Projeto Quem me ajuda?

Nome: João Alcimar Santiago Júnior

Matrícula: 114211258

Disciplina: DACA

# Introdução

O Projeto consiste num sistema on-line de tutores para alunos que precisam de reforço estudantil. Este sistema foi desenvolvido tecnologia Spring Boot em java para o Back-End e Angular em javascript e html para o Front-end. Foi feita arquitetura rest e usados conceitos MVC e Façade.

## Descrição do Projeto

Quem me ajuda objetiva auxiliar estudantes de quaisquer grau de formação por meio de um sistema de tutores. Os usuários - alunos e tutores - são capazes de se registrarem na plataforma e estabelecer uma conexão entre eles de acordo com a oferta e demanda por ajuda, presencial ou on-line, nas mais diversas disciplinas. O aluno que teve proficiência alta é capaz de se tornar tutor. Os tutores disponibilizam seus horários livres para que haja uma melhor compatibilidade com as necessidades dos ajudados. Os alunos que ficam satisfeitos com o trabalho prestado são capazes de doar valores pelo aplicativo, quantia que é dividida entre o sistema e os tutores, além de estarem aptos a votar negativamente ou positivamente no serviço prestado. Não existe limites para prestação e pedidos de ajuda, e quanto mais bem classificado é o tutor por seus alunos, melhor sua percentagem de ganho nas doações.

## 

## 

## 

## User Stories

Foram implementadas 6 USs, as quais:

## *US1 - Cadastrar Alunos*

*Como sistema, deve ser possível cadastrar alunos para que os mesmos fiquem registrados e possam ser recuperados pela matrícula.*

*Cada aluno é identificado unicamente pela matrícula e tem um nome, código do curso, telefone e e-mail. Email é obrigatório e deve ser validado (ter uma arroba e ter algo antes e depois da arroba). Não deve ser possível nomes vazios ou em branco. O telefone é opcional. Todo aluno possui uma nota de avaliação (entre 0 e 5). Essa nota inicia como 5 e muda de acordo com as avaliações retornadas pelos tutores.*

*US2 - Definir papéis (Tutores)*

*Como aluno, quero poder me disponibilizar como tutor para que eu possa atender ajudas de outros alunos de acordo com minha proficiência.*

*Cada aluno no sistema pode se disponibilizar como tutor. Ao se cadastrar como tutor, é preciso indicar a disciplina na qual a pessoa pode ajudar mais (exemplo: “Cálculo 2”, “ICC”, “Programação 2”). O aluno deve indicar também uma proficiência (valor de 1-5), ou seja, o quão hábil ele se sente na disciplina. Todo tutor possui também uma nota de avaliação (entre 0 e 5). Essa nota inicía como 4 e muda de acordo com as avaliações retornadas pelos alunos. Cada tutor tem por fim, uma quantidade de dinheiro que foi recebida pelo sistema (começa em zero).*

*US3 - Disponibilizar e listar horários e locais*

*Como tutor, quero disponibilizar meus locais e horários para que eu possa atender alunos de acordo com minha disponibilidade.*

*Cada tutor deve cadastrar um conjunto de locais (identificados por nome) e um conjunto de dias disponíveis. O tutor deve indicar sua disponibilidade nos dias úteis da semana (seg, ter, qua, qui, sex). Importante: o local não é associado ao dia.*

*US4 - Cadastrar Pedidos de Ajuda (Presencial ou Online)*

*Como aluno, quero pedir ajuda presencial ou online de forma que um tutor possa me atender de acordo com a minha necessidade.*

*Um aluno (seja tutor ou não) pode cadastrar um pedido de ajuda. Existem dois tipos de pedidos, o pedido de ajuda presencial e o pedido de ajuda online.*

