CONTEXTUALIZAÇÃO

Buscamos auxiliar pessoas com suas plantas domésticas.

Nosso projeto almeja recomendar produtos necessários para o cuidado das plantas, assim como ajudar a organizar as tarefas de cuidados.

27/10/2023

Nome do projeto:
GreenLeaf

Apresentado por:

- Diego Marchioni
- Luiza Dias
- Gabriel Martins
- Henrique Augusto
- Daniel Valadares

Script Sql

```
BEGIN;


SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;


-- ____________________________________________________
--|                                                    |
--|                   Table Usuario                    |
--|                                                    |
--|____________________________________________________|


CREATE SEQUENCE public."userId-usuario"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public."userId-usuario" OWNER TO ti2cc;

SET default_tablespace = '';

SET default_table_access_method = heap;



CREATE TABLE IF NOT EXISTS public.usuario
(
    userId integer NOT NULL DEFAULT nextval('public."userId-usuario"'::regclass),
    email text NOT NULL,
    login text NOT NULL,
    senha text NOT NULL,
    salt text NOT NULL
);

ALTER TABLE public.usuario OWNER TO ti2cc;
```

```
-- ____________________________________________________
--|                                                    |
--|                   Table Planta                     |
--|                                                    |
--|____________________________________________________|



CREATE TABLE IF NOT EXISTS public.planta
(
    slug text NOT NULL,
    nomecien text,
    imagemurl text,
    cresc_forma text,
    cresc_taxa text,
    ph_max real,
    ph_min real,
    luz_solar integer,
    solo_nutrientes integer,
    solo_salinidade integer,
    solo_textura integer,
    solo_umidade integer,
    nome_pop text
);

ALTER TABLE public.planta OWNER TO ti2cc;

-- ____________________________________________________
--|                                                    |
--|                   Table Produto                    |
--|                                                    |
--|____________________________________________________|


CREATE SEQUENCE public."prodId-produto"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public."prodId-produto" OWNER TO ti2cc;

SET default_tablespace = '';

SET default_table_access_method = heap;

CREATE TABLE IF NOT EXISTS public.produto
(
    prodId integer NOT NULL DEFAULT nextval('public."prodId-produto"'::regclass),
    tipo text,
    nome text,
    funcao text,
    preco real,
```

Script Sql

```
-- ____________________________________________________________
--|                                                            |
--|                     Table Agenda                           |
--|                                                            |
--|____________________________________________________________|


CREATE TABLE IF NOT EXISTS public.agenda
(
    planta text NOT NULL,
    usuario integer NOT NULL,
    nome text,
    descricao text,
    rega text,
    poda text,
    exposicao text
);

ALTER TABLE public.agenda OWNER TO ti2cc;

-- ____________________________________________________________
--|                                                            |
--|                   Table Recomendado                        |
--|                                                            |
--|____________________________________________________________|

CREATE TABLE IF NOT EXISTS public.recomendado
(
    planta text NOT NULL,
    produto integer NOT NULL,
    frequencia text
);

ALTER TABLE public.recomendado OWNER TO ti2cc;

-- ____________________________________________________________
--|                                                            |
--|                   Table Adicionado                         |
--|                                                            |
--|____________________________________________________________|

CREATE TABLE IF NOT EXISTS public.adicionado
(
    usuario integer NOT NULL,
    produto integer NOT NULL,
    planta text NOT NULL
);

ALTER TABLE public.adicionado OWNER TO ti2cc;
```

```
-- ____________________________________________________________
--|                                                            |
--|                     Table Empresa                          |
--|                                                            |
--|____________________________________________________________|


CREATE SEQUENCE public."companyId-empresa"
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public."companyId-empresa" OWNER TO ti2cc;

SET default_tablespace = '';

SET default_table_access_method = heap;



CREATE TABLE IF NOT EXISTS public.empresa
(
    companyId integer NOT NULL DEFAULT nextval('public."companyId-empresa"'::regclass),
    email text NOT NULL,
    login text NOT NULL,
    senha text NOT NULL,
    salt text NOT NULL
);

ALTER TABLE public.empresa OWNER TO ti2cc;
```

