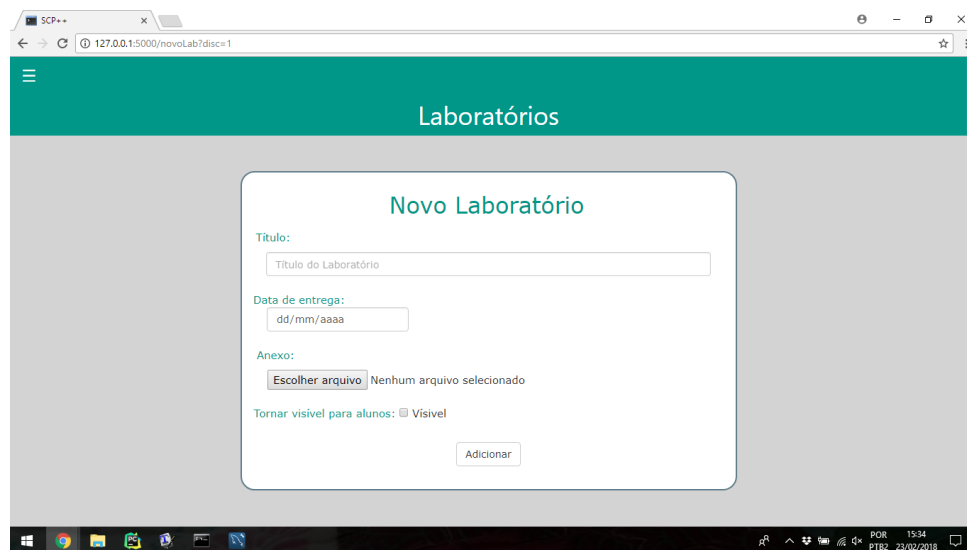
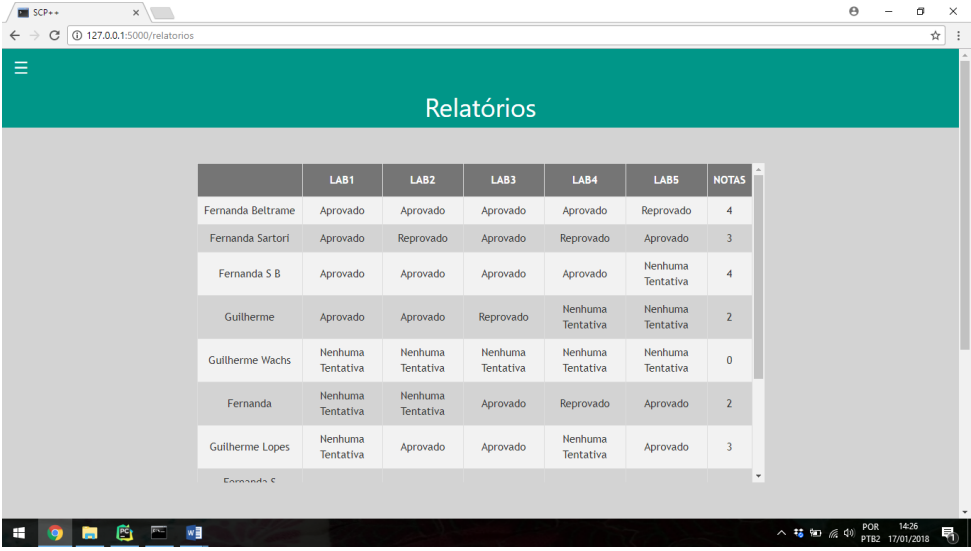
A screenshot of a web browser window showing a form titled 'Novo Laboratório'. The form is centered on a light gray background. It has a teal header bar with the word 'Laboratórios'. The form itself is a white box with a teal title 'Novo Laboratório'. It contains four fields: 'Título:' with a text input field labeled 'Título do Laboratório'; 'Data de entrega:' with a date input field labeled 'dd/mm/aaaa'; 'Anexo:' with a button labeled 'Escolher arquivo' and the text 'Nenhum arquivo selecionado'; and a checkbox labeled 'Tornar visível para alunos:' with the label 'Visível' next to it. At the bottom right of the form is a button labeled 'Adicionar'. The browser's address bar shows '127.0.0.1:5000/novoLab?disc=1'. The Windows taskbar is visible at the bottom.

