

RELATÓRIO FÍNIAL

Proposta de Iniciação Tecnológica

Aluno: Fernanda Sartori Beltrame

R.A.: 22116031-0

Sistema de Correção Automática de Código-Fonte

Período: Dezembro/2016 a Novembro/2017 (12 Meses)

Instituição: Centro Universitário da FEI

Orientador: Guilherme Wachs Lopes

São Bernardo do Campo, SP

Novembro de 2016

FENOTÍMO

Resumo

Quais sistemas?

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. Na área da Computação, muitos desses sistemas são usados até mesmo para correção automática de códigos-fonte. Contudo, um dos principais impecílios 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 ferramente 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 ensiar lógica; estruturas de dados, que envolve lógica com estruturas fundamentais para elaboração de programas; compiladores, que utiliza programação para 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 esquecer 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 [Selivon et al.,]. 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 cerca de 39.000 professores em mais de 1.350 instituições de ensino que juntas entrega 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, áudios, de tal forma que torne o entendimento da disciplina mais tangível para o aluno.

Os alunos também podem usar a ferramente 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.[Simonova et al., 2014]

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* enfatiza a criação de interação colaborativa e ambientes de ensino online focada no aluno [Cavus and Zabadi, 2014, Cole and Foster, 2007].

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

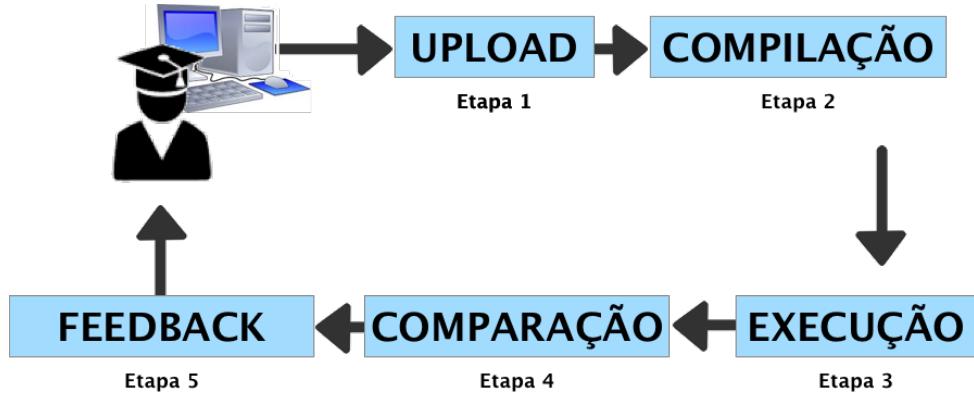


Fig. 1: Etapas do processo de correção automática

O processo de correção é feito em 5 etapas: Upload, Compilação, Execução, Comparaçao 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 linguagens 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 do aluno. O resultado dessa etapa será um arquivo de saída que foi gerado pelo programa executável.

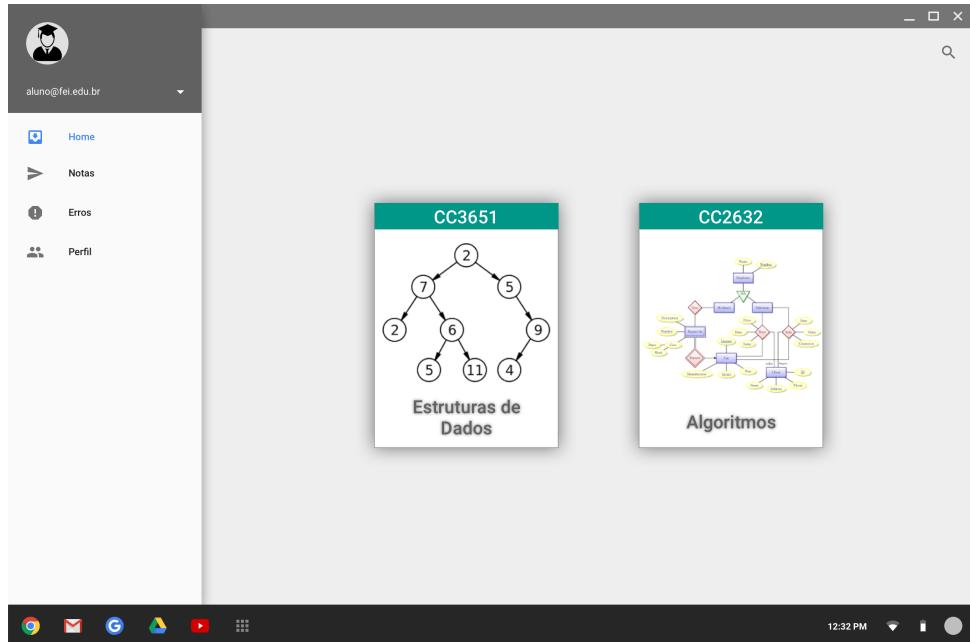


Fig. 2: *Mock-up* da tela de disciplinas

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).

A screenshot of a web application interface. On the left is a sidebar with a user icon, the email 'aluno@fei.edu.br', and four menu items: 'Home', 'Notas', 'Erros', and 'Perfil'. The main area has a title 'Exercícios - CC2632'. Below it is a table:

Código	Nome	Status
001	Recursão	Erro de Compilação
002	Sequência de Fibonacci	Erro de Saída
003	Maior e Menor Valor no Vetor	Correto
004	Dikjstra	Correto
005	BFS	Erro de Saída
006	Kruskal	Erro de Compilação
007	Ponteiros	Erro de Compilação

The status column uses color coding: red for compilation errors, orange for output errors, and green for correct submissions.

Fig. 3: Mock-up da tela de exercícios

A screenshot of a web application interface. On the left is a sidebar with a user icon, the email 'aluno@fei.edu.br', and four menu items: 'Home', 'Notas', 'Erros', and 'Perfil'. The main area has a title 'Exercício - Recursão'. Below it is a text box containing the following text:

Desenvolva um algoritmo recursivo em C que receba um vetor de inteiros como entrada e a quantidade total de elementos nesse vetor e retorne o elemento de maior valor.

Below this is a 'Exemplo de Chamada' section with the following code:

```
int v[] = {5,2,6,3,7,8,1,2};  
int m = maior(v, 8);  
  
// m = 8
```

To the right is a table:

Envio	Status
001	Erro de Compilação
002	Erro de Saída
003	Correto
004	Correto
005	Erro de Saída
006	Erro de Compilação
007	Erro de Compilação

Below the table is a search bar labeled 'Enviar:' with a magnifying glass icon and an upload icon.

Fig. 4: Mock-up da tela de envios

4 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

5 Cronograma *~~5 Celular Como ultima Seguro e atualizar.~~*



Fig. 5: 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.

~~6 Realizações~~ Resultados Parciais.

~~Para a confecção do projeto, pressupõem-se a necessidade de um nome e um logotipo que represente a proposta. Sendo assim, o nome escolhido foi "Escape++" e seu logo apresenta-se a seguir.~~

~~†1 : Com o objetivo de se ter uma identidade visual para o sistema, foi proposto o nome "Escape++" e o logo da Fig. 6.~~



6.1) DESCRIÇÃO DAS TELAS

Fig. 6: Logotipo do Escape++.

