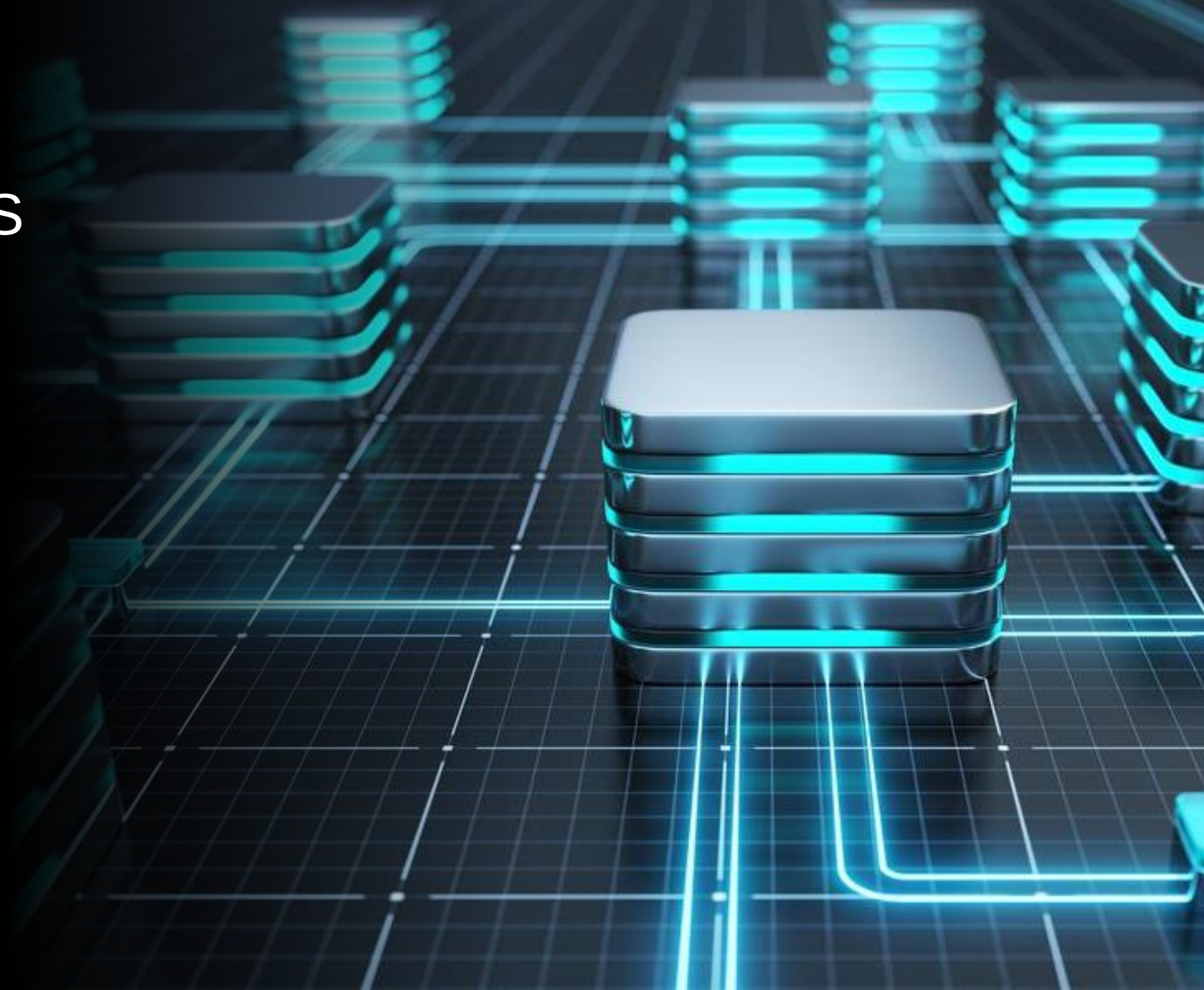
SpeedFanatic - O Portal da velocidade.

SpeedFanatic é um portal intuitivo para os amantes de Fórmula 1. Estamos aqui para oferecer facilidade de acesso às informações sobre o esporte, permitindo que até mesmo os novatos se sintam à vontade em mergulhar nesse mundo.

Integrantes:
Diego Polanski;
Guilherme de Almeida;
Pedro Marçal;
Pedro Ribeiro;

CONSIDERAÇÕES INICIAIS

- Considerações do projeto;
- Nossa motivação;
- O impacto do SpeedFanatic para a comunidade.



NOSSO SISTEMA INTELIGENTE

- Sistema de moderação de fórum;
- Serviços utilizados;
- Hospedagem do Sistema;
- Segurança

IS CANVAS - IA para Moderação de Fórum

Ferramental IA

Banco de dados;
Machine Learning;
API do Microsoft Azure.

Entradas
Perguntas dos usuários no fórum;
Respostas dos usuários do fórum;
Base de dados com palavras ofensivas.

Saídas
Exclusão de conteúdo;
Feedback para usuários.

Proposição de Valor

Maior segurança;
Facilidade de moderação;
Melhor experiência do usuário;
Redução de problemas legais;
Conformidade com as diretrizes.

Equipe
A equipe raramente entrara em contato com o usuário, ja que a IA removerá o conteúdo ofensivo. Caso algo passe desapercebido, o usuário poderá entrar em contato conosco.

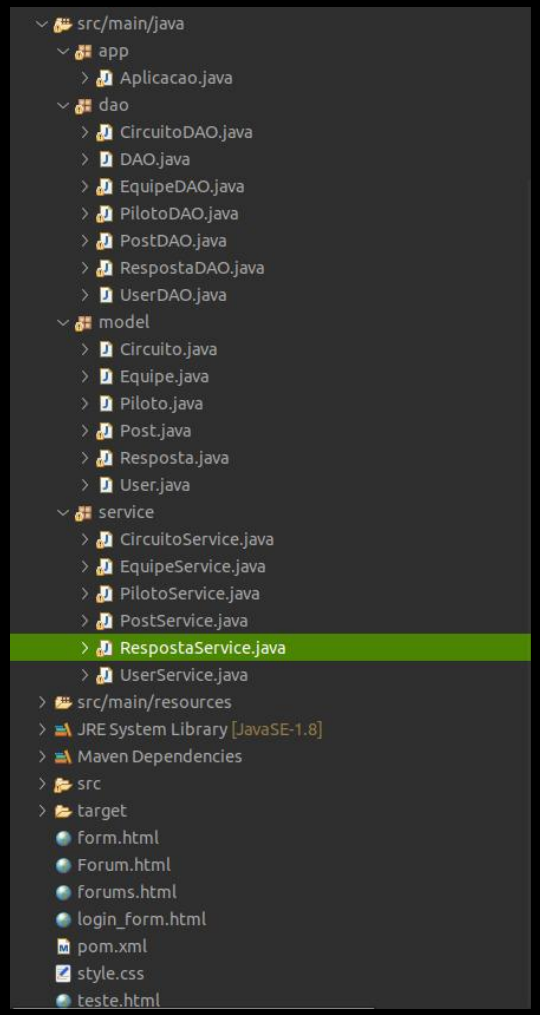
Stakeholders Chaves
Usuários do fórum;
Desenvolvedores;
Moderadores;

Clientes
Fãs de Formula 1 e do automobilismo em geral de todas as idades que querem se inteirar em uma comunidade apaixonada pelo esporte.

Custos
Custo operacional da API no Microsoft Azure.

Receitas
Não haverá receitas diretamente envolvidas no sistema inteligente, apenas com propagandas e patrocínios de terceiros.

UM POUCO DOS NOSSOS CODIGOS




```java
public User authenticate(String email, String senha, Request request) {
    User user = null;
    try {
        String sql = "SELECT * FROM \"user\" WHERE user_email = ? AND user_senha = ?";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setString(1, email);
        st.setString(2, senha);
        ResultSet rs = st.executeQuery();
        if (rs.next()) {
            user = new User(
                rs.getInt("user_id"),
                rs.getString("user_username"),
                rs.getString("user_email"),
                rs.getString("user_senha"),
                rs.getDate("user_data_criacao").toLocalDate(),
                rs.getInt("user_piloto"),
                rs.getInt("user_equipe"),
                rs.getInt("user_nota")
            );
            System.out.println("NOTA NO LOGIN" +user.getNota());

            // Crie um token JWT e armazene-o na sessão
            String token = criarTokenJWT(user);
            System.out.println(token);
            request.session().attribute("usuarioAutenticado", token);
            request.session().attribute("user_id", user.getID());
            request.session().attribute("user_username", user.getUsername());
            request.session().attribute("user_nota", user.getNota());
        }
        st.close();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    return user;
}
```

UM POUCO DOS NOSSOS CODIGOS

```sql
--
-- Name: user_user_id_seq; Type: SEQUENCE; Schema: public;
--

CREATE SEQUENCE public.user_user_id_seq
    START WITH 2
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;



--
-- Name: user; Type: TABLE; Schema: public;
--

CREATE TABLE public."user" (
    user_id integer DEFAULT nextval('public.user_user_id_seq'::regclass) NOT NULL,
    user_username character varying(16) NOT NULL,
    user_email character varying(255),
    user_senha character varying(32) NOT NULL,
    user_data_criacao timestamp with time zone DEFAULT CURRENT_TIMESTAMP,
    user_piloto integer NOT NULL,
    user_equipe integer NOT NULL,
    user_nota integer
);
```

```java
private static boolean verificarTextoModerador(String texto) {
    // Configuração da API da Microsoft Content Moderator
    String subscriptionKey = "270d010b411140d68c67f46ac838e902";
    String endpoint = "https://moderadorforum.cognitiveservices.azure.com/contentmoderator/moderate/v1.0/ProcessTe

    try {
        HttpClient httpclient = HttpClients.createDefault();

        URIBuilder builder = new URIBuilder(endpoint);
        builder.setParameter("autocorrect", "false");
        builder.setParameter("PII", "false");
        builder.setParameter("listId", "");
        builder.setParameter("classify", "True");
        builder.setParameter("language", "");

        HttpPost request = new HttpPost(builder.build());
        request.setHeader("Content-Type", "text/plain");
        request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey);

        StringEntity reqEntity = new StringEntity(texto);
        request.setEntity(reqEntity);

        HttpResponse response = httpclient.execute(request);
        HttpEntity entity = response.getEntity();

        if (entity != null) {
            String responseText = EntityUtils.toString(entity);
            System.out.println(responseText);

            // Verifique se a resposta contém um termo inadequado
            if (responseText.contains("\"Terms\":null")) {
                // A resposta indica que o texto é aprovado
                return true;
            } else {
                // A resposta indica que o texto não é aprovado
                System.out.println("Texto não aprovado pelo Content Moderator.");
                return false;

            }
```

UM POUCO DOS NOSSOS CODIGOS

```java
public class User {
    private int id;
    private String username;
    private String email;
    private String senha;
    private LocalDate dataCriacao;
    private int piloto;
    private int equipe;
    private int nota;

    public User() {
        id = 1;
        username = "";
        email = "";
        senha = "";
        dataCriacao = LocalDate.now();
        piloto = 1;
        equipe = 1;
        nota = 3;
    }

    public User(int id, String username, String email, String senha, LocalDate dataCriacao, int piloto, int equipe, int nota) {
        setID(id);
        setUsername(username);
        setEmail(email);
        setSenha(senha);
        setDataCriacao(dataCriacao);
        setPiloto(piloto);
        setEquipe(equipe);
        setNota(nota);
    }
```

```java
private static String criptografarSenhaMD5(String senha) {
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(senha.getBytes());
        byte[] digest = md.digest();

        // Converte o hash MD5 para uma representação hexadecimal
        StringBuilder hexString = new StringBuilder();
        for (byte b : digest) {
            hexString.append(String.format("%02x", b & 0xff));
        }

        return hexString.toString();
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException("Erro ao criar hash MD5", e);
    }
}
```

UM POUCO DOS NOSSOS CODIGOS

```java
public Object login(Request request, Response response) {
    String username = request.queryParams("email");
    String senha = request.queryParams("senha");
   System.out.println(username);
   System.out.println(senha);
    String resp = "";
    senha = criptografarSenhaMD5(senha);
    System.out.println(senha);
    // Verifique se o usuário com as credenciais fornecidas existe no banco de dados
    User user = userDAO.authenticate(username, senha, request);

    if (user != null) {
        resp = "Login bem-sucedido para o usuário " + username;
        response.status(200); // 200 OK
        response.redirect("/perfil");
    } else {
        resp = "Falha no login. Verifique suas credenciais.";
        response.status(401); // 401 Unauthorized

    }

    makeForm(1);
    return form.replaceFirst("<input type=\"hidden\" id=\"msg\" name=\"msg\" value=\"\">", "<input type=\"hidden\" id=\"msg\" name=\"msg\" value=\"" + res
}
```

```java
private String criarTokenJWT(User user) {
    // Defina as informações do token
    String subject = String.valueOf(user.getID()); // Use o ID do usuário como assunto do token

    // Gere uma chave HMAC-SHA256 segura
    SecretKey secretKey = Keys.secretKeyFor(SignatureAlgorithm.HS256);

    // Gere o token JWT usando a chave HMAC-SHA256 segura
    String token = Jwts.builder()
        .setSubject(subject)
        .signWith(secretKey, SignatureAlgorithm.HS256)
        .compact();

    return token;
}
```

```xml
    <modelVersion>4.0.0</modelVersion>
    <groupId>ti2cc</groupId>
    <artifactId>SpeedFanatic</artifactId>
    <version>2.0</version>
    <dependencies>
        <dependency>
            <groupId>com.sparkjava</groupId>
            <artifactId>spark-core</artifactId>
            <version>2.9.3</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>1.7.21</version>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.3.3</version>
        </dependency>
        <dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.11.5</version>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.11.5</version>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-jackson</artifactId>
    <version>0.11.5</version>
</dependency>
<dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.5.13</version>
    </dependency>
```