Script Sql - Keys

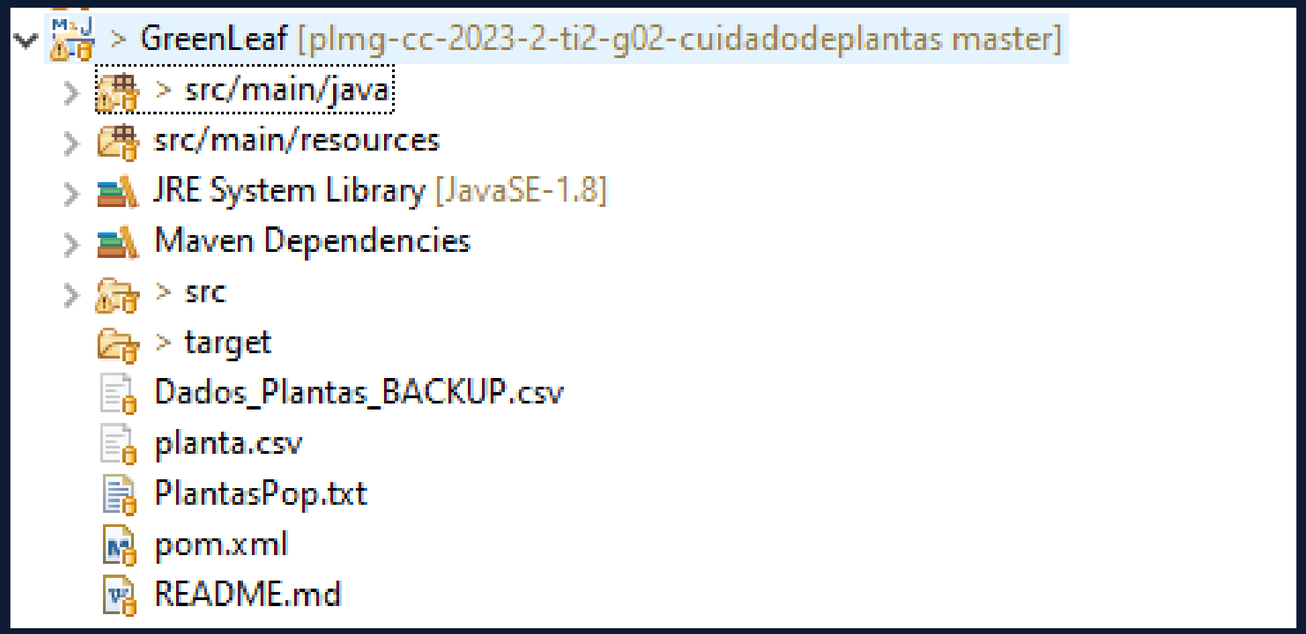
```
-- ____________________________________________________
--|                                                    |
--|                   Primary Keys                     |
--|                                                    |
--|____________________________________________________|
```

```
ALTER TABLE ONLY public.produto
    ADD CONSTRAINT produto_pkey PRIMARY KEY (prodId);

ALTER TABLE ONLY public.usuario
    ADD CONSTRAINT usuario_pkey PRIMARY KEY (userId);

ALTER TABLE ONLY public.planta
    ADD CONSTRAINT planta_pkey PRIMARY KEY (slug);

ALTER TABLE ONLY public.agenda
    ADD CONSTRAINT agenda_pkey PRIMARY KEY (planta,usuario);

ALTER TABLE ONLY public.recomendado
    ADD CONSTRAINT recomendado_pkey PRIMARY KEY (planta,produto);

ALTER TABLE ONLY public.adicionado
    ADD CONSTRAINT adicionado_pkey PRIMARY KEY (planta,produto,usuario);

ALTER TABLE ONLY public.empresa
    ADD CONSTRAINT empresa_pkey PRIMARY KEY (companyId);
```

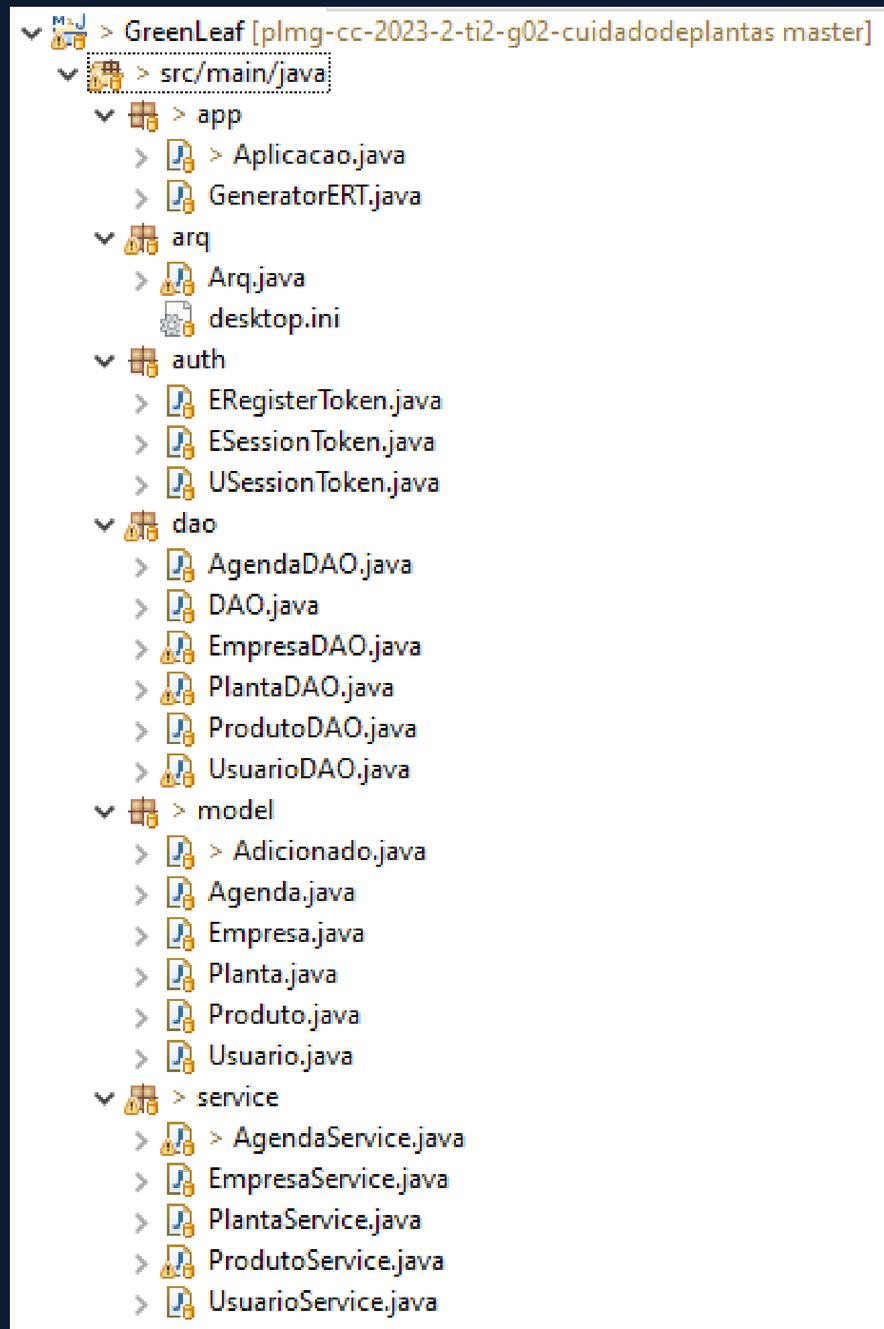
```
-- ____________________________________________________
--|                                                    |
--|                   Foreign Keys                     |
--|                                                    |
--|____________________________________________________|
```

```
ALTER TABLE IF EXISTS public.produto
    ADD FOREIGN KEY (empresa)
    REFERENCES public.empresa (companyId) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;


ALTER TABLE IF EXISTS public.agenda
    ADD FOREIGN KEY (planta)
    REFERENCES public.planta (slug) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;


ALTER TABLE IF EXISTS public.agenda
    ADD FOREIGN KEY (usuario)
    REFERENCES public.usuario (userId) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;


ALTER TABLE IF EXISTS public.recomendado
    ADD FOREIGN KEY (planta)
    REFERENCES public.planta (slug) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;


ALTER TABLE IF EXISTS public.recomendado
    ADD FOREIGN KEY (produto)
    REFERENCES public.produto (prodId) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;


ALTER TABLE IF EXISTS public.adicionado
    ADD FOREIGN KEY (produto)
    REFERENCES public.produto (prodId) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;


ALTER TABLE IF EXISTS public.adicionado
    ADD FOREIGN KEY (usuario)
    REFERENCES public.usuario (userId) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;


ALTER TABLE IF EXISTS public.adicionado
    ADD FOREIGN KEY (planta)
    REFERENCES public.planta (slug) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE CASCADE
    NOT VALID;
```

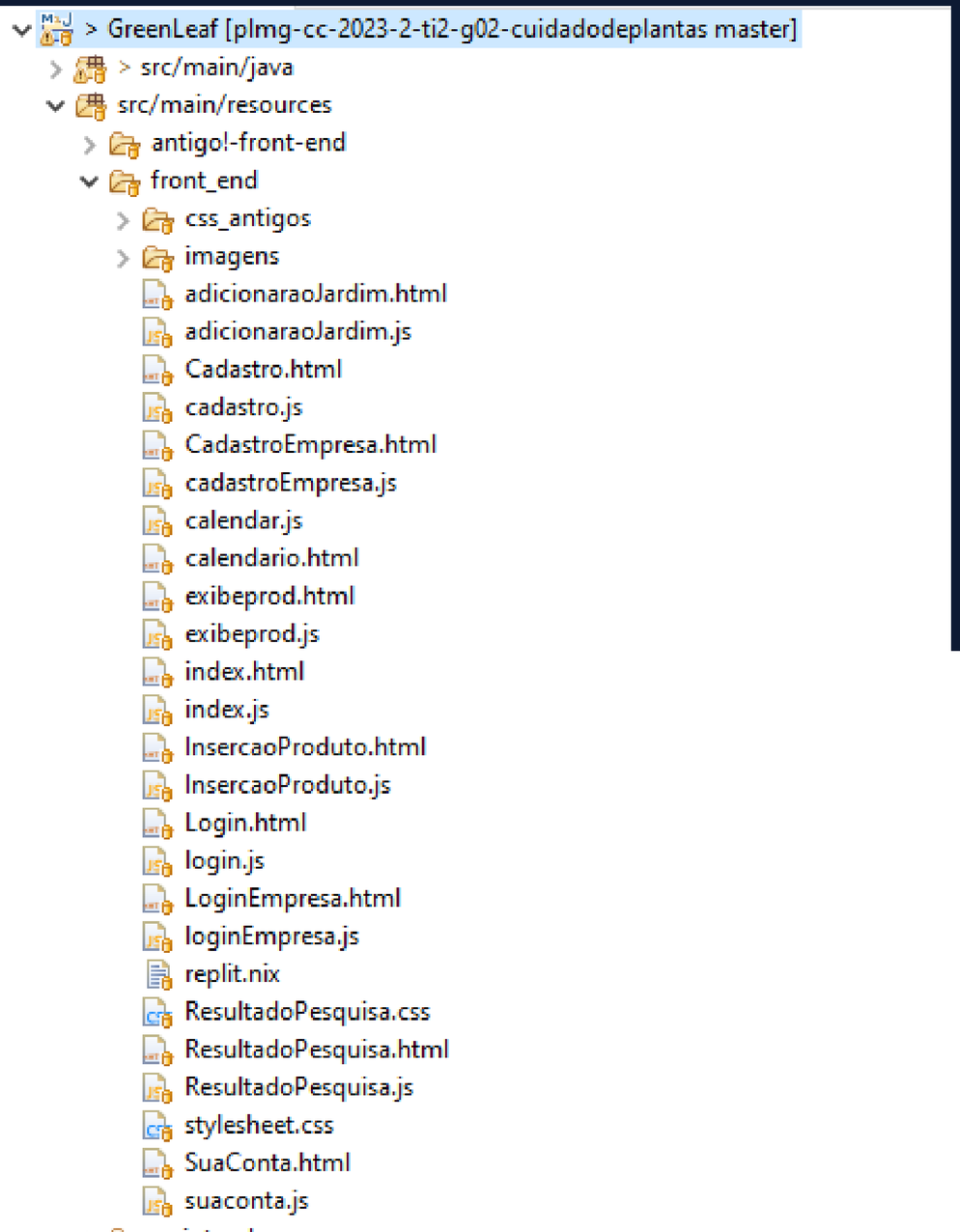
Organização Eclipse

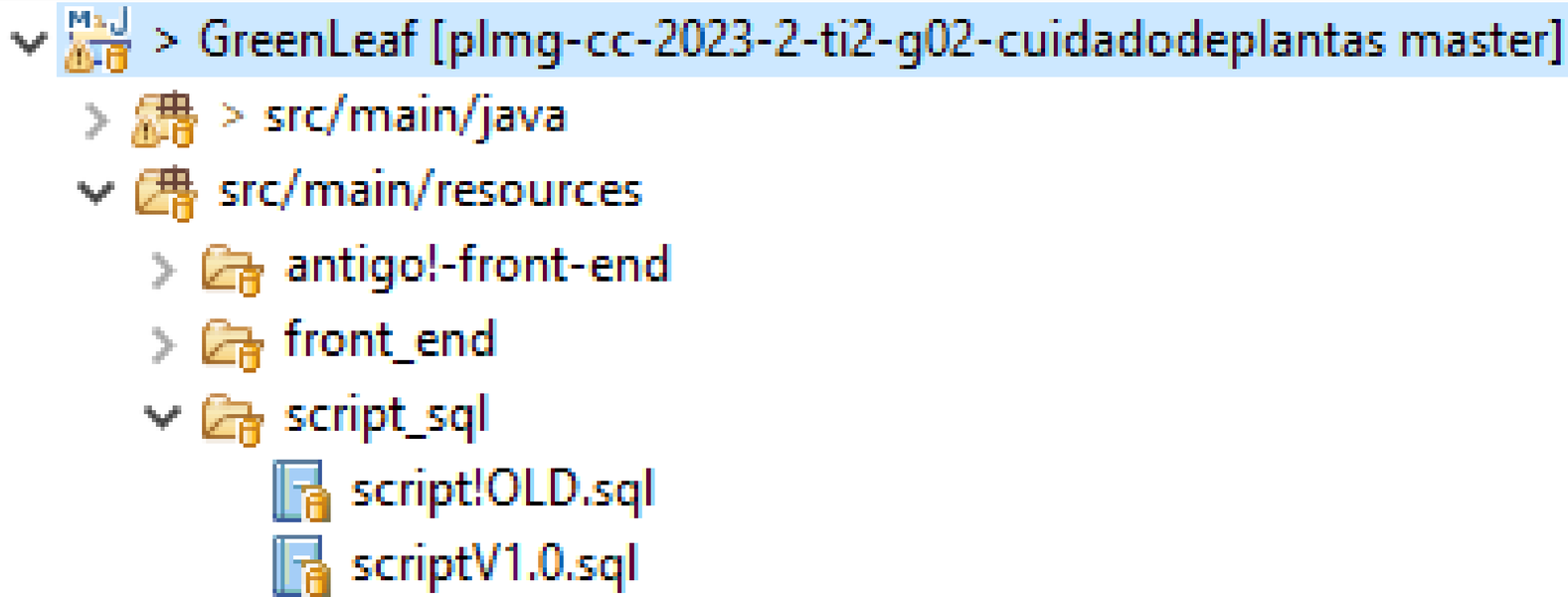


Pasta java



Pasta resources





Aplicacao.java

```java
package app;

import static spark.Spark.*;

public class Aplicacao {

    //inicizaliza plantaservice
    private static PlantaService plantService = new PlantaService();

    //inicializa objeto usuarioservice
    private static UsuarioService userService = new UsuarioService();

    //inicializa empresaservice
    private static EmpresaService empresaService =  new EmpresaService();

    //inicializa produtoService
    private static ProdutoService produtoService = new ProdutoService();

    //inicializa agendaService
    private static AgendaService agendaService = new AgendaService();


    //main
    public static void main(String[] args) {
        //escolhe a porta
        port(6789);

        //opcoes do CORS, essencial para fetch requests! essas opcoes definem o que pode ou nao ser enviado numa
        //requisicao
        options("/*", (request,response)->{

            String accessControlRequestHeaders = request.headers("Access-Control-Request-Headers");
            if (accessControlRequestHeaders != null) {
                response.header("Access-Control-Allow-Headers", accessControlRequestHeaders);
            }

            String accessControlRequestMethod = request.headers("Access-Control-Request-Method");
            if(accessControlRequestMethod != null){
            response.header("Access-Control-Allow-Methods", accessControlRequestMethod);
            }

            return "OK";
        });

        //Parecido com o options, define a url que possui permissão para enviar requisições, tal como
        //quais tipos de requisições esta pode enviar e quais headers ela pode ver
        before((request, response) -> {
            response.header("Access-Control-Allow-Origin", "http://127.0.0.1:5500");
            response.header("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE");
            response.header("Access-Control-Allow-Headers", "Content-Type, Authorization");
            response.header("Access-Control-Expose-Headers", "Authorization");
        });
```

```
//define a rota da funcao  LoginTOken
//FUNÇÃO: Validar a autenticidade do token do usuario
get("/usuario/token", (request,response) -> userService.LoginToken(request, response) );

//define a rota da funcao  SenhaToken
//FUNÇÃO: Validar a senha do usuario por meio do token de login
get("/usuario/token/:senha", (request,response) -> userService.SenhaToken(request, response) );

//define a rota da funcao  Insert
//FUNÇÃO: Realiza cadastro de usuario
post("/usuario/insert", (request,response) -> userService.Insert(request, response) );

//define a rota da funcao  Login
//FUNÇÃO: Realiza o login do usuario, assim como gera e envia o token de autenticacao
get("/usuario/login", (request,response) -> userService.Login(request, response));

//define a rota da funcao  Atualizar
//FUNÇÃO: Atualiza as informacoes do usuario no banco de dados
put("/usuario/update", (request,response) -> userService.Atualizar(request, response));


//define a rota da funcao Deletar
//FUNÇÃO: Deleta as informacoes do usuario no banco de dados
delete("/usuario/delete", (request,response) -> userService.Deletar(request, response));
```