1. Desenvolvimento das Views:

~~R2~~ ~~Alunos~~ A seguir serão exibidas as views realizadas. 6.1.1) ALUNOS

(a) Tela de Login:

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

~~R2:~~ O desenvolvimento das telas ^{de aluno} foi feito todo

Como base a sequência de passos da Fig. 1, o objetivo aqui foi fazer uma interface simples e intuitiva seguindo os mesmos moldes do modelo.

(a) Tela DE LOGIN: ...

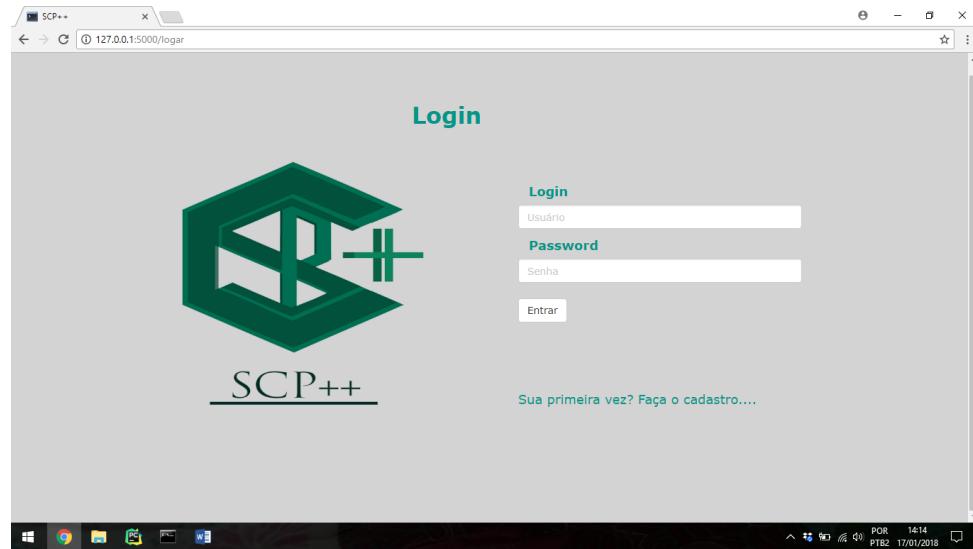


Fig. 7: Tela de Login.

(b) Tela de Cadastro:

Como os alunos serão encaminhados pelo Moodle, a página de cadastro será destinada ao cadastro de novos professores.

Explique melhor isso.

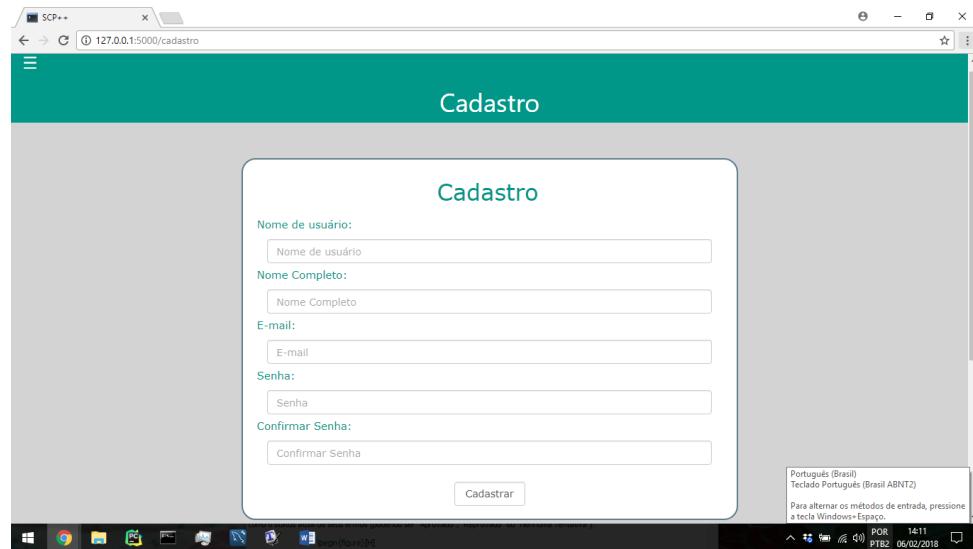


Fig. 8: Tela de Cadastro para novos professores.

(c) Portal do Aluno (Home):

Aqui estarão disponíveis as disciplinas que o aluno está inscrito.

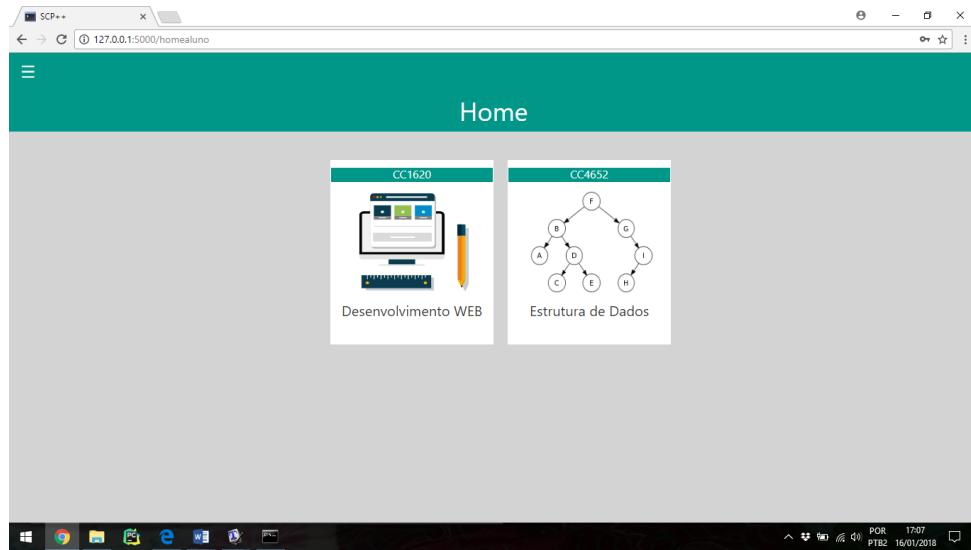


Fig. 9: Tela de Menu Principal do Portal do Aluno.

(d) 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”).