Fig. 21: Tela para criação de novo laboratório.

12. Relatórios da Disciplina:

Nessa página, o professor poderá acompanhar o processo de desenvolvimento do aluno em sua disciplina. Uma lista contendo os nomes dos alunos inscritos na disciplina e suas participações nos laboratórios, bem como sua nota (baseada na quantidade de laboratórios “aprovado”).



	LAB1	LAB2	LAB3	LAB4	LAB5	NOTAS
Fernanda Beltrame	Aprovado	Aprovado	Aprovado	Aprovado	Reprovado	4
Fernanda Sartori	Aprovado	Reprovado	Aprovado	Reprovado	Aprovado	3
Fernanda S B	Aprovado	Aprovado	Aprovado	Aprovado	Nenhuma Tentativa	4
Guilherme	Aprovado	Aprovado	Reprovado	Nenhuma Tentativa	Nenhuma Tentativa	2
Guilherme Wachs	Nenhuma Tentativa	Nenhuma Tentativa	Nenhuma Tentativa	Nenhuma Tentativa	Nenhuma Tentativa	0
Fernanda	Nenhuma Tentativa	Nenhuma Tentativa	Aprovado	Reprovado	Aprovado	2
Guilherme Lopes	Nenhuma Tentativa	Aprovado	Aprovado	Nenhuma Tentativa	Aprovado	3

Fig. 22: Tela para acompanhamento do desenvolvimento dos alunos.

4.2 Desenvolvimento dos Controllers e API

No projeto, foi utilizado o framework para *Python* chamado Flask [5, 6, 7]. Esse framework tem por objetivo ser simples e fácil de fazer manutenções. O Flask possui interações com o Jinja2 para criação de *views* e pode ser facilmente conectado a diversos Bancos de Dados. Outra característica do Flask é a possibilidade de criação de rotas (URLs) customizadas. O Flask possui seu próprio servidor Web, embora possa ser incorporado como WSGI ou FastCGI.

Na Figura 23, é possível observar a customização da rota. Na URL está disponível o id do laboratório em que o usuário está.

Outro exemplo está disposto na Figura 24, onde é possível, através da URL ter acesso ao id da disciplina e do laboratório que o professor está editando.

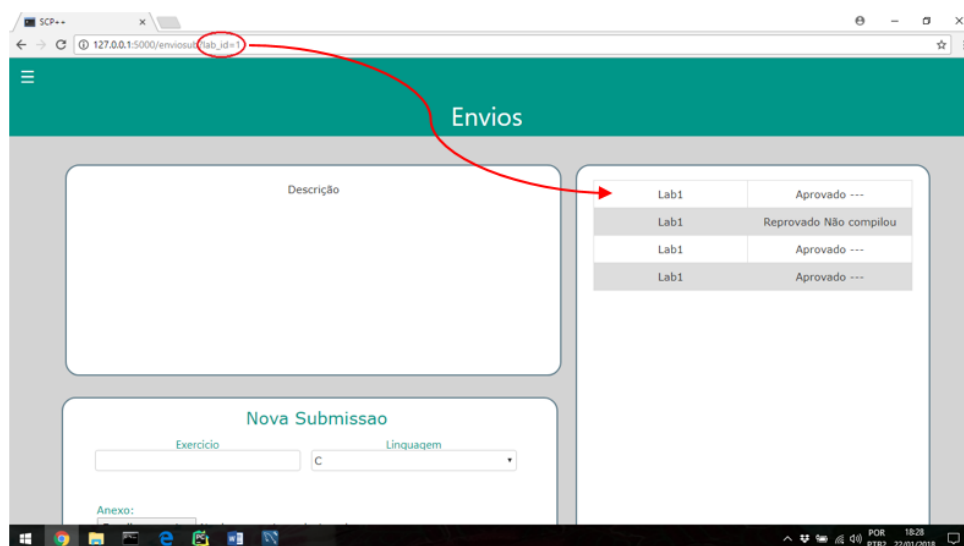


Fig. 23: Conforme o *id* do laboratório disponível na URL, a busca no banco retorna o *status* e a mensagem da submissão.