Aplicacao.java

```java
        //define a rota da funcao  LoginTOken
        //FUNÇÃO: Validar a autenticidade do token da empresa
        get("/empresa/token", (request,response) -> empresaService.LoginToken(request, response) );

        //define a rota da funcao  EmpresaToken
        //FUNÇÃO: Validar a senha da empresa por meio do token de login
        get("/empresa/token/:senha", (request,response) -> empresaService.SenhaToken(request, response) );

        //define a rota da funcao  Insert
        //FUNÇÃO: Realiza cadastro da empresa
        post("/empresa/insert", (request,response) -> empresaService.Insert(request, response) );

        //define a rota da funcao  Login
        //FUNÇÃO: Realiza o login da empresa, assim como gera e envia o token de autenticacao
        get("/empresa/login", (request,response) -> empresaService.Login(request, response));

        //define a rota da funcao  Atualizar
        //FUNÇÃO: Atualiza as informacoes da empresa no banco de dados
        put("/empresa/update", (request,response) -> empresaService.Atualizar(request, response));

        //define a rota da funcao Deletar
        //FUNÇÃO: Deleta as informacoes da empresa no banco de dados
        delete("/empresa/delete", (request,response) -> empresaService.Deletar(request, response));

        //define a rota da funcao  Insert
        //FUNÇÃO: Realiza cadastro de produtos
        post("/produto/insert", (request, response) -> produtoService.insert(request, response, empresaService));

        //define a rota da funcao  Get
        //FUNÇÃO: Realiza get de produtos
        get("/produto/get", (request, response) -> produtoService.get(request, response, empresaService));

        //define a rota da funcao Delete
        //FUNÇÃO: Realiza delete de produtos
        delete("/produto/delete/:prodid", (request, response) -> produtoService.delete(request, response));

        //Planta GETS
        get("/planta/get", (request,response) -> plantService.getAll(request, response) );

        //get search name
        get("/planta/get/:name", (request,response) -> plantService.searchName(request, response) );

        //Agenda insert
        post("/agenda/insert/:slug", (request,response) -> agendaService.Inserir(request, response, userService));

        //Pega Calendario
        get("/agenda/calendario", (request,response) -> agendaService.getCalendario(request, response, userService, plantService));

        //deleteCalendario
        delete("/agenda/delete/:planta", (request, response) -> agendaService.delete(request, response, userService));

    }

}
```

EmpresaDAO.java

```java
//funcao deleta empresa
public boolean delete(int id) {

    boolean status = false;
    try {
        //string sql contra injecao sql
        String sql = "DELETE FROM empresa WHERE companyId = ?";
        PreparedStatement st = conexao.prepareStatement(sql);

        st.setInt(1, id);
        st.executeUpdate();
        st.close();
        status = true;
    } catch (SQLException u) {
        throw new RuntimeException(u);
    }
    return status;
}


public boolean insert(Empresa empresa) {
    boolean status = false;
    try {
        String sql = "INSERT INTO empresa (email,login,senha, salt) "
                   + "VALUES (?,?,?,?)";
        PreparedStatement st = conexao.prepareStatement(sql);
        //metodo de criptografia igual ao da funcao update
        String salt = BCrypt.gensalt();
        String senhaWSalt = salt + empresa.getSenha();
        String senha = BCrypt.hashpw(senhaWSalt, salt);
        st.setString(1, empresa.getEmail() );
        st.setString(2, empresa.getLogin() );
        st.setString(3, senha );
        st.setString(4, salt);
        st.executeUpdate();
        st.close();
        status = true;
    } catch (SQLException u) {
        throw new RuntimeException(u);
    }

    return status;
}
```

```java
//autorizacao da empresa
 public Empresa Auth(Empresa empresa){
     Empresa result = null;
     //tenta encontrar uma empresa considerando que a string salva no atributo login do objeto recebido
     //por parametro realmente e um login
     Empresa found = getByLogin(empresa.getLogin());
     //caso found seja diferente de null, era login
     if(found != null){

         //senha salva e criptografada no banco de dados
         String pw = found.getSenha();
         //somando o salt a senha digitada pela empresa
         String pwcheck = found.getSalt() + empresa.getSenha();

         //checa se a string pwcheck e igual a string criptografada pw
         if(BCrypt.checkpw(pwcheck,pw)){
             //caso seja, result a encontrada
             result = found;
         }
     } else {
         //tenta encontrar uma empresa considerando que a string salva no atributo login do objeto recebido
         //por parametro e, na verdade, um email
         found = getByEmail(empresa.getLogin());
         if(found != null) {
             //mesma coisa que a do login
             String pw = found.getSenha();
             String pwcheck = found.getSalt()+ empresa.getSenha();
             if(BCrypt.checkpw(pwcheck,pw)){
                 result = found;
             }
         }
     }

     //se a autenticacao falhar, vai retornar null
     //se a autenticacao der certo, vai retornar um objeto da classe Empresa com as informacoes da empresa
     return result;
 }
}
```