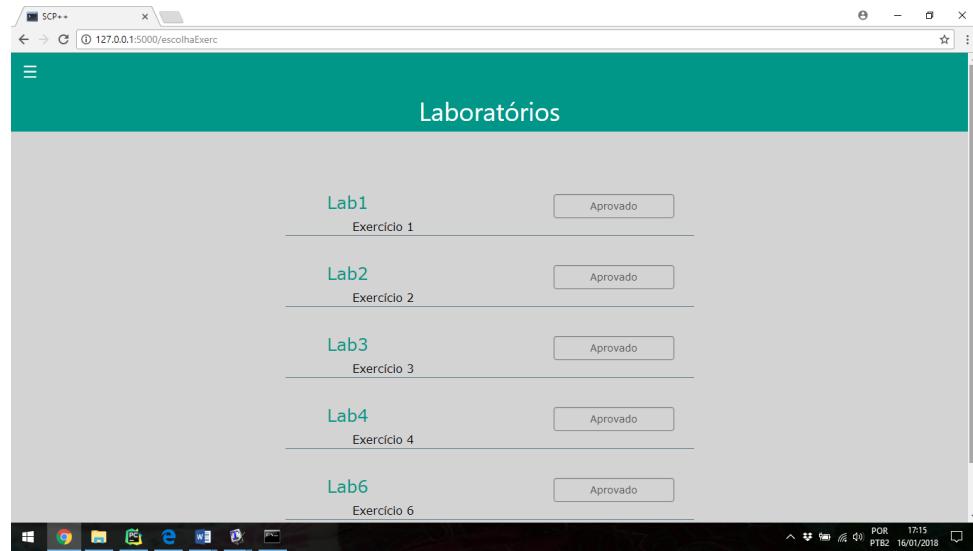


Fig. 10: Menu da disciplina.

(e) 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.

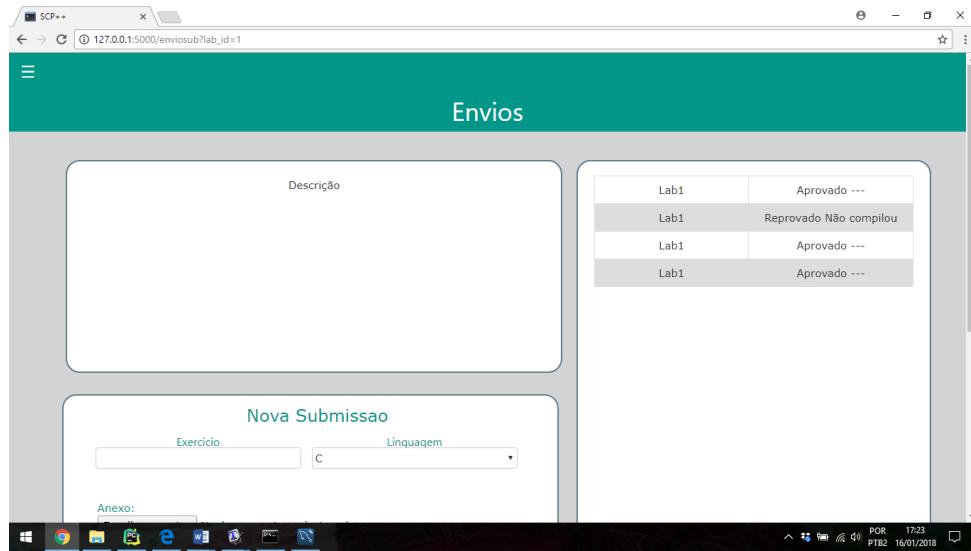


Fig. 11: Tela do exercício destinada ao envio de submissões.

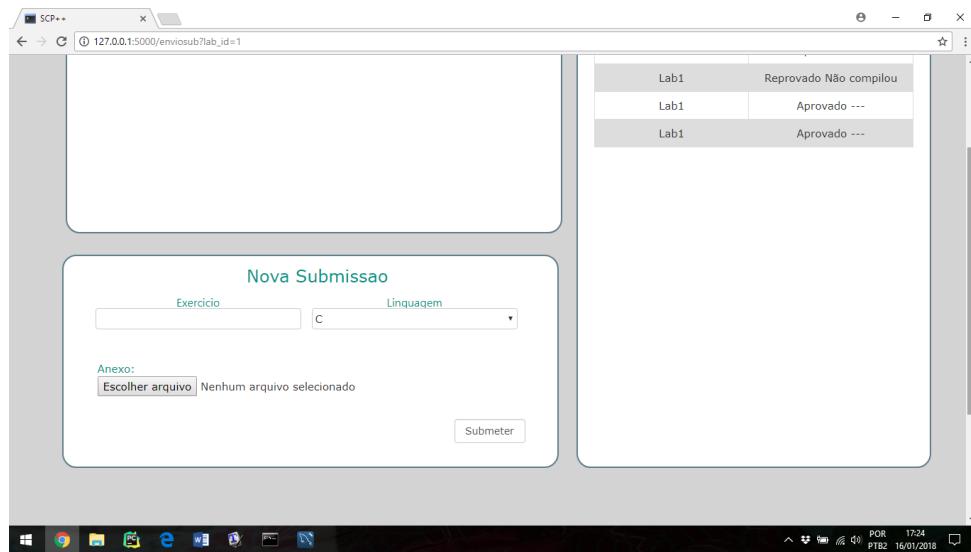


Fig. 12: Tela do exercício destinada ao envio de submissões.

6.1.2) Professor

(f) 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 disci-

plina, excluir disciplina e torná-la disponível somente para usuários professores.

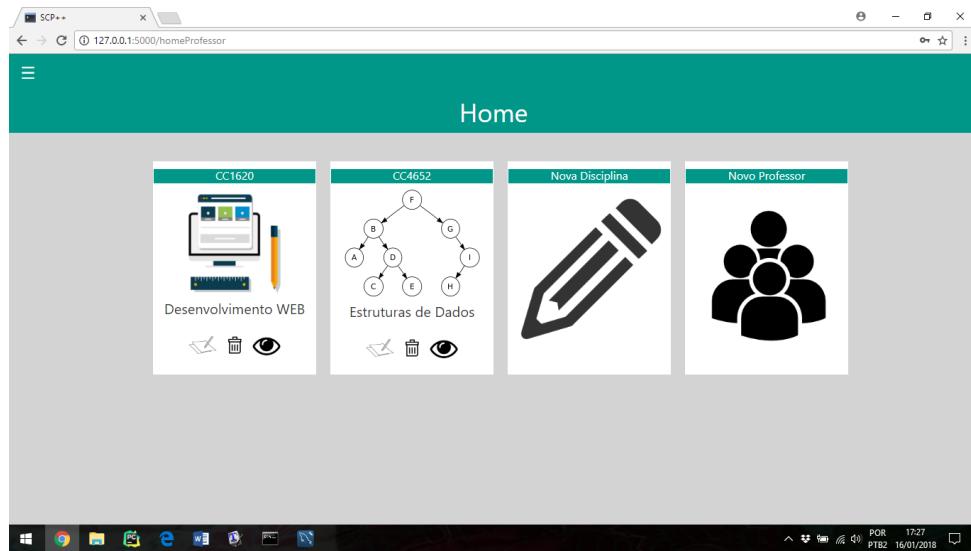


Fig. 13: Menu inicial do Portal do Professor.

(g) 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, portantno, editar o título referente à disciplina, sua chave de acesso e seu logo. Segue abaixo um exemplo da busca e da página de edição.