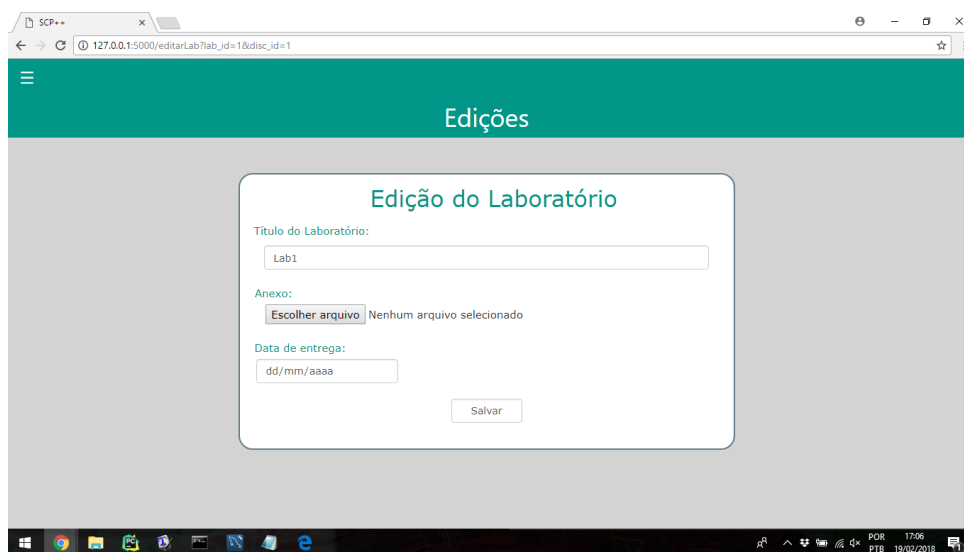


Fig. 24: O *id* da disciplina e do laboratório estão disponíveis para fácil identificação.

Novos professores poderão fazer cadastro no sistema, porém os alunos serão encaminhados pelo Moodle. Uma das características do Moodle é a de estabelecer uma URL única para cada usuário aluno. Assim, é possível que uma URL personalizada seja cri-

ada, e através dela, o aluno será encaminhado diretamente para seu portal inicial. O link <http://127.0.0.1:5000/homealuno?username=uniffbeltrame> é um exemplo.

4.3 Criação do Banco de Dados

Para o armazenamento e gerenciamento dos dados gerados pelo sistema, foi escolhido, como Sistema de Gerenciamento de Banco de Dados, o MySQL. A escolha por essa solução deve-se por três fatores: o MySQL tem uma comunidade ativa e com muitas soluções online, faz parte de Software Livre e possui integração com o Flask através do SQLAlchemy. O SQLAlchemy é uma biblioteca, para *Python*, com comunidade ativa, cujo objetivo é promover acesso ao banco de dados com facilidade e eficiência [8].

Para manuseio do banco de dados e eventuais testes, está sendo utilizado o MySQL Workbench, ainda que seja possível acessar, criar e editar os dados diretamente da IDE chamada PyCharm.

