Desafio Strider – Etapa 1. Candidato: Jhonatan Vinícius Mota Corrêa

Descrição:

Sistema de gerenciamento de tarefas: Criar um sistema no qual é possível cadastrar tarefas em uma interface web e concluir as mesmas em um aplicativo android. O servidor deverá armazenar as informações sobre a tarefa em um banco de dados MySQL e salvar a foto em arquivo, fornecendo os endpoints REST para a interface web e o client android. Ficar a vontade para criar funcionalidades que melhorem a experiencia do usuário.

Implementação:

Back end:

A implementação no backend foi feita com o Spring Tool Suite 4 e MySQL Community Server 8.0.16. O projeto foi criado utilizando a função Spring Starter Project, onde os arquivos já são configurados automaticamente para início. As dependências usadas foram colocadas no arquivo pom.xml (Jackson, Hibernate e afins).

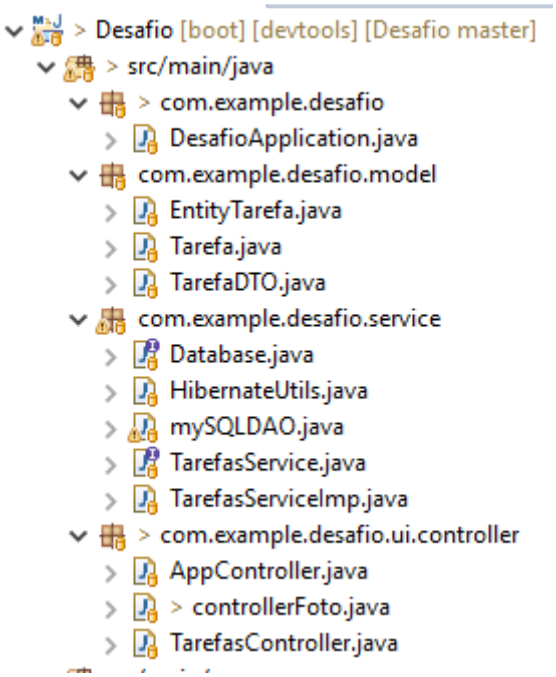
As configurações do Hibernate também foram condiguradas no arquivo src/main/resources/hibernate.cfg.xml como mostrado abaixo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLInnoDBDialect</property>
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/tarefas?serverTimezone=UTC</property>
        <property name="hibernate.connection.username">usuario</property>
        <property name="hibernate.connection.password">senha</property>

        <property name="hibernate.id.new_generator_mappings">true</property>
        <property name="show_sql">true</property>
        <property name="connection.pool_size">10</property>
        <property name="hibernate.hbm2ddl.auto">update</property>


