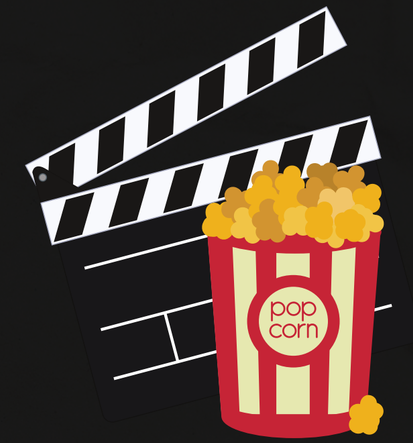
Uma Produção de:

Arthur Gonçalves
Breno Amorim
Gustavo Garcia
Victor Hugo

SPRINT 4

MOVIEMETR



BANCO DE DADOS

BACK-END

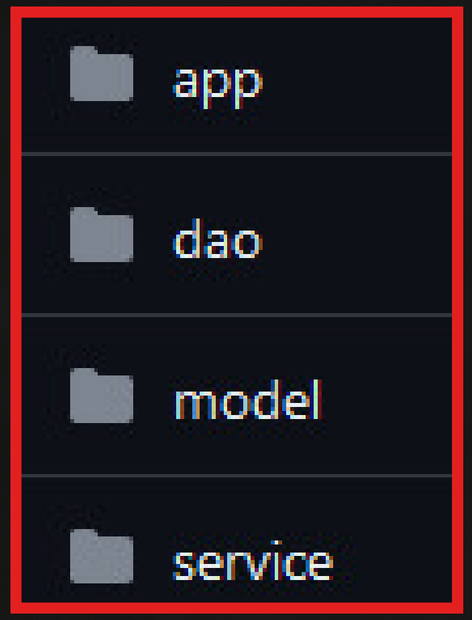
OVERVIEW

Utiliza as classes Model, Data Access Object, Service e Aplicação

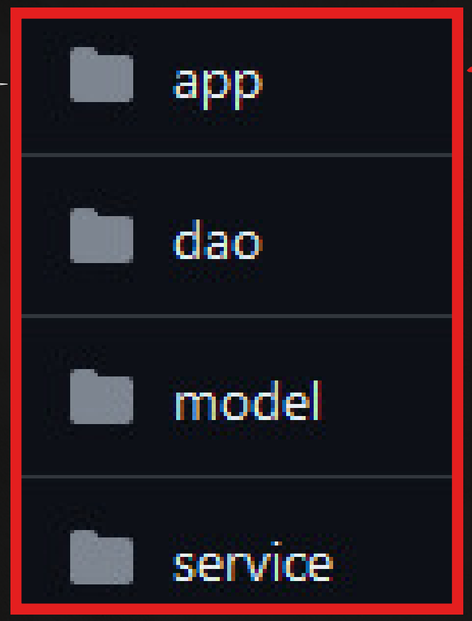
Banco de Dados feito com PostgreSQL

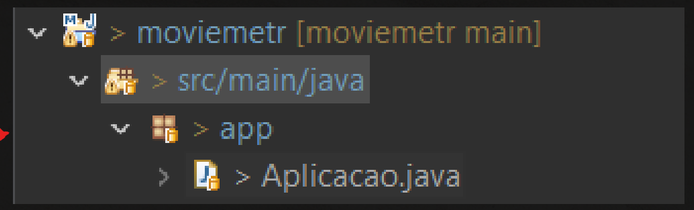
O arquivo DAO.java faz a conexão com o banco de dados