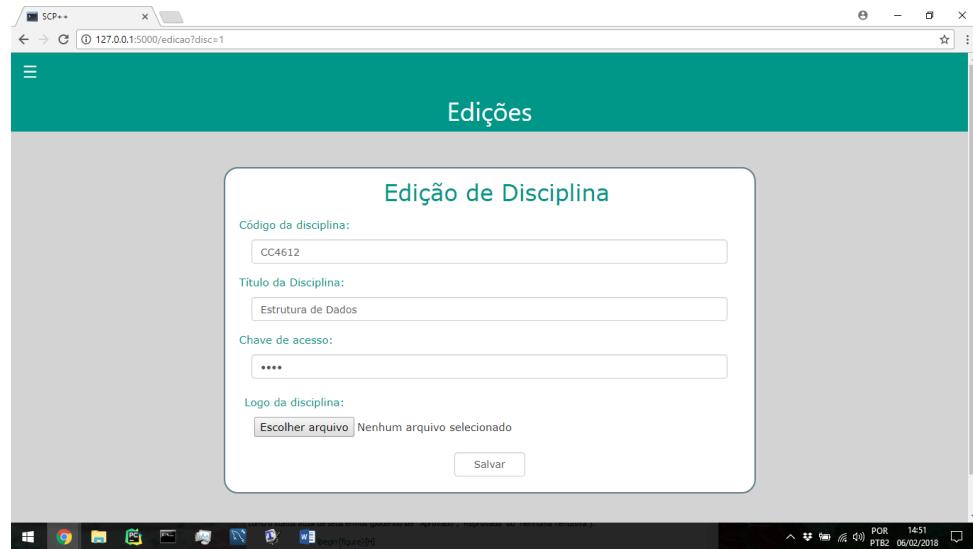


Fig. 14: Tela para edição de disciplinas já existentes.

(h) 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.

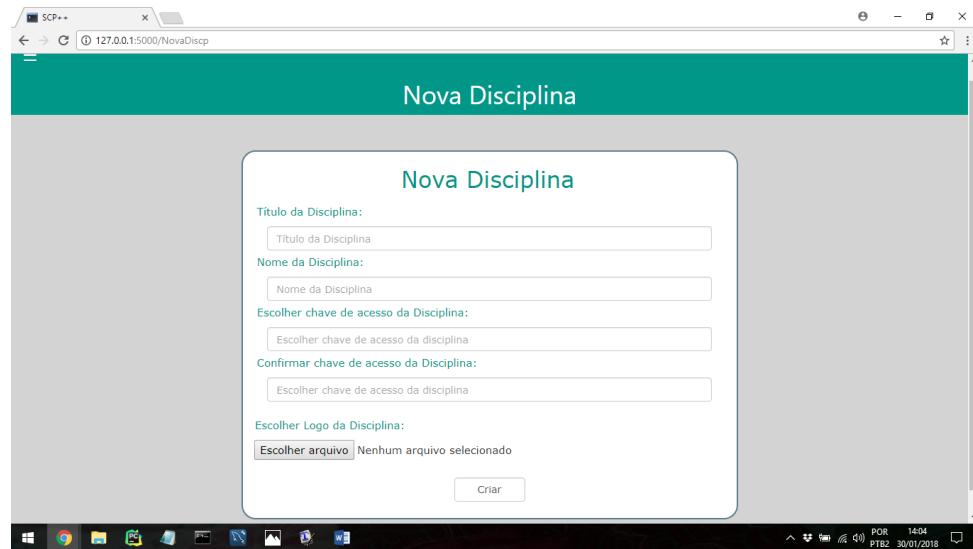


Fig. 15: Tela para criação de nova disciplina.

(i) Novo professor:

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

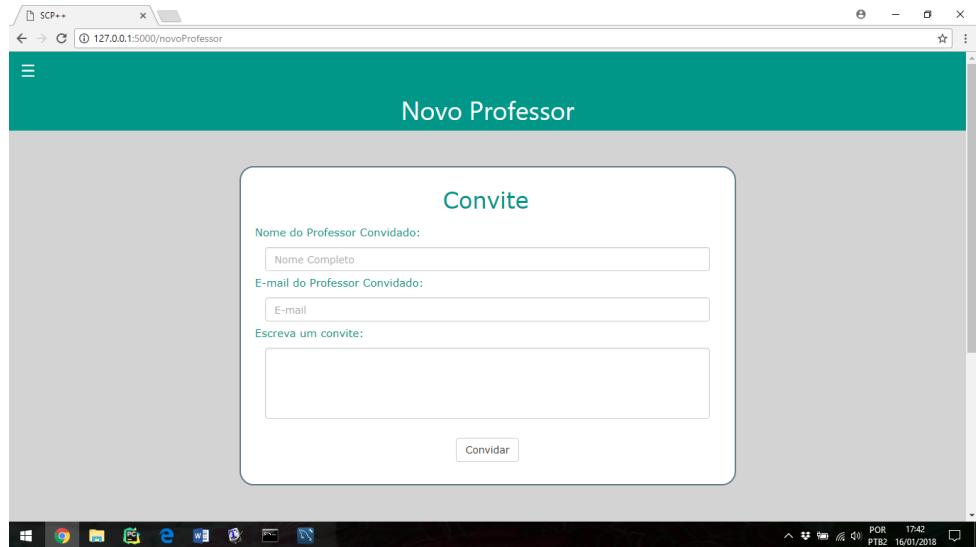


Fig. 16: Tela destinada para convidar um novo professor.

(j) Menu da Disciplina (professor):

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.

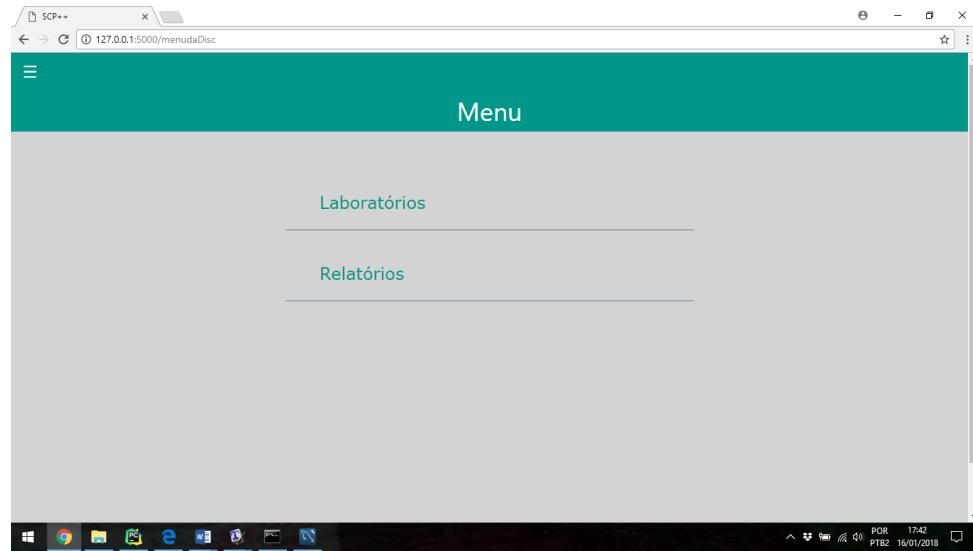


Fig. 17: Menu da disciplina para o Professor.