        <mapping class="com.example.desafio.model.EntityTarefa" />

        </session-factory>
</hibernate-configuration>
```

Dependendo da máquina em que o banco de dados esteja rodando, talvez seja necessário configurar o URL e o dialect do MySQL, além de usuário e senha.

Ainda sobre o banco de dados, o Hibernate consegue gerar entradas e até colunas, mas não o esquema e a tabela, portanto estes deverão ter sido criados previamente no MySQL :

Como tarefaID é primary key da tabela, é interessante cria-la junto com a tabela.



O Back End tem a organização de arquivos ao lado. No pacote com.example.desafio temos apenas o arquivo de inicialização do Spring Boot da aplicação.

No pacote model temos as classes que descrevem a tarefa:

EntityTarefa é a classe que contém os atributos que o Hibernate salvará no banco de dados. Ela contém os seguintes atributos:

- tarefaID: Identificação da tarefa e chave primária (gerada automaticamente pelo Hibernate).
- Concluido: se a tarefa foi concluída.
- Nome: nome da tarefa.
- imgSrc: Caminho local do arquivo onde a imagem foi salva.

A classe Tarefa é responsável por receber os dados do front end da tarefa, sendo basicamente um espelho da EntityTarefa, porém sem o ID, já que esse é gerado pelo Hibernate que só acessa a ultima classe.

TarefaDTO serve apenas para transferir as características de Tarefa para EntityTarefa, tendo as mesmas características da primeira.

No pacote service temos as classes e interfaces responsáveis pela comunicação entre as classes do model e o banco em si.

HibernateUtils é responsável por abrir a comunicação via Hibernate, mySQLDAO tem as implementações dos métodos responsáveis pelas atividades no banco de dados e TarefasServiceImp tem os métodos para a comunicação entre os controllers e o banco de dados.

O pacote controllers tem os RestControllers responsáveis por criar os EndPoints Rest. O TarefasController é responsável por gerar o envio e recebimento das tarefas, controllerFoto por gerar a comunicação das imagens e AppController tem apenas um endpoint para informar ao aplicativo que a comunicação que a comunicação com o servidor foi iniciada.

Quando uma nova tarefa é criada, automaticamente seu status é definido para não concluído e o caminho da imgSrc para NULL e então salva no banco de dados. Quando a tarefa é completada pelo aplicativo, o status é mudado para concluído e a imagem recebida é salva como um arquivo png na pasta /Uploads, na raíz do projeto, sendo esse caminho repassado ao banco de dados.

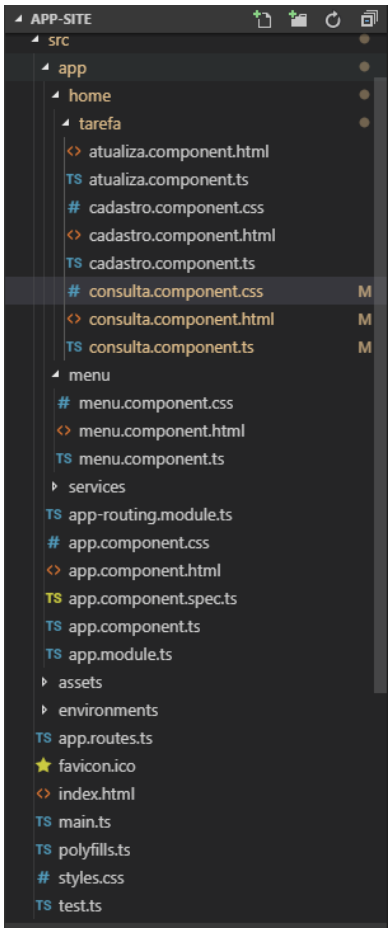
Quando a listagem de tarefas é solicitada, o servidor envia uma lista JSON com as informações das tarefas cadastradas. Também quando solicitada uma imagem de alguma tarefa, o servidor busca o endereço salvo no banco de dados, carrega essa imagem e repassa como um recurso web.

Front End.