*Na ajuda presencial, o aluno indica a disciplina e dia que quer uma ajuda e o local que tem interesse. Ao realizar um pedido no sistema, o sistema deve associar um tutor a esse pedido. O tutor escolhido precisa ter proficiência na disciplina e disponibilidade no local e no horário/dia indicado. Caso mais de um tutor esteja disponível naquele dia o de maior pontuação deve ser retornado (ou o primeiro aluno cadastrado em caso de empate). O mesmo tutor pode ser retornado para vários pedidos de ajuda diferentes.*

*Na ajuda online, a interação irá acontecer via email (ou hangouts) sem necessidade de disponibilidade de horário. Há necessidade de especificar apenas a disciplina de interesse da ajuda. O tutor associado deve ter proficiência nessa disciplina e ser aquele de maior pontuação (ou o primeiro cadastrado em caso de empate).*

*Ao cadastrar um pedido, é retornado um ID referente a ajuda marcada. Deve ser possível pegar informações sobre esse pedido de ajuda (qual foi a disciplina, dia e local de interesse). Deve ser possível pegar a matrícula do tutor que atenderá aquela ajuda, bem como pegar informações sobre a ajuda em si (horário, dia, local e disciplina).*

*US5 - Avaliar Tutor*

*Como aluno, desejo avaliar o tutor de forma que ele possa ser priorizado nos atendimentos e possa receber mais verba de doações ao sistema.*

*US6 - Doar ao Tutor*

*Como tutor, quero poder receber dinheiro de doações de acordo com meu nível.*

*É possível doar dinheiro a um tutor. Independente da ajuda que foi dada, é possível especificar um tutor e doar dinheiro para este. O tutor recebe de acordo com suas avaliações. Para cada doação, uma parte (fatia tutor) é dada ao tutor e o resto é destinado ao sistema. O montante destinado ao tutor é determinado no momento de doação de acordo com o nível do tutor naquele mesmo momento. Caso o tutor venha a subir ou descer de nível posteriormente, o total recebido anteriormente pelo tutor não é alterado.*

# 

# Arquitetura

Na arquitetura foi usado como foco padrão MVC(model-view-controller)através do Spring. O Spring MVC é um framework que ajuda no desenvolvimento de aplicações web, a partir dele é possível desenvolver aplicações tanto flexíveis quanto robustas. Ele é capaz de atender requisições http, e delegar responsabilidade de processamento de dados para outros componentes. O padrão MVC é dividido em camadas como mostra a figura 1:

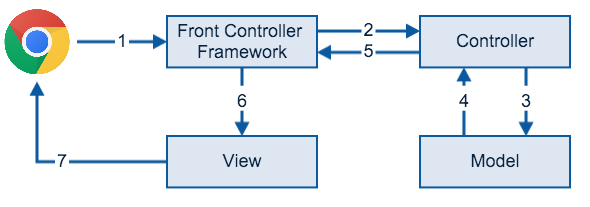


Figura 1. Camadas do padrão MVC

Os quais,

1. É acessada uma URL no browser que envia a requisição HTTP para o servidor que roda a aplicação web com Spring MVC. Quem recebe a requisição é o controlador do framework, o Spring MVC.

2. O controlador do framework irá procurar qual classe é responsável por tratar essa requisição, entregando a ela os dados enviados pelo browser. Essa classe faz o papel do controller.

3. O controller passa os dados para o model, que por sua vez executa todas as regras de negócio, como cálculos, validações e acesso ao banco de dados.

4. O resultado das operações realizadas pelo model é retornado ao controller.

5. O controller retorna o nome da view, junto com os dados que ela precisa para renderizar a página.

6. O Framework encontra a view que processa os dados, transformando o resultado em um HTML.

7. Finalmente, o HTML é retornado ao browser do usuário.

Componentes

* BackEnd:

1. Model
   1. Ajuda - Representa entidade ajuda.
   2. Aluno - Representa entidade aluno.
   3. Tutor - Representa entidade tutor.
2. Controller
   1. AvaliarTutorService - Interface para a US5.
   2. AvaliarTutorServiceImpl - Implementação da interface da US5.
   3. CadastrarAlunoService - Interface para a US1.
   4. CadastrarAlunoServiceImpl - Implementação da interface da US1.
   5. CadastrarPedidoAjudaService - Interface para a US4.
   6. CadastrarPedidoAjudaServiceImpl - Implementação da interface da US4.
   7. DefinirTutorService - Interface para a US2.
   8. DefinirTutorServiceImpl - Implementação da interface da US2.
   9. DisponibilizarHorarioService - Interface para a US3.
   10. DisponibilizarHorarioServiceImpl - Implementação da interface da US3.
   11. DoarService - Interface para a US6.
   12. DoarServiceImpl - Implementação da interface da US6.
3. RestApiController - Comunicação com o front-end.

# 

# 

# 

# Tecnologias

* Randoop - gera testes unitários usando a geração de testes aleatórios direcionados por feedback. Essa técnica, aleatoriamente, mas de maneira inteligente, gera seqüências de invocações de método / construtor para as classes em teste. O Randoop executa as sequências que cria, usando os resultados da execução para criar asserções que capturam o comportamento do seu programa. Randoop cria testes a partir das sequências de código e asserções. É uma ferramenta bastante útil pois gera um grande volume de testes em pouco tempo.

# Framework Spring Boot

Como descrito na arquitetura, o Spring Boot auxilia na interação Back-end~Front-end, dentro da classe RestApiController.java podemos ver alguns dos seguintes exemplos de uso:

1 POST - @RequestMapping(value = "/pedido/", method = RequestMethod.POST)

public ResponseEntity<?> CriarPedidoPresencial(@RequestBody Ajuda a, UriComponentsBuilder ucBuilder) throws ObjetoInexistenteException {

try {

cpService.pedirAjudaPresencial(a.getAluno().getMatricula(), a.getDisciplina(),

a.getHorario(), a.getDia(), a.getLocal());

} catch (Rep e) {

return new ResponseEntity<List>(HttpStatus.BAD\_REQUEST);

}

return new ResponseEntity<Ajuda>(a, HttpStatus.CREATED);

}

Métodos Post usualmente são chamados no intuito de algum tipo de cadastro. O front manda informações que são manipuladas pelo back.

2 GET - @RequestMapping(value = "/pedido/{id}", method = RequestMethod.GET)

public ResponseEntity<?> consultarPedido(@PathVariable("id") int id) throws ObjetoInexistenteException {

Ajuda a = cpService.getPedidoPorId(id);

if (a == null) {

return new ResponseEntity(new CustomErrorType("Pedido with id " + id

+ " not found"), HttpStatus.NOT\_FOUND);

}

return new ResponseEntity<Ajuda>(a, HttpStatus.OK);

}

Por outro lado, métodos Get são usados para fazer consultas, o front manda alguma informação e o back devolve outra a partir dela.

# Qualidade da Solução (ISO 25010)

## Funcionalidade

1. Completude - No escopo das USs selecionadas, especificações de backend foram cumpridas.
2. Adequação - Funções coesas e modularizadas.

## Eficiência

1. Tempo - Não apresentou delay sob testes feitos.

## Compatibilidade

1. Iteroperabilidade - Desenvolvimento MVC trás flexibilidade; troca versátil de informações com o Front-end.

## Usabilidade

1. Acessibilidade - Interface simples no Front, manipulável facilmente por diversos tipos de usuários.

## Manutenabilidade

1. Modularidade - Componentes bastante desmontáveis.
2. Reuso - Componentes coesos.
3. Modificabilidade - Uso intenso de interfaces facilita a manutenção e extensão do código.
4. Testabilidade - Modelo facilmente testável.

## 

Referências

<http://blog.algaworks.com/spring-mvc/>

data de acesso: 05/08/18

<https://randoop.github.io/randoop/>

data de acesso: 05/08/18

<http://iso25000.com/index.php/en/iso-25000-standards/iso-25010>

data de acesso: 05/08/18