ORGANIZAÇÃO DAS PASTAS



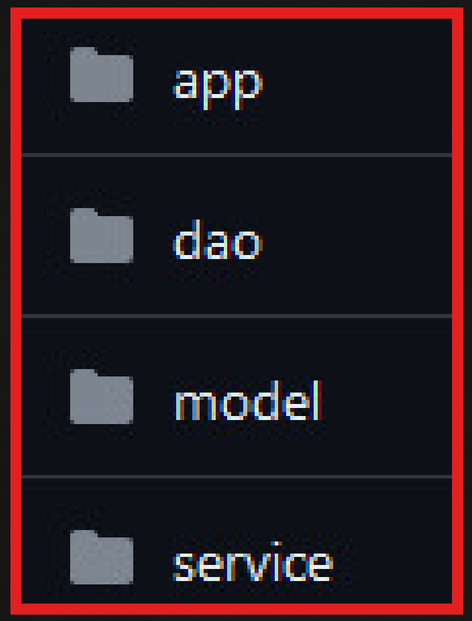
ORGANIZAÇÃO DAS PASTAS

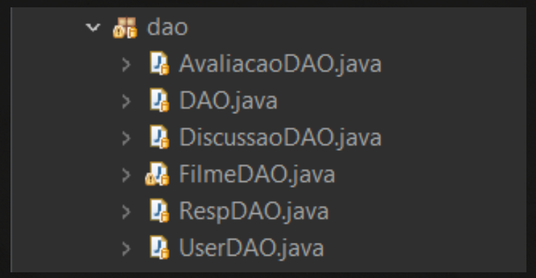





```
// login/cadastro
get("/login", (request, response) -> loginService.makeForm(request, response));

get("/criarConta", (request, response) -> signinService.makeForm(request, response));

get("/logout", (request, response) -> {
    request.session().invalidate();
    response.redirect("/login");
    return null;
});

post("/logar", (request, response) -> loginService.login(request, response));

post("/registrar", (request, response) -> signinService.registrar(request, response));
// ---------

get("/", (request, response) -> discussaoService.getAll(request, response));


get("/detalhes/:id/:type", (request, response) -> filmeService.makeFilme(request, response));

get("/disc/:id", (request, response) -> discussaoService.getDisc(request, response));

get("/delete/:id", (request, response) -> discussaoService.delete(request, response));

get("/deleteResp/:id", (request, response) -> discussaoService.deleteResp(request, response));

post("/avaliar", (request, response) -> filmeService.insert(request, response));

post("/criar", (request, response) -> discussaoService.insert(request, response));

post("/criarResp", (request, response) -> discussaoService.insertResp(request, response));
```

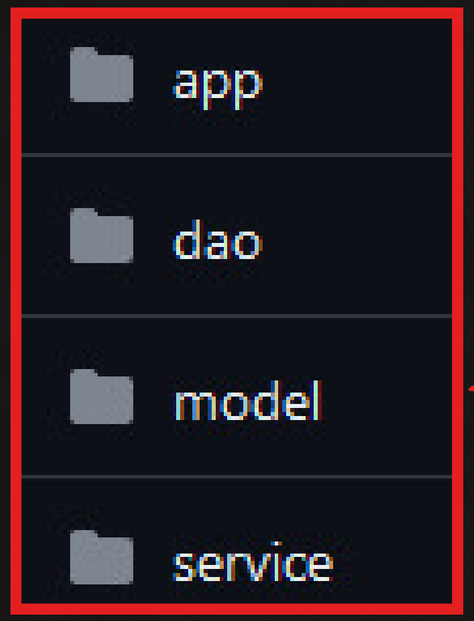
ORGANIZAÇÃO DAS PASTAS

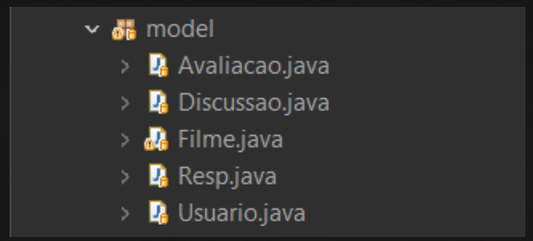





```java
public boolean insert(Avaliacao avaliacao) {
    boolean status = false;
    try {
        String sql = "INSERT INTO avaliacao (valor, id_usr, id_filme) "
                   + "VALUES (" + avaliacao.getValor() + ", " + avaliacao.getId_usr() + ", " + avaliacao.getId_film
        PreparedStatement st = conexao.prepareStatement(sql);
        st.executeUpdate();
        st.close();
        status = true;
    } catch (SQLException u) {
        throw new RuntimeException(u);
    }
    return status;
}

/*
 *  RETRIEVE
 */

public Avaliacao get(int id) {
    Avaliacao avaliacao = null;

    try {
        Statement st = conexao.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);
        String sql = "SELECT * FROM avaliacao WHERE id="+id;
        ResultSet rs = st.executeQuery(sql);
        if(rs.next()){
            avaliacao = new Avaliacao(rs.getInt("id"), rs.getInt("valor"),  rs.getInt("id_usr"),  rs.getInt("id_fi
        }
        st.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
```

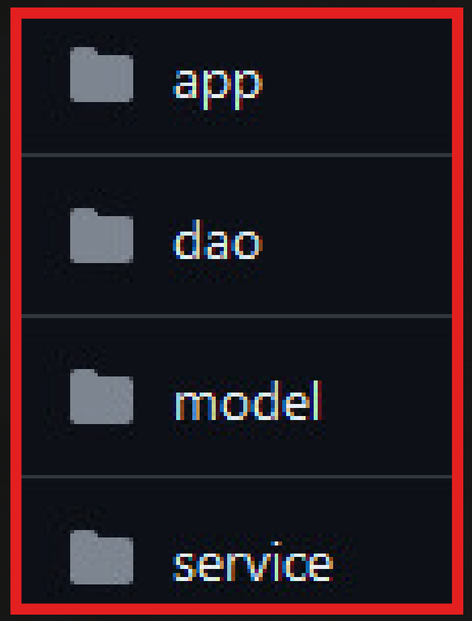
ORGANIZAÇÃO DAS PASTAS

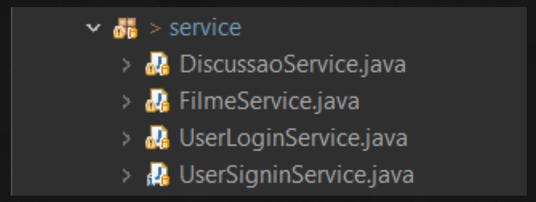





```java
public Discussao() {
    id = -1;
    titulo = "nao definido";
    conteudo = "nao definido";
    autor = "nao definido";
    curtidas = -1;
    data = "nao definido";
}

public Discussao(int id, String titulo, String conteudo, String autor, int curtidas, String data) {
    setId(id);
    setTitulo(titulo);
    setConteudo(conteudo);
    setAutor(autor);
    setCurtidas(curtidas);
    setData(data);
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}


public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
```

ORGANIZAÇÃO DAS PASTAS

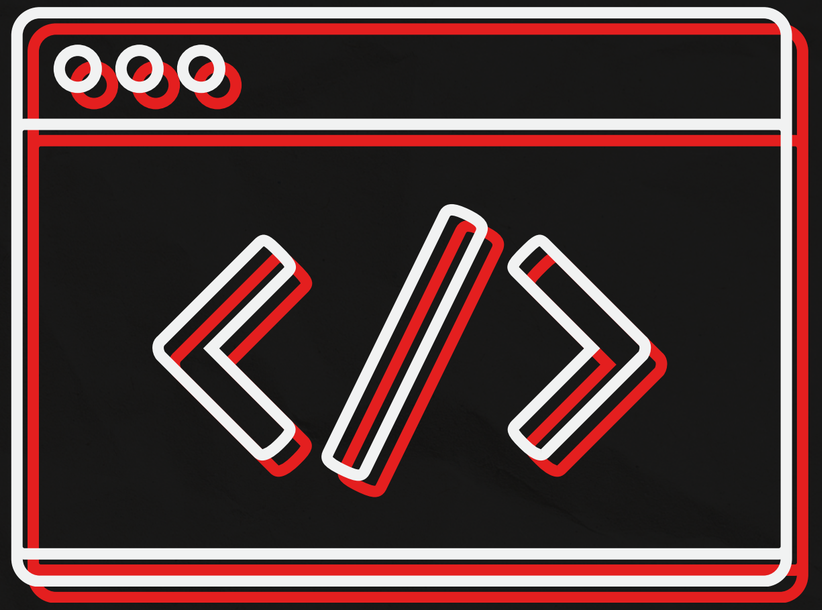





```java
public int[] getValuesFromAzureML() {
    int idUsr = 1;
    List<Avaliacao> movieId =  avalDAO.getIdsByUsr(0);
    if (usuarioGlobal != null) {
        idUsr = userDAO.getByNome(usuarioGlobal).getId();
        movieId = avalDAO.getIdsByUsr(idUsr);
    }
    //System.out.println(movieId);

    int[] resultValues = null;
       try {
           String inputData = "";
           // Your code for URL, API key, and input data
           String url = "https://ussouthcentral.services.azureml.net/workspaces/82f549a0cf89466bbac338c6e30ebcf3/s
           String apiKey = "nEqm0BF85BjLLjm3bZvdYrZzXCyNaDUagpyxy8GVIOlqZgBCtgn7CfM6XuwANf/OxN11g86PtNW7+AMC1nh+sw
           if(movieId.size()>4) {
               inputData = "{\"Inputs\": {\"input1\": {\"ColumnNames\": [\"Col1\", \"Col2\", \"Col3\", \"Col4\", \
                       "\"Values\": [[" + idUsr + ", \"value\"," + movieId.get(movieId.size()-1).getId_filme() + "
                       "[" + idUsr + ", \"value\", " + movieId.get(movieId.size()-2).getId_filme() + ", \"value\",
                       "[" + idUsr + ", \"value\", " + movieId.get(movieId.size()-3).getId_filme() + ", \"value\",
                       "[" + idUsr + ", \"value\", " + movieId.get(0).getId_filme() + ", \"value\", " + movieId.ge
                       "[" + idUsr + ", \"value\", " + movieId.get(1).getId_filme() + ", \"value\", " + movieId.ge

           } else {
               inputData = "{\"Inputs\": {\"input1\": {\"ColumnNames\": [\"Col1\", \"Col2\", \"Col3\", \"Col4\", \
           }

           // Create a connection
           URL obj = new URL(url);
           HttpURLConnection con = (HttpURLConnection) obj.openConnection();
```

INTEGRAÇÃO

FRONT-END



OVERVIEW

Feita em código Java, por meio das classes Service

Utiliza-se o Spark Framework para exibir os formulários no front-end

O arquivo ‘Aplicação.java’ lida com as requisições do back-end

INTEGRAÇÃO
FRONT-END

Aplicacao.java

Spark

Metodo Service

Pagina de Login:

```
DE
ect  Run  Window  Help
o.java ×

    // login/cadastro
    get("/login", (request, response) -> loginService.makeForm(request, response));

    get("/criarConta", (request, response) -> signinService.makeForm(request, response));

    get("/logout", (request, response) -> {
        request.session().invalidate();
        response.redirect("/login");
        return null;
    });

    post("/logar", (request, response) -> loginService.login(request, response));

    post("/registrar", (request, response) -> signinService.registrar(request, response));
    // ---------

    get("/", (request, response) -> discussaoService.getAll(request, response));


    get("/detalhes/:id/:type", (request, response) -> filmeService.makeFilme(request, response));

    get("/disc/:id", (request, response) -> discussaoService.getDisc(request, response));

    get("/delete/:id", (request, response) -> discussaoService.delete(request, response));

    get("/deleteResp/:id", (request, response) -> discussaoService.deleteResp(request, response));

    post("/avaliar", (request, response) -> filmeService.insert(request, response));

    post("/criar", (request, response) -> discussaoService.insert(request, response));

    post("/criarResp", (request, response) -> discussaoService.insertResp(request, response));
```

Pagina de Login:

Metodo Service

Formulário

Resposta

```java
public class UserLoginService {

    private static UserDAO userDAO = new UserDAO();
    private String form;


    public Object makeForm(Request request, Response response) {
        // Carregue o formulário de login (HTML) e substitua o valor de 'form' aqui
        // O código do formulário de login deve estar aqui
        form =  "<style>" + //
                "                           body {" + //
                "                               font-family: Arial, sans-serif;" + //
                "                               background-color: rgb(48,48,48);" + //
                "                               margin: 0;" + //
                "                               padding: 0;" + //
                "                               display: flex;" + //
                "                               align-items: center;" + //
                "                               justify-content: center;" + //
                "                               height: 100vh;" + //
                "                           }" + //

                "                           .formulario {" + //
                "                               background-color: rgb(0,0,0);;" + //
                "                               padding: 20px;" + //
                "                               border-radius: 8px;" + //
                "                               box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);" + //
                "                               width: 300px;" + //
```

INTEGRAÇÃO
FRONT-END

Resposta

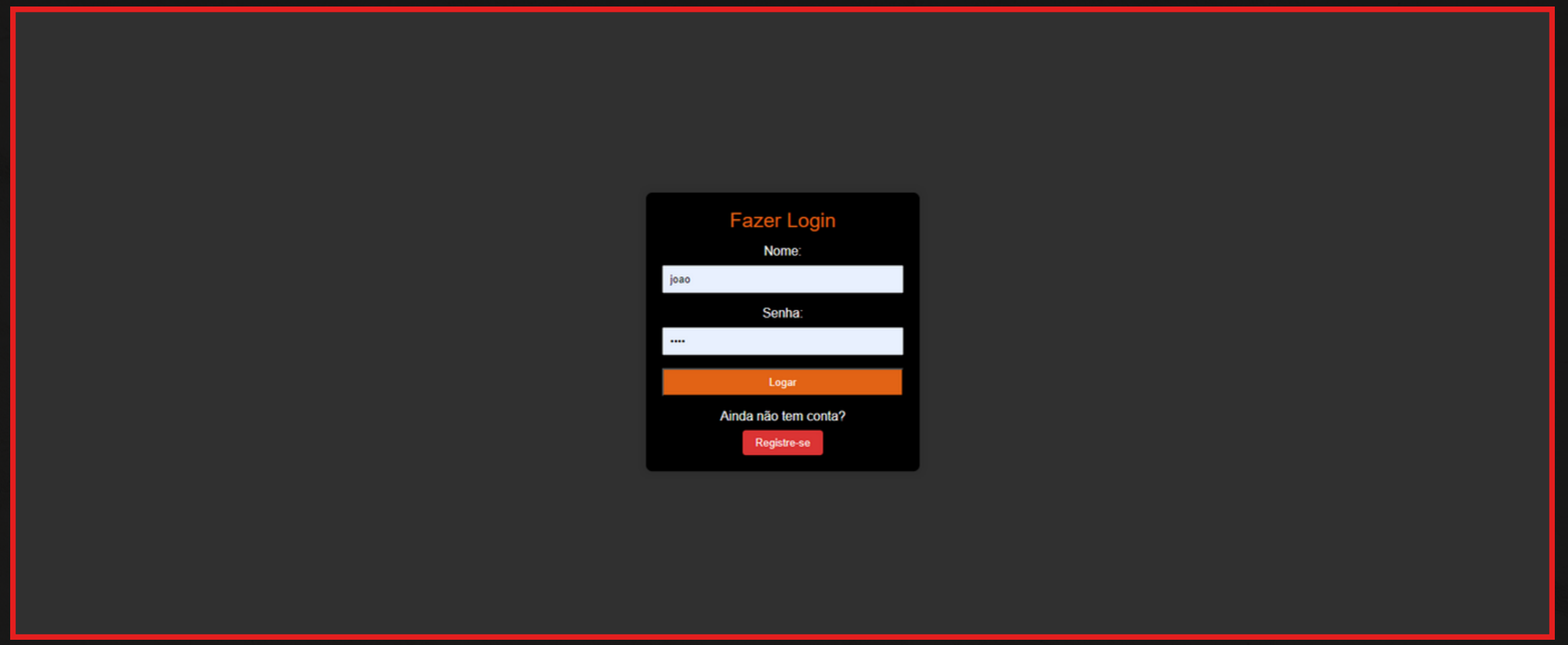
Aplicaçao/Spark

Postagem Na DB

Pagina de Login:

```
DE
ect  Run  Window  Help
o.java ×

    // login/cadastro
    get("/login", (request, response) -> loginService.makeForm(request, response));

    get("/criarConta", (request, response) -> signinService.makeForm(request, response));

    get("/logout", (request, response) -> {
        request.session().invalidate();
        response.redirect("/login");
        return null;
    });

    post("/logar", (request, response) -> loginService.login(request, response));

    post("/registrar", (request, response) -> signinService.registrar(request, response));
    // ---------

    get("/", (request, response) -> discussaoService.getAll(request, response));


    get("/detalhes/:id/:type", (request, response) -> filmeService.makeFilme(request, response));

    get("/disc/:id", (request, response) -> discussaoService.getDisc(request, response));

    get("/delete/:id", (request, response) -> discussaoService.delete(request, response));

    get("/deleteResp/:id", (request, response) -> discussaoService.deleteResp(request, response));

    post("/avaliar", (request, response) -> filmeService.insert(request, response));

    post("/criar", (request, response) -> discussaoService.insert(request, response));

    post("/criarResp", (request, response) -> discussaoService.insertResp(request, response));
```

Pagina de Login:



SISTEMA INTELIGENTE

RECOMENDADOR

OVERVIEW

Treinado por meio de Machine Learning no Azure

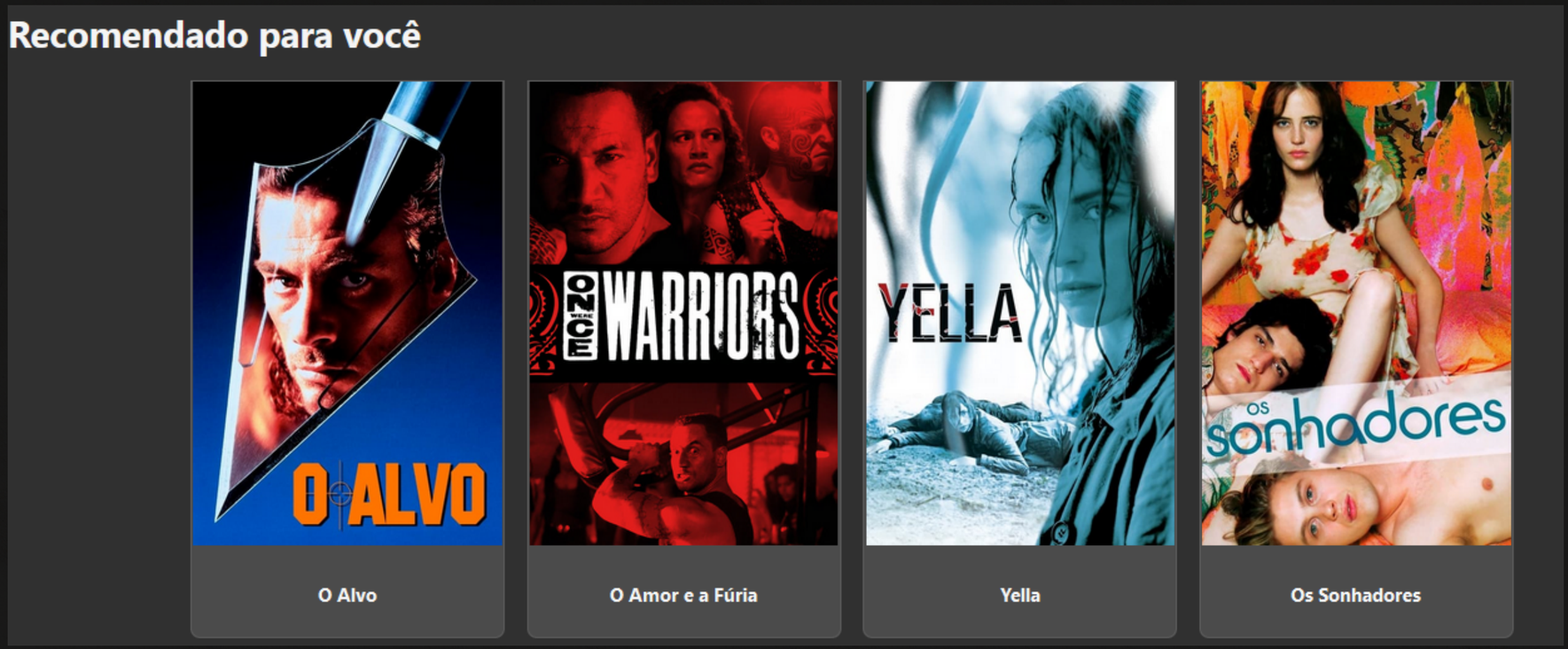
Utiliza as avaliações do úsuario e recomenda filmes curtidos por outros úsuarios com perfís semelhantes (Collaborative Filtering)

Treinado com a base de dados ‘Movie Lens’

SISTEMA INTELIGENTE

RECOMENDADOR

DENTRO DO WEBAPP



SISTEMA INTELIGENTE

RECOMENDADOR

INTELLIGENT SYSTEM CANVAS

Ferramental de IA

O nosso sistema inteligente recomendará filmes a usuarios do site. Como o site terá foco na discussão e na analize de recomendações. planejamos usar o sistema Collaborative Filtering, para que os próprios usuários possam recomendar séries e filmes a outros usuários.

Entradas

A entrada consistirá em dados de filmes, e o 'peso' de certos gêneros de filme acerca da preferência de um determinado úsuario. Por exemplo: Um determinado úsuario, baseado em seu histórico, tem os seguintes pares peso-genêro: (Ação:0.8); (Drama:0.86); (Sci-fi:0.44); (Terror:0.02).

Saídas

Ainda no exemplo da entrada, o sistema inteligente tentará recomendar filmes que contenham Ação e Drama. O sistema não se importará com filmes sci-fi e evitará de recomendar filmes de terror.

Proposição de valor

Facilitar o descobrimento de séries e filmes que contenham narrativas que se encaixam em um determinado perfil de úsuario.

Equipe

Arthur Moraes - Banco de dados(back end)

Gustavo Garcia - Html(front end)

Breno Amorim - Pesquisa e Documentação

Victor Braz - Pesquisa e Documentação

Stakeholders Chaves

Estudio de filmes

Usuário

The Moviedb

Clientes

Amantes do cinema

Amantes de series

Criticos de cinema

Jovens adultos

Qualquer usuario adolescente ou mais velho

Custos

o custo levantado até o momento é aproximadamente $5 (USD), uma vez que as ferramentas que utilizaremos para tal disponibilizadas pela azure

Receitas

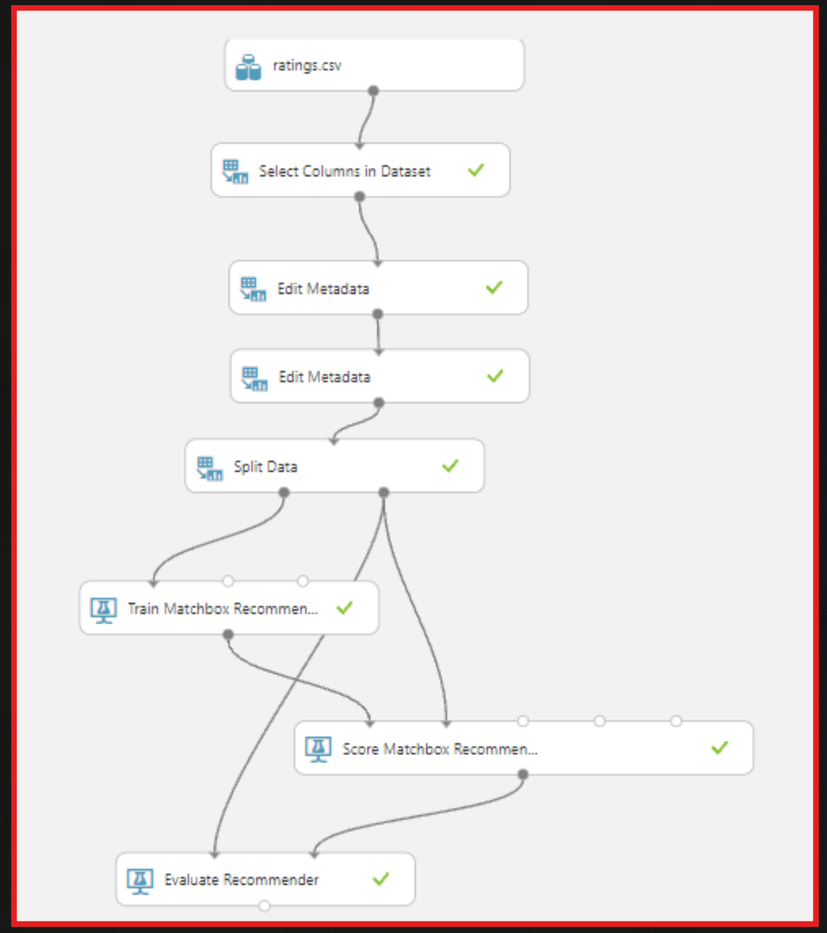
O retorno que o sistema inteligente trás ao projeto é uma facilitação de certos processos e uma melhoria de qualidade de vida no projeto.