```
C:\Users\Jhon\Documents>npm -v
6.9.0

C:\Users\Jhon\Documents>node -v
v10.16.0

C:\Users\Jhon\Documents>
```

O frontend foi implementado com com Angular CLI com NodeJS e NPM.

A organização dos arquivos ficou como mostrado abaixo:



O arquivo “index.html”, na raiz do projeto, tem o template da página que será renderizada pelo angular. Ela conta com um menu com as entradas básicas e a página a ser renderizada (Cadastro ou consulta de tarefas).

As rotas dos aplicativos estão no arquivo app.routes.ts. O menu está na pasta app/menu, e é basicamente uma lista horizontal com os caminhos para a consulta e cadastro de tarefas (a atualização de tarefas foi um componente desenvolvido para teste do servidor e que não é utilizado na versão final).

Os componentes estão na pasta app/home/tarefa, e contam com as views para o cadastro e consulta de tarefas. A consulta consiste de uma listagem das tarefas cadastradas no banco de dados, com opção para visualização da imagem das tarefas já concluídas.

O cadastro consiste de um formulário para a inserção do nome de uma nova tarefa, solicitando ao servidor a criação desta.

Há ainda o componente de serviço, que é responsável pelas requisições HTTP ao servidor, na pasta app/services e a classe de tarefa, que recebe os dados da tarefa do servidor para serem interpretadas pelo Angular.

O front end foi desenvolvido tomando por base que tanto o Angular quanto o Spring rodam no mesmo servidor, portanto a comunicação é feita diretamente com http://localhost:8080/, que é a porta padrão do Spring. Se rodarem em máquinas diferentes, talvez seja necessário alterar isso no componente de serviço do front end.

App Android.

O app foi desenvolvido com o Android Studio. Ele conta basicamente com três activitys:

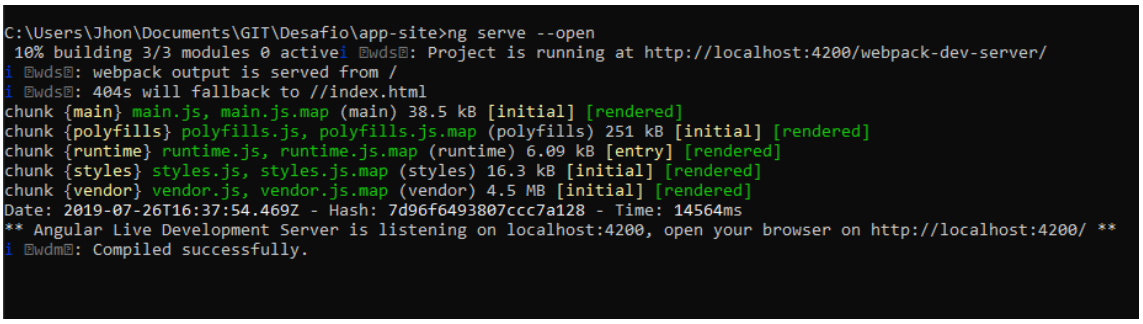
- A MainActivity serve como uma inicialização da comunicação com o servidor e do aplicativo em si. Ela conta com uma caixa de texto para se inserir o IP do servidor e um botão para entrar (começar a comunicação). Novamente foi desenvolvido tomando por base de que o Spring roda na porta padrão 8080. Talvez seja necessário ajustar isso se o servidor não seguir essa premissa.