(k) Laboratórios da Disciplina (professor):

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

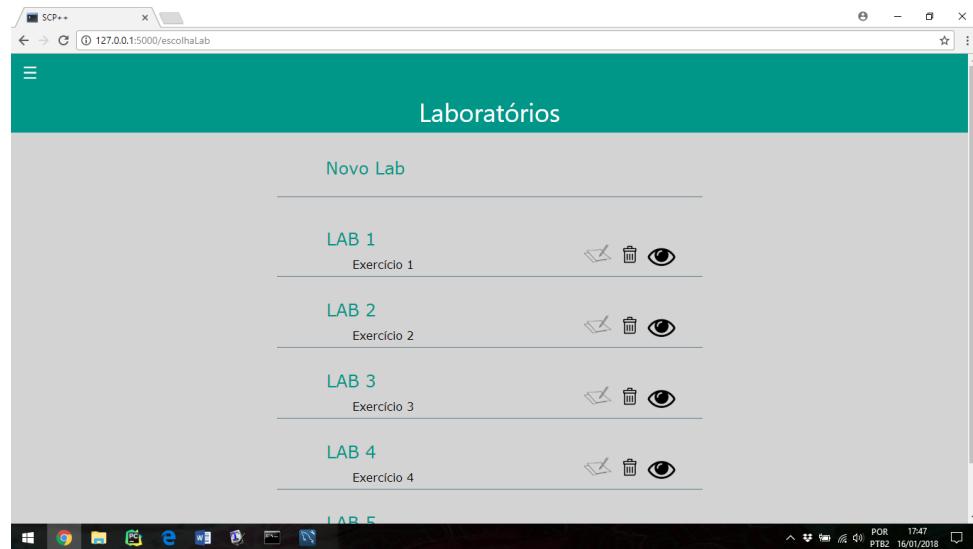


Fig. 18: Menu dos laboratórios.

(l) 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.

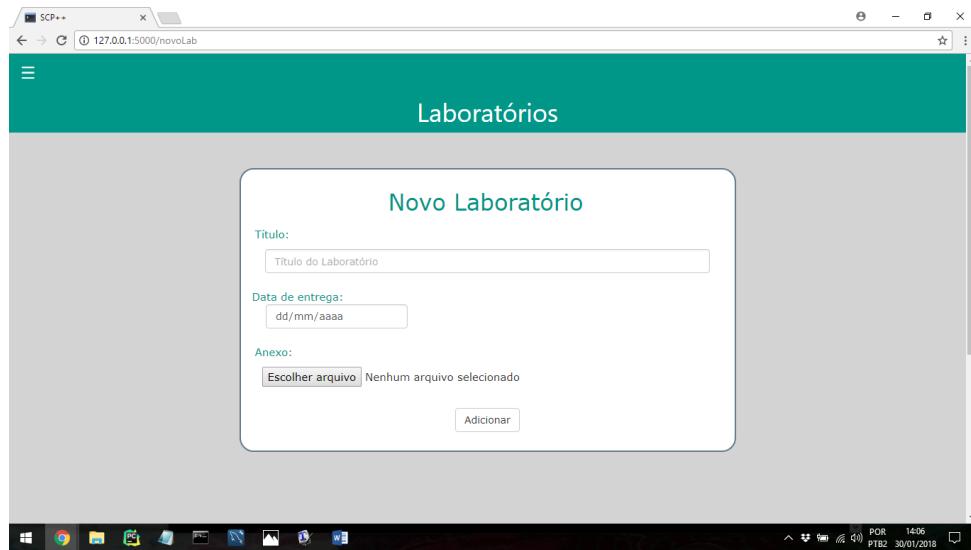


Fig. 19: Tela para criação de novo laboratório.

(m) 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	0				
Fernanda	Nenhuma Tentativa	Nenhuma Tentativa	Aprovado	Reprovado	Aprovado	2
Guilherme Lopes	Nenhuma Tentativa	Aprovado	Aprovado	Nenhuma Tentativa	Aprovado	3

Fig. 20: Tela para acompanhamento do desenvolvimento dos alunos.

(n) 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.

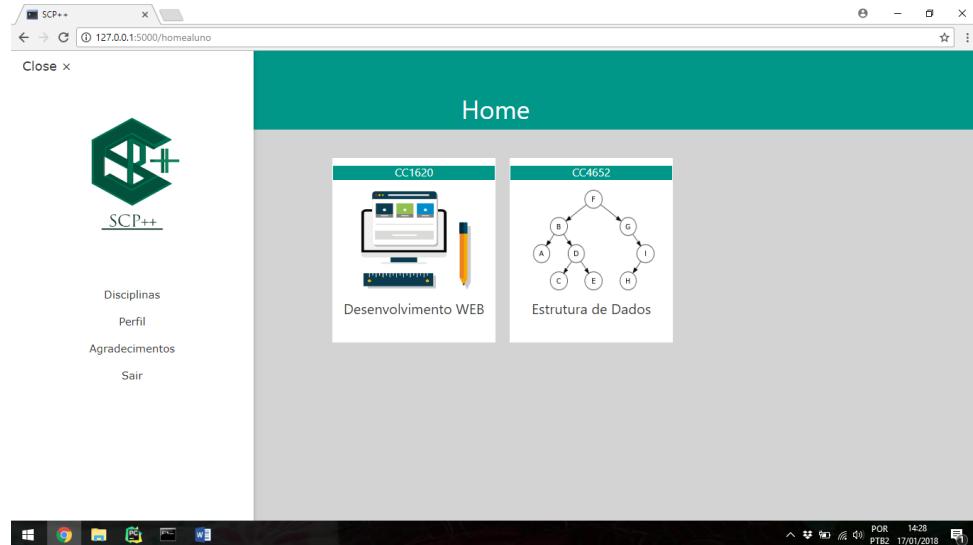


Fig. 21: Menu lateral.

(o) Perfil:

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

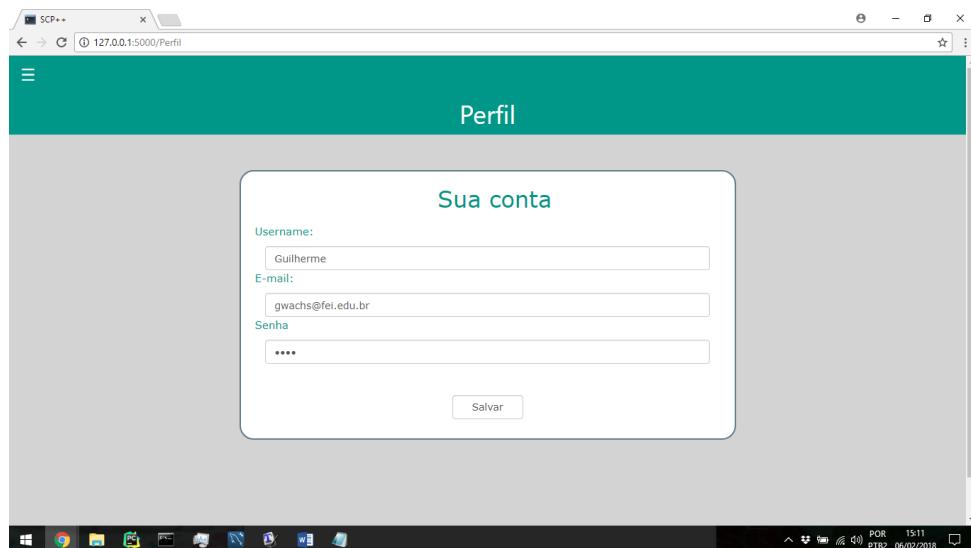


Fig. 22: Tela de acesso ao perfil.