UM POUCO DOS NOSSOS CODIGOS

```
--
-- Name: user_user_id_seq; Type: SEQUENCE; Schema: public;
--

CREATE SEQUENCE public.user_user_id_seq
    START WITH 2
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;



--
-- Name: user; Type: TABLE; Schema: public;
--

CREATE TABLE public."user" (
    user_id integer DEFAULT nextval('public.user_user_id_seq'::regclass) NOT NULL,
    user_username character varying(16) NOT NULL,
    user_email character varying(255),
    user_senha character varying(32) NOT NULL,
    user_data_criacao timestamp with time zone DEFAULT CURRENT_TIMESTAMP,
    user_piloto integer NOT NULL,
    user_equipe integer NOT NULL,
    user_nota integer
);
```

```
public Object insert(Request request, Response response) {
    String conteudo = request.queryParams("conteudo");
    int post = Integer.parseInt(request.queryParams("post"));
    String usuario = request.session().attribute("user username");
    int resppai = Integer.parseInt(request.queryParams("resp_pai"));
    LocalDate data = LocalDate.now();
    int categoria = Integer.parseInt(request.queryParams("categoria"));
    String resp = "";
    int tmp = request.session().attribute("user_nota");
    if(tmp > 0)
    {
    boolean textoAprovado = verificarTextoModerador(conteudo);
    Resposta resposta = new Resposta(1, conteudo, post, usuario, resppai, data, categoria);
    // Se o texto for aprovado, insere a resposta no serviço
    if (textoAprovado) {

        if (respostaDAO.insert(resposta) == true) {
            resp = "Resposta (" + conteudo + ") inserida!";
            response.status(201); // 201 Created
            response.redirect("/post/"+post);
    } else {
         resp = "Resposta (" + conteudo + ") não inserida!";
         response.status(404); // 404 Not found
         response.redirect("/forum");
    }
    }
    else
    {
        System.out.println("Texto nao aprovado");

    User user = userDAO.get(request.session().attribute("user_id"));
    System.out.println("NOTA ANTES: " +user.getNota());
    int temp = user.getNota()-1;
    user.setNota(temp);
    System.out.println("NOTA DEPOIS: " + user.getNota());
    userDAO.update(user);
    response.redirect("/forum");
    }
    }
```

```
  <!-- https://mvnrepository.com/artifact/com.microsoft.azure.cognitiveservices/azure-cognitiveservices-contentmoderator -->
<dependency>
    <groupId>com.microsoft.azure.cognitiveservices</groupId>
    <artifactId>azure-cognitiveservices-contentmoderator</artifactId>
    <version>1.0.2-beta</version>
</dependency>
```

UM POUCO DOS NOSSOS CODIGOS

```java
private static String criptografarSenhaMD5(String senha) {
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(senha.getBytes());
        byte[] digest = md.digest();

        // Converte o hash MD5 para uma representação hexadecimal
        StringBuilder hexString = new StringBuilder();
        for (byte b : digest) {
            hexString.append(String.format("%02x", b & 0xff));
        }

        return hexString.toString();
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException("Erro ao criar hash MD5", e);
    }
}
```

```java
private static boolean usuarioEstaAutenticado(Request request) {


    Object usuarioAutenticado = request.session().attribute("usuarioAutenticado");
    System.out.println(usuarioAutenticado);

    return usuarioAutenticado != null;
}
```

URL’s DO PROJETO

- BANCO DE DADOS EM NUVEM: http://polanski-ti-ex04.postgres.database.azure.com/
- API CONTENT MODERATOR AZURE: https://moderadorforum.cognitiveservices.azure.com/
- CODIGO GIT: https://github.com/ICEI-PUC-Minas-CC-TI/plmg-cc-2023-2-ti2-g26-speedfanatic.git

VÍDEO DO PROJETO