TOKENS

```java
package auth;
import com.auth0.jwt.*;


public class ERegisterToken {

    //segredo do token (o algoritmo de codificacao se baseia nessa linha para codificar)
    private static final String secret = "t^yC/'5m*nSr}BkFyq5qH|}T@UY}]#b7g@xIZ(b<Oe2z9p/E)quv,]St*]4\"6+1";

    //tempo de expiracao do token (7 dias)
    private static final long EXPIRATION_TIME_MS = 604800000;

    //gerador de tokens
    public static String TokenGenerator(String CompanyName) {
        //data atual (de criacao)
        Date Criacao = new Date();
        //data de validade (data de criacao + tempo de expiracao)
        Date Validade = new Date(Criacao.getTime() + EXPIRATION_TIME_MS);

        //cria o algoritmo de codificacao em HMAC256, baseado na string segredo
        Algorithm assinatura = Algorithm.HMAC256(secret);

        //cria o token JWT com base no nome da empresa, data de validade/criacao e assinatura (algoritmo de codificacao)

        return JWT.create()
                .withSubject(CompanyName)
                .withIssuedAt(Criacao)
                .withExpiresAt(Validade)
                .sign(assinatura);
        //ao decodificar um token JWT, vc pode pegar o subject dele (no caso, o nome da empresa)

        //Os nomes serao unicos para cada empresa, isso que faz o token ser de uso unico.

        //a partir do momento que uma empresa tiver o nome do token, esse token nao podera ser mais usado
        //visto que daria erro de conflito (o backend devera checar se ja existe uma empresa com o nome do token)
    }

    //verifica validade e autenticidade do token, depois decodifica
    public static DecodedJWT verificaToken(String token) {
        //cria o algoritmo de codificacao em HMAC256, baseado na string segredo
        Algorithm assinatura = Algorithm.HMAC256(secret);

        //Construcao do verificador propriamente dito
        JWTVerifier verificador = JWT.require(assinatura).build();
        //Cria, mas nao inicializa o token decodificado (Tokens decodificados sao objetos de DecodedJWT)
        DecodedJWT result;
        //Tenta verificar o token com o verificador
        try {
            //Se tiver sucesso, o token decodificado sera salvo na variavel result criada anteriormente
            result = verificador.verify(token);
        }
        //Excecao enviada quando o token e invalido
        catch(JWTVerificationException e) {
            //A funcao ira retornar um objeto null caso o token seja invalido
            result = null;
        }

        return result;

    }

}
```

```java
//   Essa classe gera tokens de autenticacao JWT(JSON Web Token) de CURTO-PRAZO para empresas

package auth;

import com.auth0.jwt.*;

public class ESessionToken {

    //segredo do token (o algoritmo de codificacao se baseia nessa linha para codificar)
    private static final String secret = "LNpByv&`0)U`,]UtWD/$xel',1KHGAsFIU5?K(]2?di=tL)m1lXi^Um$+t;*FN(";

    //tempo de expiracao do token (1 hora)
    private static final long EXPIRATION_TIME_MS = 3600000;

    public static String TokenGenerator(String companyId) {
        Date Criacao = new Date();
        Date Validade = new Date(Criacao.getTime() + EXPIRATION_TIME_MS);

        Algorithm assinatura = Algorithm.HMAC256(secret);
        //gera o token com base no companyId (mais informacoes olhar no ERegisterToken.java)
        return JWT.create()
                .withSubject(companyId)
                .withIssuedAt(Criacao)
                .withExpiresAt(Validade)
                .sign(assinatura);
    }

    //verifica e decodifica o token (mais informacoes olhar no ERegisterToken.java)
    public static DecodedJWT verificaToken(String token) {
        Algorithm assinatura = Algorithm.HMAC256(secret);
        JWTVerifier verificador = JWT.require(assinatura).build();
        DecodedJWT result;
        try {
            result = verificador.verify(token);
        } catch(JWTVerificationException e) {
            result = null;
        }

        return result;

    }

}
```

cadastroEmpresa.js

```
//funcao para o post propriamente dito
function enviaCadastro() {
  //pega senha do input
  let senha = document.getElementById('campoSenha').value;
  //pega email do input
  let email = document.getElementById('campoEmail').value;
  //pega login do input
  let login = document.getElementById('campoNome').value;

  //checa se são validos (nao é o ideal)
  if (validEmail && validNome && validSenha) {
    //envia as informações digitadas para o backend por meio de queryparams
    fetch("http://localhost:6789/empresa/insert?senha=" + senha + "&email=" + email + "&login=" + login, {
      //metodo definido no spark
      method: "POST",
      //enviando por CORS (regula as permissões de entrada, saida, etc do backend)
      mode: "cors"
    })
      //promise
      .then(response => {
        //response.ok pega a range de codigos de status http que sinalizam sucesso (entre 200 e 299)
        if (response.ok) {
          //envia o empresa para a pagina de login para que ele possa logar (o melhor era fazer o login automatico, mas preguica)
          window.location.href = "./LoginEmpresa.html"
        }
        //erro 409 sinaliza conflito (ou seja, existe outra pessoa com esse login ou email)
        else if (response.status == 409) {
          //pegar o body de resposta para tratamento correto do erro
          return response.text();
        }
        //erro 500 sinaliza um erro interno, nao da parte de empresa
        else if (response.status == 500) {
          alert("ERRO no servidor");
        }
      })
```

cadastroEmpresa.js

```
    //promise para tratar erro do empresa
    .then(data => {
      //pega no body da response a parte que sinaliza se o problema é com o email, login ou ambos
      let erro = data.split(" ")[1];
      //se for com email, marcar input e label do email
      if (erro === "EMAIL") {
        labelEmail.setAttribute('style', 'color: red');
        labelEmail.innerHTML = 'Email já em uso';
        document.getElementById('campoEmail').style.boxShadow = "0px 3px 5px red";
        validEmail = false;
      }
      //se for com o login, marcar input e label do login
      else if (erro === "LOGIN") {
        labelNome.setAttribute('style', 'color: red');
        labelNome.innerHTML = 'Usuário já em uso';
        validNome = false;
        document.getElementById('campoNome').style.boxShadow = "0px 3px 5px red";
      }
      //caso seja com ambos, marcar ambos
      else{
        labelNome.setAttribute('style', 'color: red');
        labelNome.innerHTML = 'Usuário já em uso';
        validNome = false;
        document.getElementById('campoNome').style.boxShadow = "0px 3px 5px red";
        labelEmail.setAttribute('style', 'color: red');
        labelEmail.innerHTML = 'Email já em uso';
        document.getElementById('campoEmail').style.boxShadow = "0px 3px 5px red";
        validEmail = false;
      }
    })
  }
  //enquanto o empresa nao alterar os campos vermelhos, impedir envio de informações
  else {
    alert("Corrija os erros em vermelho por favor!");
  }
}


// Configura os botões

document.getElementById('btnCadastro').addEventListener('click', enviaCadastro);
```