(p) Agradecimentos:

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



Fig. 23: Agradecimentos.

6.2) Desenvolvimento dos Controllers e API:

Na construção desse projeto, foi utilizado o microframework, para Python, chamado Flask. Flask é “baseado em Werkzeug, Jinja 2 e ‘boas intenções’” [1].

O Flask foi utilizado para fazer um melhor gerenciamento das rotas, criação do servidor e conexão com o banco de dados.

Cologne new
item or ROTAS
DO SISTEMA

Fig. 24: Exemplo da utilização do Flask para gerenciamento das rotas.

IC [C:\Users\Fernanda\Dropbox\IC-Fernanda\Flask\IC] - app\controllers\default.py [IC] - PyCharm

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
```

Project: IC

app controllers default.py

```
IC C:\Users\Fernanda\Dropbox\IC-Fernanda\Flask\IC
```

```
172     def inicio():
173         return render_template('index.html')
174
175     @app.route('/escolhaExerc', methods=['GET', 'POST'])
176     def exercicio():
177         l = Lab.query.all()
178         r = Envios.query.filter_by(cod_lab=1).first()
179         print(r)
180         return render_template('escolhaExerc.html', lab=l, r=r)
181
182     @app.route('/menuDisc', methods=['GET', 'POST'])
183     def menuDisc():
184         return render_template('menuDisciplina.html')
185
186     @app.route('/escolhaLab', methods=['GET', 'POST'])
187     def escolhaLab():
188         return render_template('escolhaLab.html')
189
190     @app.route('/novLab', methods=['GET', 'POST'])
191     def novoLab():
192         return render_template('novoLab.html')
193
194     @app.route('/relatorios', methods=['GET', 'POST'])
195     def relatorios():
196         return render_template('relatorio.html')
197
198     @app.route('/novoProfessor', methods=['GET', 'POST'])
199     def novoProfessor():
200         return render_template('novoProfessor.html')
201
202     @app.route('/exercicio.html')
203     def exercicio():
204         return render_template('exercicio.html')
205
206     @app.route('/HomeProf.html')
207     def HomeProf():
208         return render_template('HomeProf.html')
209
210     @app.route('/index.html')
211     def index():
212         return render_template('index.html')
213
214     @app.route('/inscricao.html')
215     def inscricao():
216         return render_template('inscricao.html')
```

PyCharm status bar: 144/67 CRLF: UTF-8 15:28 PTB2 22/01/2018

Fig. 25: Exemplo da utilização do Flask para gerenciamento das rotas.

#3) Neste projeto foi utilizado o Framework para PYTHON chamado FLASK. Esse framework tem por objetivo ser simples e fácil de fazer manutenção. O Flask pônei interação com o MySQL para criação de views e pode ser facilmente conectado a diferentes bancos

6 Realizações DE DADOS.

24

(6.3) 6. Criação do Banco de Dados:

Embora apresente "micro" em seu nome, não significa que o microframework Flask deva apresentar toda a aplicação em somente um arquivo Python, mas sim que com o Flask procura tornar o núcleo extensível, e que o desenvolvedor Python tem liberdade para tomar decisões, como por exemplo, o banco de dados [2]. Nesse projeto utilizamos o MySql com auxílio do SQLAlchemy.

Para a construção do projeto, está sendo utilizada a IDE para Python chamada PyCharm onde é possível implementar, através do Flask, o servidor e banco de dados.

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.

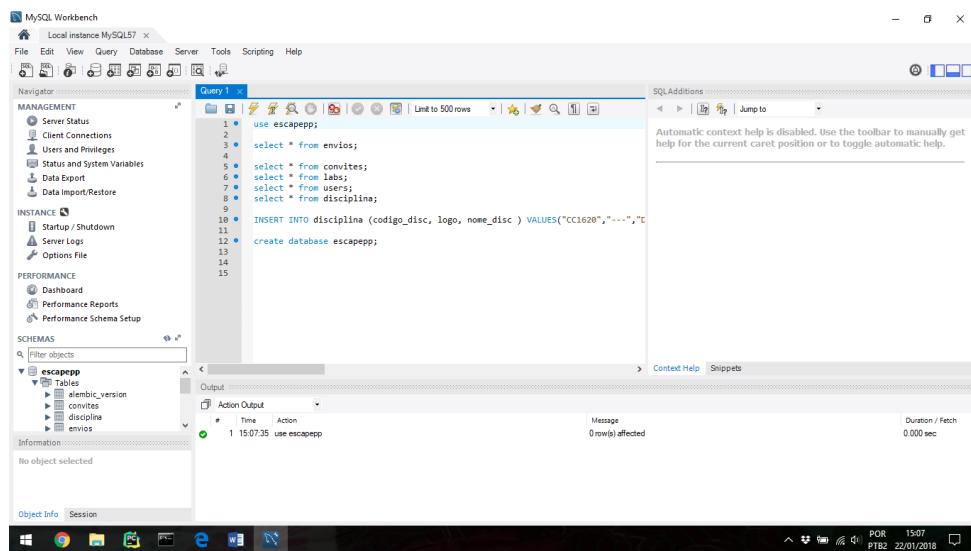


Fig. 26: Exemplo de utilização do SQL Workbench para gerenciar o banco de dados.

(#3 - Continuação)

Outra característica do flask é que ele possibilita a criação de rotas (URLs) customizadas de forma simplificada. O flask também pôde ser propried servir web, embora possa ser integrado como WSGI ou FASTCGI.

*4: 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 esse sistema deve-se por 3 fatores. O primeiro é que o MySQL tem uma

6 Realizações

25