miro

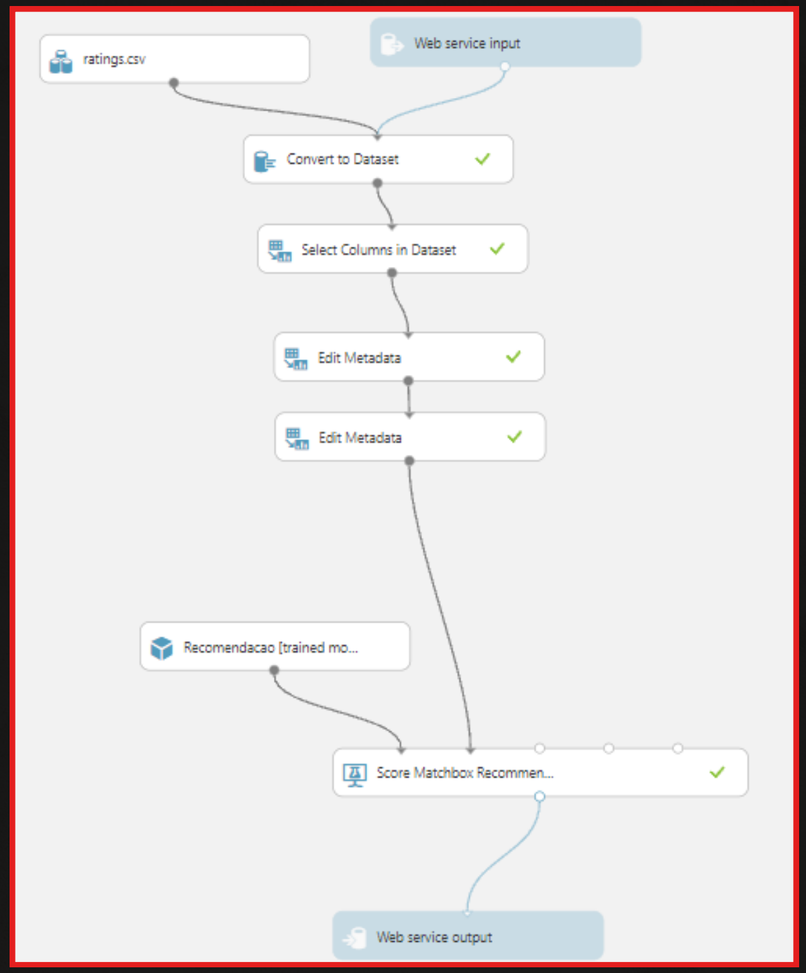
SISTEMA INTELIGENTE

RECOMENDADOR

MACHINE LEARNING



AGENTE IMPLEMENTADO



SEGURANÇA

SQL E CRIPTOGRAFIA

CRIPTOGRAFIA MD5

```
39
40      public Object registrar(Request request, Response response) throws NoSuchAlgorithmException {
41          MessageDigest m=MessageDigest.getInstance("MD5");
42
43          String nome = request.queryParams("nome");
44          String senha = request.queryParams("senha");
45          m.update(senha.getBytes(),0,senha.length());
46          senha = new BigInteger(1,m.digest()).toString(16);
47
48          //System.out.println(nome);
49          //System.out.println(userDAO.getByNome(nome));
50
51
52          if (nome != null && senha != null && !nome.equals(userDAO.getByNome(nome))) {
53              Usuario user = new Usuario(-1, nome, senha);
54              if (userDAO.insert(user)) {
55                  response.redirect("/login");
56                  return "Usuário " + no
57              }
58          }else {
59              response.status(400); //
60              return "Fa    ao   astra
61          }
62          return "a";
63      }
```

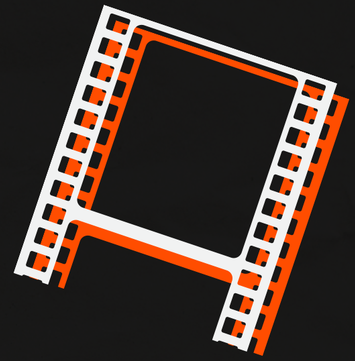
id [PK] integer	nome text	senha text
30	alirio	81dc9bdb52d04dc20036dbd8313ed0...
31	joao	81dc9bdb52d04dc20036dbd8313ed0...
32	marceloXD	fa246d0262c3925617b0c72bb20eeb1d

PREPARED STATEMENT

```java
public boolean insert(Usuario user) {
    boolean status = false;

    try {
        String sql = "INSERT INTO usuario (nome, senha) VALUES (?, ?)";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setString(1, user.getNome());
        st.setString(2, user.getSenha());
        st.executeUpdate();
        st.close();
        status = true;
    } catch (SQLException u) {
        throw new RuntimeException(u);
    }
    return status;
}
```

Prepared Statements são utilizados para prevenir ataques de injeção SQL

MOVIEMETR



OBRIGADO

https://github.com/ArthurGMoraes/moviemetr