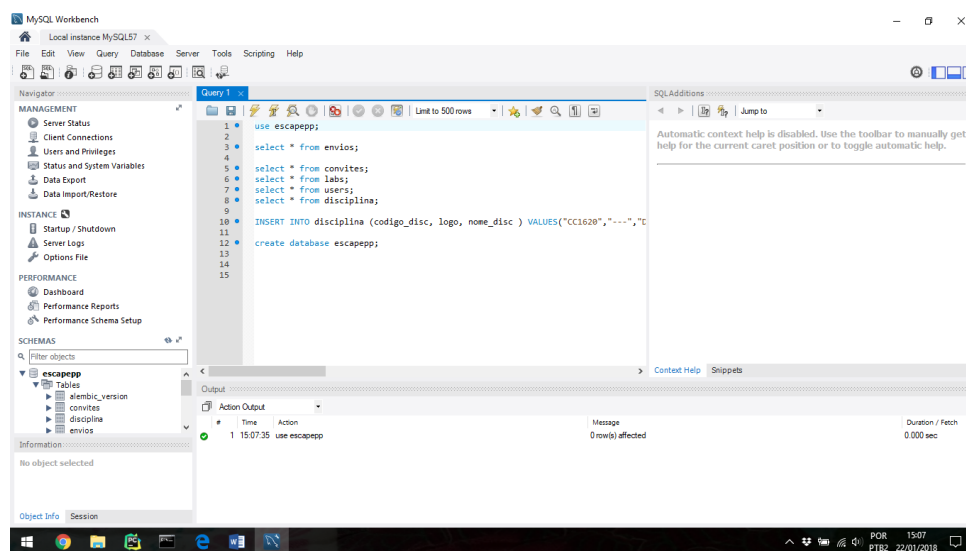


Fig. 25: Exemplo de utilização do SQL Workbench para gerenciar o banco de dados.

Na figura a seguir, está disposto o MER (Modelo Entidade Relacionamento), que representa, de forma abstrata, a estrutura do Banco de Dados utilizada no projeto.

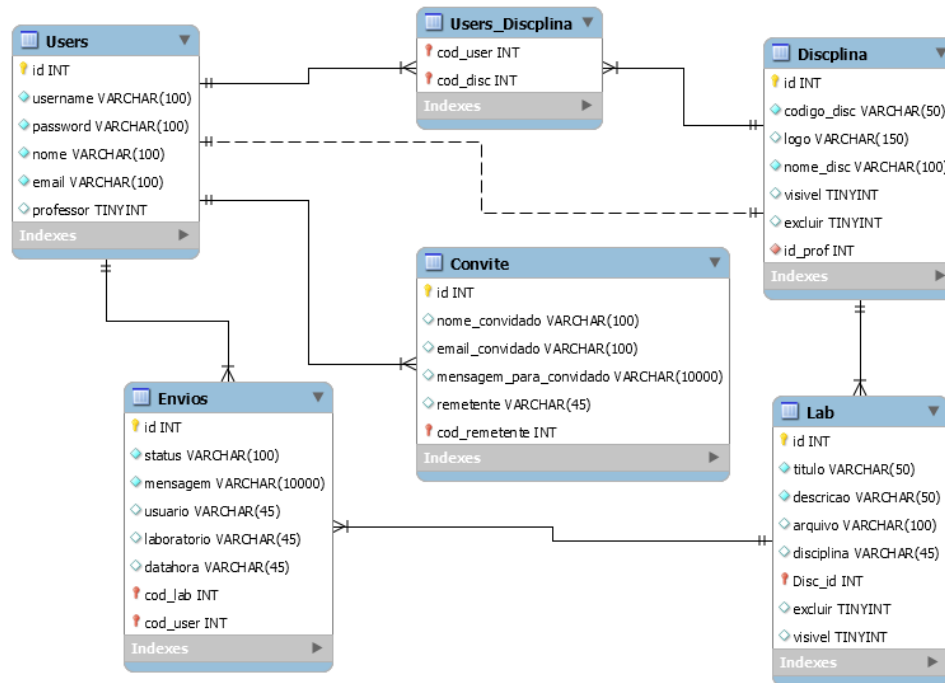


Fig. 26: Diagrama de Classes do Banco de Dados.

4.4 Criação da camada de acesso ao banco

A interação entre o banco de dados e a aplicação web ainda está em desenvolvimento, porém, é possível adicionar novos laboratórios às disciplinas já existentes, criar novas disciplinas, fazer novos cadastros, realizar buscas no banco e editar dados.

O acesso ao banco começa com o *login*, produzindo uma checagem entre o *username* e a senha. A partir dessa comparação, outro acesso é efetivado filtrando usuário como aluno ou professor. Fundamentado nessa seleção, os usuários são destinados ao menu à eles proposto, apresentado as disciplina que ou eles administram, caso professor, ou estão inscritos, caso alunos.