```
IC [C:\Users\Fernanda\Dropbox\IC-Fernanda\Flask\IC] - ...app\models\tables.py [IC] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
File app models tables.py default.py EscolaExercicio.html NovoProfessor.html Login.html base.html Relatorios.html Listagem.html
Project Z-Structure
IC C:\Users\Fernanda\Dropbox\IC-Fernanda\Flask\IC
app
  +-- views
    +-- library_root
  +-- controllers
    +-- default.py
  +-- models
    +-- forms.py
    +-- tables.py
  +-- static
  +-- templates
    +-- idea
      +-- Agregados.html
      +-- base.html
      +-- basehtml.html
      +-- calculo.html
      +-- Cadastro.html
      +-- CadastroAluno.html
      +-- CadastroProfessor.htm
      +-- Edico.html
      +-- EditarLab.html
      +-- EnvioArquivo.html
      +-- EnvioSubmissao.html
      +-- FazendaExercicio.html
      +-- FazLab.html
      +-- Fazun.html
      +-- exercicio.html
      +-- HomeProf.html
      +-- index.html
      +-- InscreverDisc.html
    Lab
tables.py
default.py
EscolaExercicio.html
NovoProfessor.html
Login.html
base.html
Relatorios.html
Listagem.html

class User(db.Model):
    __tablename__ = "users"
    # campos da tabela coluna do banco de dados

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(100), unique=True)
    password = db.Column(db.String(100), nullable=False)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(100), unique=True)
    professor = db.Column(db.Boolean)

    @property # propriedades para saber se o usuário está logado
    def is_authenticated(self):
        return True

    @property # propriedades para saber se o usuário está logado
    def is_active(self):
        return True

    @property # propriedades para saber se o usuário está logado
    def is_anonymous(self):
        return False # retorna falso se o usuário esta logado

    def get_id(self): # função que retorna um unicode que identifica um usuário
        return str(self.id) # no python3 a strig é unicode, no python2 é o code(self.id)

    # construtor que inicializa os users
    def __init__(self, username, password, name, email, professor):
        self.username = username
        self.password = password
        self.name = name
        self.email = email

Event Log
53:42 CRLF: 1530
PTB2 22/01/2018
```

Fig. 27: Exemplo de criação de tabela do banco de dados realizado na IDE.

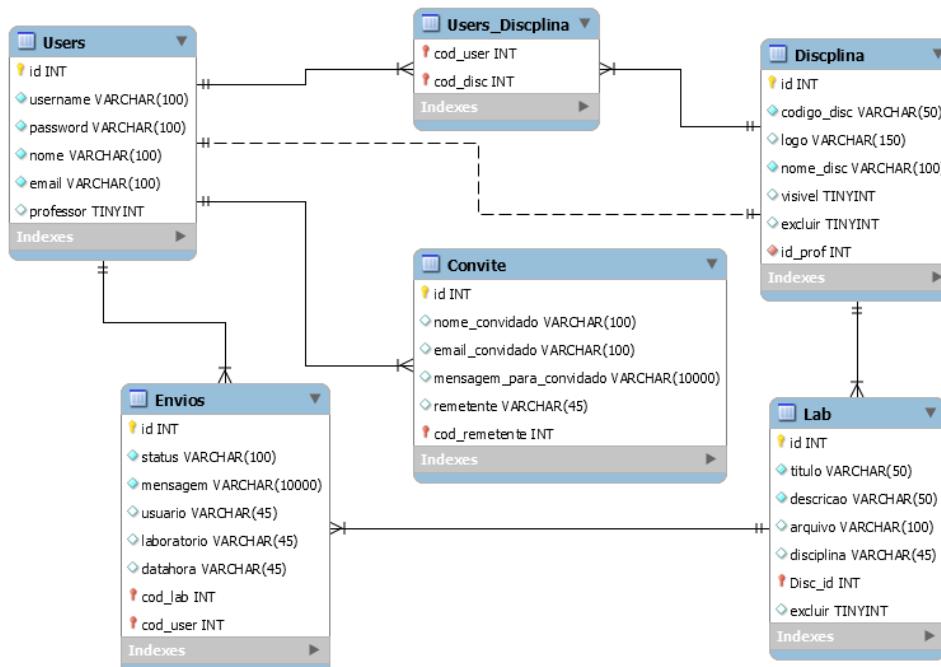


Fig. 28: Diagrama de Classes do Banco de Dados.

4 Criacão da camada de acesso ao banco:

4. Criação da camada de acesso ao banco.

Comunidade ativa e com muitos soluções prontas online. A segunda é que faz parte do Software line. A última é que tem fácil integração com o Firenz através do SQL Alchemy.

A interação entre o banco de dados e a aplicação web ainda está em desenvolvimento, porém é possível, de forma manual, adicionar, deletar e excluir dados do banco de dados através da IDE.

```
# EXEMPLO DO CREATE. ADICIONANDO UM REGISTRO AO BANCO DE DADOS
@app.route("/write/info")
@app.route("/write", defaults={"info": None})
def write(info):
    i = User("el", "1234", "Mel S B", "seifel.edu.br", False) # passando os parametros que quero inserir
    db.session.add(i) # é o periodo que meu usuário esta logado
    db.session.commit() # o salvamento de informações, a sessão e temporaria, logo para não recarregar eu as salvo com o commit
    return "OK, foi criado com sucesso!"

# EXEMPLO DO READ. LENDO OS REGISTROS EXISTENTES
@app.route("/read/info")
@app.route("/read", defaults={"info": None})
def read(info):
    if info:
        r = User.query.filter_by(username="fernandinha").first() # se for selecionando apenas um dos registros filtrados por um campo
        print(r)
    else:
        r = User.query.order_by(User.username).all() # se for usar .all() seleciona todos os registros filtrados por um campo
    print(r)
    _print(r.username, r.name, r.email, r.cidade, r.id) # funciona se vc imprimir usando first(), se for com all() ele da erro
    return "OK, foi lido com sucesso!"

# EXEMPLO DO UPDATE. ALTERANDO ALGUM REGISTRO EXISTENTE
@app.route("/update/info")
@app.route("/update", defaults={"info": None})
def update(info):
    if info:
        r = User.query.filter_by(username="Fernanda").first() # estou pegando o primeiro registro que tem o username F
        r.name = "Fernanda" # devo o registro que já usei lá em cima, o id ou o name dele
        db.session.add(r)
        write()
```

Fig. 29: Exemplo de CRUD no banco de dados através da IDE.

Fig. 30: Exemplo de CRUD no banco de dados através da IDE.

Um exemplo da aplicacão do banco de dados no projeto é a listagem dos laboratórios

existentes. A seguir estão representadas figuras para exemplificar a busca no banco de dados e a exibição na tela.

```

IC [C:\Users\Fernanda\Dropbox\IC-Fernanda\Flask\IC - ...app\templates\EscolhaExercicio.html [IC] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
IC app templates EscolhaExercicio.html default.py MenudaDisciplina.html EnvioSubmissao.html forms.py
Project Z-Structure 2 Favorites
base.html basehtml.html basehtlm.html cabecalho.html Cadastro.html CadastroAluno.html CadastroProfessor.htm Edicao.html EditarLab.html EnvioArquivo.html EnvioSubmissao.html EnvioSubmissao.html EscolhaExercicio.html EscolhaLab.html Exrun.html exercio.html HomeProf.html index.html InscricaoDsc.html Listagem.html Login.html loginsse.html MenudsDisciplina.htm NovoAluno.html NovoLab.html NovoProfessor.html Perfil.html Relatorios.html View.html
init_.py TODO Python Console Terminal
77:16 CRLF: UTF-8s 18:12 PTB2 22/01/2018
Event Log

```

```

<form action="" method="post">
    <div>
        <div class="espaco">
            <div class="vertical-menu" style="...>
                <% for i in lab %>
                    <a href="/enviodesc/lab_id={{ i.id }})" style="...>
                        <div face="verdana" color="#009688" size="5" style="...>
                            {{ i.titulo }}
                        </div>
                    </a>
                    <div style="...> class="Formulariologin">
                        {{ r.status }}
                    </div>
                    <div face="verdana" color="black" size="3" style="...>
                        Exercicio {{ i.id }}
                    </div>
                    <br style="..."/>
                <% endfor %>
            </div>
        </div>
    </div>
</form>

```

Fig. 31: Utilização do Flask para exibir, buscando do banco de dados, os laboratórios disponíveis.