loginEmpresa.js

```
//funcao simples login
function logar() {
  //pega o campo usuario
  var pegaUsuario = document.getElementById('usuario').value;

  //pega o campo senha
  var pegaSenha = document.getElementById('senha').value;

  //envia os campos para teste de autenticação no backend por meio de queryParams
  //(a funcao foi feita de forma que ambos email e nome sejam aceitos)
    fetch("http://localhost:6789/empresa/login?login=" + pegaUsuario + "&senha=" + pegaSenha)
    .then(response => {
      //response.ok pega a range de codigos de status http que sinalizam sucesso (entre 200 e 299)
      if(response.ok){
        //pega o header authorization da resposta (contem o token de autenticacao do usuario)
        const valorDoCabecalho = response.headers.get('Authorization');
        //salvar o token de autenticacao no session storage
        //(a pratica comum é enviar antes do token a string: "Bearer ", o split separa os dois pelo espaço)
        //Exemplo de token: Bearer FK@$(_)KJ@)(M)FMJR)$J@M
        sessionStorage.setItem("UserLogado", valorDoCabecalho.split(" ")[1] );
        //retorna para a pag inicial, agora autenticado
        window.location.href = "./InsercaoProduto.html"
      }
      //em caso de erro avisa que o login ou a senha estão errados
       else if (response.status == 404){
        alert("ERRO: Login/Senha incorretos");
      }
    })
```

AgendaDAO.java

```java
package dao;

import java.sql.PreparedStatement;


public class AgendaDAO extends DAO{

    public AgendaDAO() {
        super();
        conectar();


    }

    public void finalize() {
        close();
    }

    public ArrayList<Agenda> getCalendario(int usuario) throws SQLException{
        ArrayList<Agenda> result = null;
        String sql = "SELECT * FROM agenda WHERE usuario = ?";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setInt(1, usuario);
        ResultSet rs = st.executeQuery();
        if(rs.next()) {
            result = new ArrayList<Agenda>();
            Agenda p = new Agenda(rs.getInt("usuario"),rs.getString("planta"),rs.getString("nome"), rs.getString("rega")
                    ,rs.getString("poda"), rs.getString("exposicao"), rs.getString("descricao"));
            result.add(p);
            while(rs.next()) {
                p = new Agenda(rs.getInt("usuario"),rs.getString("planta"),rs.getString("nome"), rs.getString("rega")
                        ,rs.getString("poda"), rs.getString("exposicao"), rs.getString("descricao"));
                result.add(p);
            }
        }
        return result;
    }

    public boolean fitJardim(int userId, String slug) {
        boolean result = true;
        try {

            String sql = "SELECT * FROM agenda WHERE planta = ? and usuario = ?";
            PreparedStatement st = conexao.prepareStatement(sql);
            st.setString(1, slug);
            st.setInt(2, userId);
            ResultSet rs = st.executeQuery();

            if(rs.next()) {
                result = false;
            }
            st.close();
        }catch (Exception e) {
            System.err.println(e.getMessage());
        }
        return result;
    }
```

```java
public boolean nomeAvaliable(String nome) {
    boolean result = true;
    try {

        String sql = "SELECT * FROM agenda WHERE nome = ?";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setString(1, nome);
        ResultSet rs = st.executeQuery();

        if(rs.next()) {
            result = false;
        }
        st.close();
    }catch (Exception e) {
        System.err.println(e.getMessage());
    }
    return result;
}

public boolean Inserir(Agenda x) throws SQLException {
    boolean status = false;
        String sql = "INSERT INTO agenda (planta,usuario,nome, descricao, exposicao, poda, rega )"
                + "VALUES (?, ? , ? ,?, ? , ? , ?)";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setString(1, x.getPlanta());
        st.setInt(2, x.getUsuario());
        st.setString(3, x.getNome());
        st.setString(4, x.getDescricao());
        st.setString(5, x.getExposicao());
        st.setString(6, x.getPoda());
        st.setString(7, x.getRega());
        st.executeUpdate();
        st.close();
        status = true;

    return status;
}
```

```java
public boolean update(Agenda x) throws SQLException {
    boolean status = false;
        String sql = "UPDATE agenda SET nome = ? , descricao = ?, exposicao = ?, poda = ? , rega = ? "
                + "WHERE planta = ? AND usuario = ?";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setString(1, x.getNome());
        st.setString(2, x.getDescricao());
        st.setString(3, x.getExposicao());
        st.setString(4, x.getPoda());
        st.setString(5, x.getRega());
        st.setString(6, x.getPlanta());
        st.setInt(7, x.getUsuario());
        st.executeUpdate();
        st.close();
        status = true;
    return status;
}

public boolean delete(String slug, int user) throws SQLException {
    boolean status = false;

        String sql = "DELETE FROM agenda "
                + "WHERE planta = ? AND usuario = ?";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setString(1,slug);
        st.setInt(2, user);
        st.executeUpdate();
        st.close();
        status = true;
    return status;
}
```

```java
package dao;

import model.Planta;


public class PlantaDAO extends DAO {
    //conecta pelo DAO
    public PlantaDAO() {
        super();

        conectar();
        if(getAll() == null) {
            sendCsv();
        }


    }

    //fecha a conexao
    public void finalize() {
        close();
    }

public void sendCsv() {

        try {
            BaseConnection a = (BaseConnection) conexao;
            new CopyManager(a)
            .copyIn(
                "COPY planta FROM STDIN (FORMAT csv, HEADER)",
                new BufferedReader(new FileReader("./planta.csv"))
                );
        }catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }
```

PlantaDAO.java

Headers do arquivo csv:

- slug
- nomecien
- imagemurl
- cresc_forma
- cresc_taxa
- ph_max,ph_min
- luz_solar
- solo_nutrientes
- solo_salinidade
- solo_textura
- solo_umidade
- nome_pop

Exemplos de rows no BD

Produtos

	prodid [PK] integer	tipo text	nome text	funcao text	preco real	produrl text	empresa integer
1	10	Matador de Pragas	Insceticida	Matar pragas	12	https://	1