Um exemplo da aplicação do banco de dados no projeto é a listagem dos laboratórios existentes. Através do *id* da disciplina, uma busca ao banco é realizada, filtrando os laboratórios pertencentes à ela. Porém, para a listagem dos laboratórios disponíveis ao alunos, outra filtragem é efetuada, destinando apenas os laboratórios cuja opção de visualização para o aluno é permitida.

4.5 Código-Fonte Aberto

Para que este projeto possa ser continuado pela comunidade, abrimos o código-fonte no GitHub. O código está disponível através do link <https://github.com/IIoT-Fei/IT-FernandaSartori>.

5 Resultados Esperados

Espera-se que o trabalho proposto contribua nos seguintes tópicos:

1. Implementação de um sistema para submissão e correção automática de exercícios que envolvem programação
2. Feedback imediato ao aluno para que o mesmo saiba sobre seu progresso na disciplina
3. Identificação dos problemas no código que levaram o exercício a não ser compilado

6 Cronograma

A seguir, é apresentado o cronograma atualizado referente às atividades realizadas durante o período de seis meses. Em verde, estão representadas as atividades já realizadas, em laranja as que estão em desenvolvimento e em azul, as que necessitam ser feitas.

	2017				2018							
	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
Projeto												
Desenvolvimento das Views												
Des. Dos Controllers e APIs												
Criação do Banco de Dados												
Acesso ao Banco												
Painel do Professor												
Painel do Aluno												
Upload de Arquivo do Aluno												
Interface com Compilador												
Corretor Automático												
Integração com Moodle												

Fig. 27: Cronograma para o período

1. Desenvolvimento das *Views*:

Elaboração da parte visual do portal

2. Desenvolvimento dos *Controllers* e *API*:

O estudo de rotas da web será feito e implementado nos *Controllers* e *APIs*

3. Criação do Banco de Dados:

Instalação e configuração do banco de dados MySQL

4. Criação da camada de acesso ao banco:

Implementação das classes de persistência da informação

5. Implementação do Painel do Professor:

Desenvolvimento da área de administração de disciplinas

6. Implementação do Painel do Aluno:

Desenvolvimento da funcionalidade do aluno

7. Upload de Arquivos do Aluno:

Desenvolvimento da funcionalidade de envio de arquivos do aluno

8. Interface com Compilador:

Desenvolvimento de uma camada para comunicação com o compilador GNU/GCC.

9. Corretor Automático:

Desenvolvimento da camada de correção automática. Aqui os programas devem ser compilados automaticamente e os resultados enviados ao sistema.

10. Integração com o Moodle:

Criação de *urls* especiais para login de alunos a partir do moodle.

Referências

- [1] M. Selivon, J. L. Bez, N. A. Tonin, and R. Erechim, “Uri online judge academic: Integração e consolidação da ferramenta no processo de ensino/aprendizagem,”
- [2] I. Simonova, P. Poulova, P. Kriz, and M. Slama, *Intelligent e-Learning/Tutoring – The Flexible Learning Model in LMS Blackboard*, pp. 154–163. Cham: Springer International Publishing, 2014.

- [3] N. Cavus and T. Zabadi, “A comparison of open source learning management systems,” *Procedia - Social and Behavioral Sciences*, vol. 143, pp. 521 – 526, 2014.
- [4] J. Cole and H. Foster, *Using Moodle, 2Nd Edition*. O’Reilly, second ed., 2007.
- [5] J. Rizza, “Aula 1 - introdução - curso de flask,” feb 2018. disponível em https://www.youtube.com/watch?v=r40pC9kyoj0&list=PL3BqW_m3m6a05ALSBW02qDXmfDKIip2KX.
- [6] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*. O’Reilly Media, Inc., 1st ed., 2014.
- [7] S. Aggarwal, *Flask Framework Cookbook*. Packt Publishing, 2014.
- [8] R. Copeland, *Essential SQLAlchemy*. O’Reilly, first ed., 2008.