```

IC [C:\Users\Fernanda\Dropbox\IC-Fernanda\Flask\IC - ...app\templates\EnvioSubmissao.html [IC] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
IC app templates EnvioSubmissao.html default.py EscolhaExercicio.html MenudaDisciplina.html EnvioSubmissao.html forms.py
Project Z-Structure 2 Favorites
base.html basehtml.html basehtlm.html cabecalho.html Cadastro.html CadastroAluno.html CadastroProfessor.htm Edicao.html EditarLab.html EnvioArquivo.html EnvioSubmissao.html EnvioSubmissao.html EscolhaExercicio.html EscolhaLab.html Exrun.html exercio.html HomeProf.html index.html InscricaoDsc.html Listagem.html Login.html loginsse.html MenudsDisciplina.htm NovoAluno.html NovoLab.html NovoProfessor.html Perfil.html Relatorios.html View.html
init_.py TODO Python Console Terminal
49:37 CRLF: UTF-8s 18:17 PTB2 22/01/2018
Event Log

```

```

<div style="...> style="...>
    <div style="...>
        <div style="...>
            <table>
                <% for i in env %>
                    <tr>
                        <td> {{ i.lab.titulo }} </td>
                        <td> {{ i.lab.descricao }} </td>
                        <td> {{ i.lab.sugem }} </td>
                    </tr>
                <% endfor %>
            </table>
        </div>
    </div>
</div>
<div id="enviando">
    <div style="...>
        <p class="textotopico" style="...><font face="verdana" color="#009688" size="5"> Nova Submissao</font></p>
        <center> <div> <div> <div> <div> <table> <tr> <td>

```

Fig. 32: Código referente à tela de submissões. Faz a busca no banco de dados relacionado ao valor do id e retorna os dados encontrados.

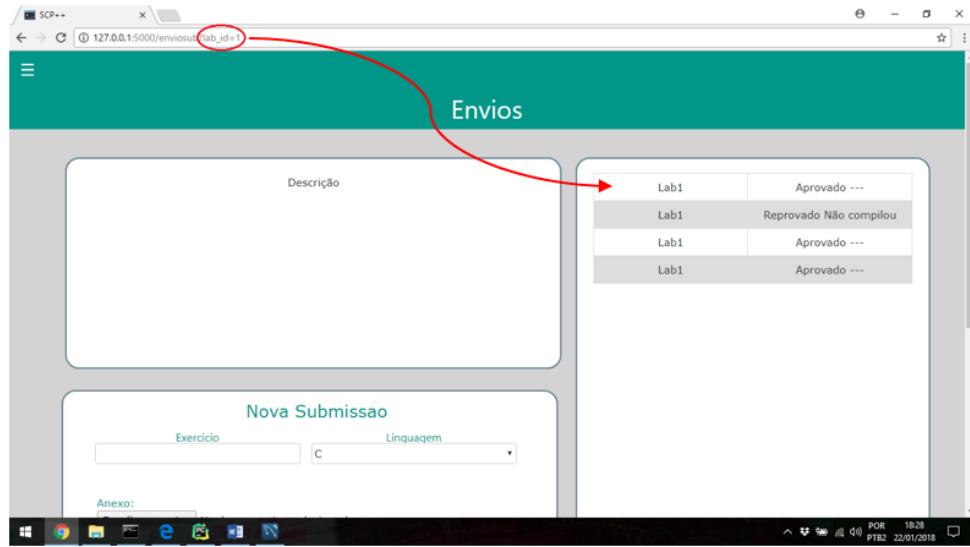


Fig. 33: Conforme o id do laboratório, a busca no banco retorna o status e a mensagem da submissão.

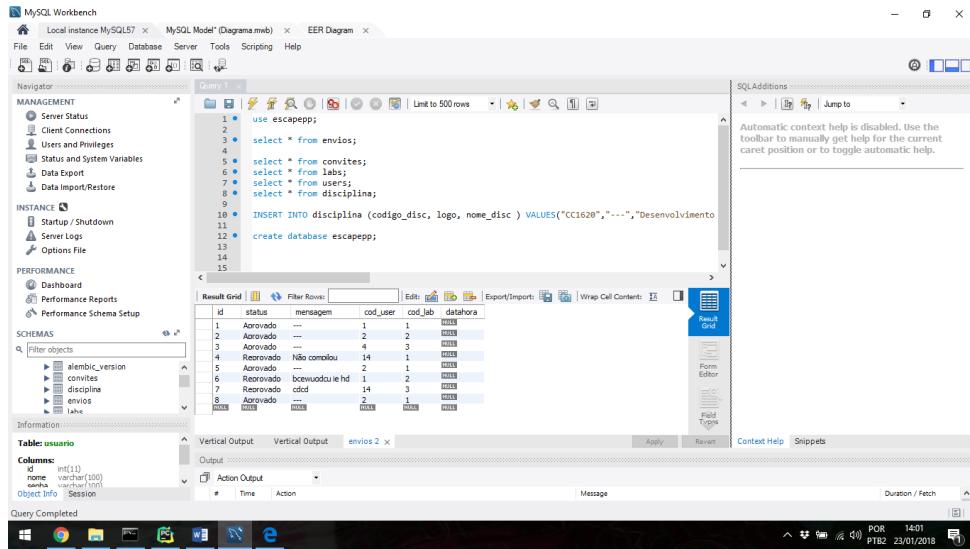


Fig. 34: Exemplo de como os dados se apresentam no banco para que seja representável a Figura 32.

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

- [Cavus and Zabadi, 2014] Cavus, N. and Zabadi, T. (2014). A comparison of open source learning management systems. *Procedia - Social and Behavioral Sciences*, 143:521 – 526.
- [Cole and Foster, 2007] Cole, J. and Foster, H. (2007). *Using Moodle, 2Nd Edition*. O'Reilly, second edition.
- [Selivon et al.,] Selivon, M., Bez, J. L., Tonin, N. A., and Erechim, R. Uri online judge academic: Integração e consolidação da ferramenta no processo de ensino/aprendizagem.
- [Simonova et al., 2014] Simonova, I., Poulova, P., Kriz, P., and Slama, M. (2014). *Intelligent e-Learning/Tutoring – The Flexible Learning Model in LMS Blackboard*, pages 154–163. Springer International Publishing, Cham.