Plantas

	slug [PK] text	nomecien text	imagemurl text	cresc_forma text	cresc_taxa text	ph_max real	ph_min real
1	quercus-rotundifolia	Quercus rotundifolia	https://d2seqvvyy3b8p2.cloudfront.net/40ab8e7cdddbe3e78a581b84efa4e893.jpg	null	null	-1	-1
2	urtica-dioica	Urtica dioica	https://bs.plantnet.org/image/o/9db58cbb3538a6b77384f972971d51869228e545	null	null	7	6.5
3	dactylis-glomerata	Dactylis glomerata	https://bs.plantnet.org/image/o/f84a7d4fc2e627ccd451f568479b1932c2b2d900	Bunch	Moderate	7.5	5.2
4	plantago-lanceolata	Plantago lanceolata	https://bs.plantnet.org/image/o/f8d7d6fe52e36d04f5ad1fc03f46f604d5c3cc43	null	null	6.5	5.5

Usuario

	userid [PK] integer	email text	login text	senha text	salt text
1	1	marchioni2003@hotmail.com	Diego	$2a$10$gcUNmVcdxqmJrUgY05k.SuamrePW4Y1mFGFMbemKC5S4n.3/UR0...	$2a$10$gcUNmVcdxqmJrUgY05k.Su

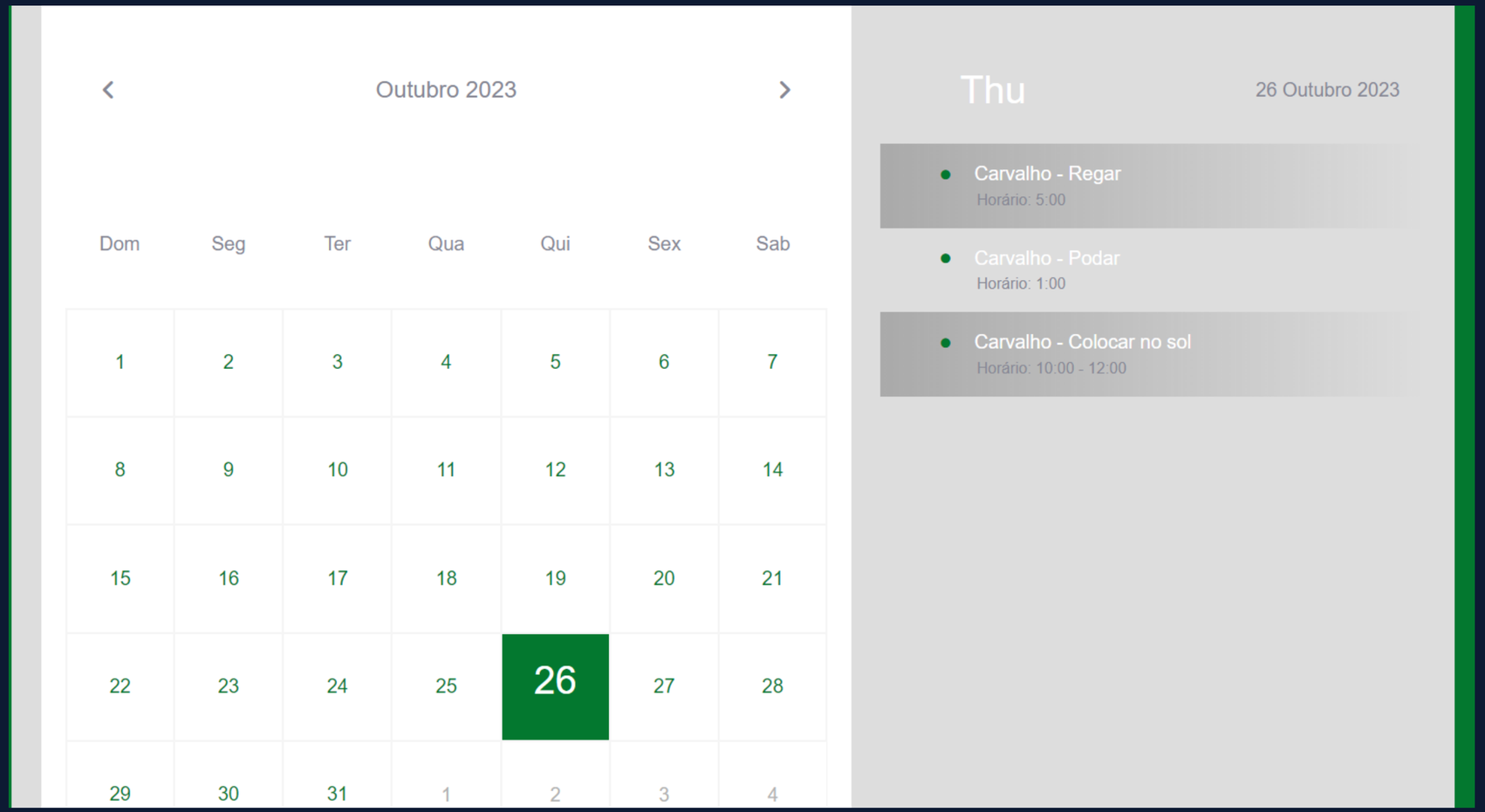
Empresa

	companyid [PK] integer	email text	login text	senha text	salt text
1	1	marchioni2003@gmail.com	Empresa	$2a$10$ZlpTT7k5OcOiblIbU7DaJuJonPr1vzOClUyZQyyYEWGV3Lx1ISWDm	$2a$10$ZlpTT7k5OcOiblIbU7DaJu

Agenda banco de dados

	planta [PK] text	usuario [PK] integer	rega text	poda text	exposicao text	nome text	descricao text
1	quercus-rotundifolia	1	5,2	1,14	10 - 12	Carvalho do Joel	Estou testando esse site para ver se consigo me organizar com relação

Agenda no frontend



Inteligência Artificial