- A acitivity Menu conta com uma lista das tarefas cadastradas. Quando essa view é carregada, o App busca no servidor uma lista das tarefas carregadas. Essa lista é recebida via JSON e interpretada via Jackson em uma ArrayList de objetos Tarefa (classe criada no App). Essa ArrayList é passada então para uma RecyclerView através de uma Adapter que mostra a listagem para o usuário. Nesse Recycler, as tarefas que ainda não estão concluídas contam com a opção de mandar serem concluídas, levando para a terceira e última activity.
- A FotoActivity é responsável por tirar uma foto e com o celular e manda-la para o servidor, concluído assim a tarefa. A foto é mandada como um arquivo MultiPart via HTTP Post para o servidor, que automaticamente salva a imagem da tarefa correspondente e atualiza seu status para concluído. A atividade retorna então para o Menu de tarefas.

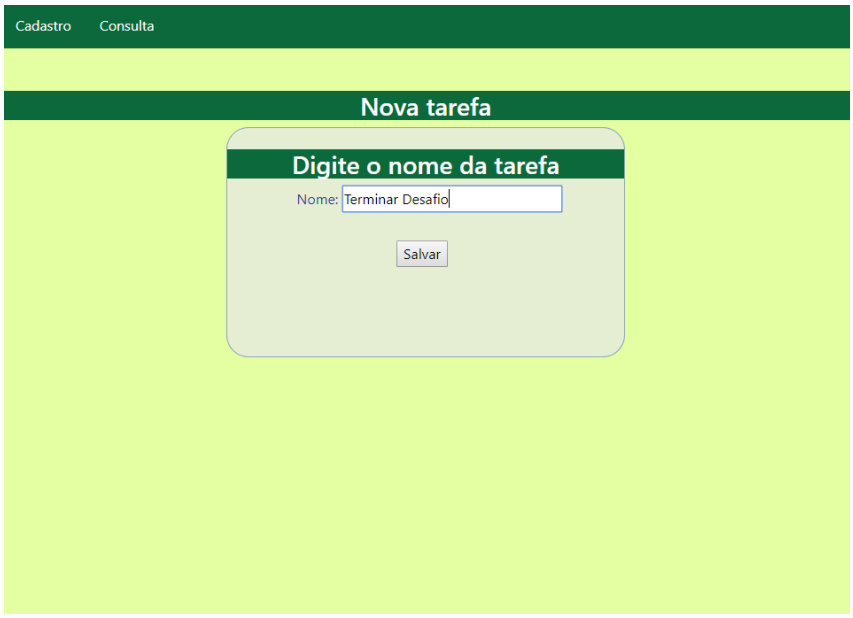
Resultados e Uso:

Com tudo implementado, banco de dados criado, já é possível executar a aplicação. No Spring, basta executar o projeto como Spring Boot App e o back end estará online na porta 8080:

O Front End também, basta executar via prompt o comando “ng serve –open” na pasta raiz do projeto angular:

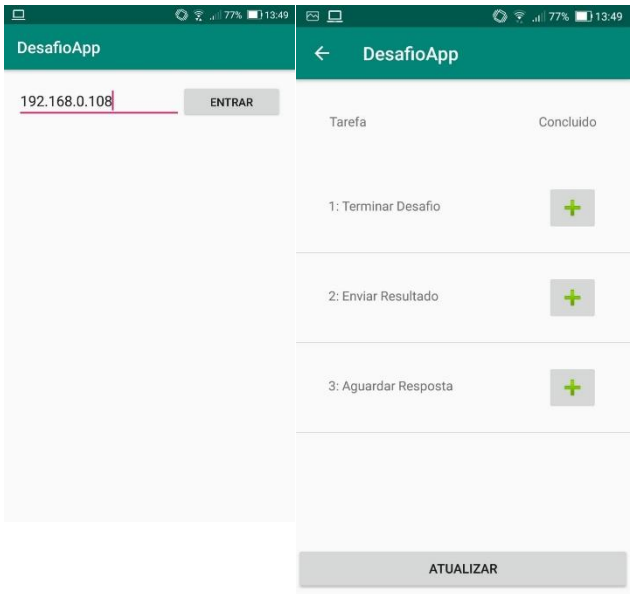


Com isso o Angular já deve abrir a pagina principal do projeto no navegador padrão, através de http://localhost:4200/, como mostrado ao lado. Como não há nenhuma tarefa para mostrar, A página estará vazia. Podemos então cadastrar nossas primeiras tarefas a partir do link no topo da página:



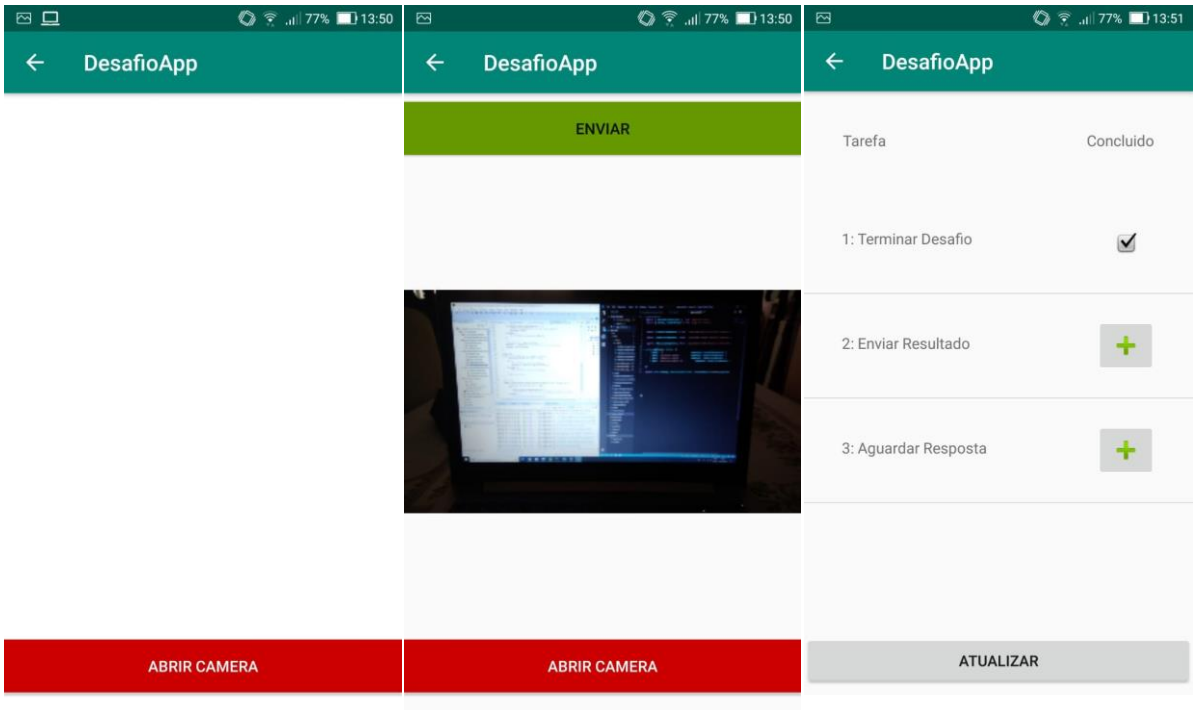
Com isso temos nossa primeira tarefa cadastrada já no banco de dados. Podemos cadastrar novas tarefas assim como necessário.

Com isso já é possível acessar e concluir as tarefas pelo aplicativo. A primeira página pede o IP do servidor. No caso de uso, como o servidor e o smartphone estavam na mesma rede, o IP local era 192.168.0.108:



Como nenhuma das tarefas foi concluída, em todas aparecem a opção de concluir.

Clicando nessa opção somos direcionados a página de conclusão, onde temos a opção de tirar uma foto:



Com isso já podemos atualizar a página do front end e visualizar o resultado da tarefa: