**Semana DEVSUPERIOR 2.0 – 04/01/2021**

Vídeos: <https://devsuperior.com.br/sds2-evento>

Fórum: <https://discord.gg/4DrwAMH3Hz>

User: Geydel

Email: [geydel@hotmail.com](mailto:geydel@hotmail.com)

Pwd: 99793525

Setup Ambiente de DEV: <https://github.com/devsuperior/sds1/tree/master/ferramentas/windows>

Inf. Setup Projeto: <https://github.com/devsuperior/jdbc-postgres>

Repositório Projeto: <https://github.com/devsuperior/sds2>

**Banco Postgres**

>> Senha do usuário master - postgres: admin

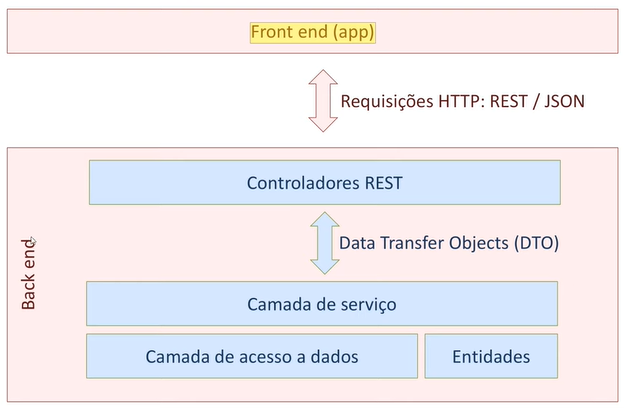
Verificar se o serviço está rodando corretamente:



>> Para abrir o Banco Postgres utilizamos o gerenciado de banco pgAdmin. Basta abrir Menu iniciar: ***pgAdmin*** ... ele vai subir um servidor local e abrir a aplicação via browser.

|  |
| --- |
| Erro: o pgAdmin não abre... fica tentando carregar no browser e não abre!  <https://stackoverflow.com/questions/64829748/pgadmin-is-not-loading>  <https://www.postgresql-archive.org/pg-Admin-4-v4-28-Errors-on-launch-td6162407.html>    Após aplicar, stop/start na máquina! |

**Back End**

**Back End x Front End**

**Controlador REST** – Responsável por receber as requisições da aplicação (Front End) e encaminhar as chamadas para os serviços responsáveis por realizar as ações do usuário.

**Rest/JSON** – Padrão de consulta aos dados no Back.

**Camada de Serviço** – Tem a lógica de negócio, cálculos, processamentos, etc..

**Camada de Acesso a Dados** – Oferece os métodos para persistências dos dados em banco.

**Data Transfer Objects** – responsável apenas pela simples transferência dos dados da camada de controle para a camada de serviço.

**Entidades** – serão instanciadas dentro da camada de serviço e apresentam um controle muito maior que o DTO.

**Rotas**

No passado as rotas e suas ações eram definidas na própria URL, por exemplo:

<https://seudominio.com/clientesSalvar> [POST]

<https://seudominio.com/clientesDeletar> [POST]

<https://seudominio.com/clientesBuscar?nome=Ana> [GET]

Com REST você precisa expressar a ação desejada com o próprio HTTP, mantendo o mesmo prefixo da rota.

<https://seudominio.com/clientes> [POST] .. Inserir - Não Idempotente (Toda vez que for executada salvará um novo registro)

<https://seudominio.com/clientes/5> [DELETE] .. Deletar - Idempotente (Pode ser executada a mesma ação, o impacto é o mesmo)

<https://seudominio.com/clientes/5> [GET] .. Consulta - Idempotente

<https://seudominio.com/clientes/5> [PUT] .. Alterar - Idempotente

**Criação do Projeto BackEnd**

O projeto foi criado através do site: [Spring Initializr](https://start.spring.io/)

**Arquivos de Configuração**

Cada ambiente possui seu arquivo de configuração, inclusive o ambiente local do analista (Teste).

|  |  |
| --- | --- |
|  | 🡪 Seta o profile de Teste para execução (Ativo)  🡪 Informa que o JPA não estará disponível na camada de Controller, apenas na camanda de serviço. Isso gera uma segurança, pois evita acesso direto ao banco na camanda Controller. |
|  | Configurações do Banco de Teste H2 que será criado dinamicamente em tempo de execução da aplicação.  Pode ser utilizado para implementação de testes integrados! |
|  | Configurações do banco de DEV Postgre. |
|  | Configurações do banco de Produção. |

Classe de Configuração de Segurança

Aclas

|  |  |
| --- | --- |
| @Configuration  @EnableWebSecurity  public class SecurityConfig extends WebSecurityConfigurerAdapter {  @Autowired  private Environment env;  @Override  protected void configure(HttpSecurity http) throws Exception {  if (Arrays.asList(env.getActiveProfiles()).contains("test")) {  http.headers().frameOptions().disable();  }    http.cors().and().csrf().disable();  http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);  http.authorizeRequests().anyRequest().permitAll();  }  @Bean  CorsConfigurationSource corsConfigurationSource() {  CorsConfiguration configuration = new CorsConfiguration().applyPermitDefaultValues();  configuration.setAllowedMethods(Arrays.asList("POST", "GET", "PUT", "DELETE", "OPTIONS"));  final UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();  source.registerCorsConfiguration("/\*\*", configuration);  return source;  }  } | Libera o acesso da aplicação ao banco H2.  **csrf().disable()** -> deabilita a proteção contra de ataque em sessão (Aplicação é REST, não guarda dados em sessão)  **SessionCreationPolicy.STATELESS** -> não guarda dados de sessão  **anyRequest().permitAll()** -> permite todas as requisições.  **Cors** -> recursos que os navegadores possuem que bloqueiam quando uma aplicação de um domínio tentam acessar outra de outro domínio. Objetivo é liberar o FrontEnd para para acessar o BackEnd. |

**Front Web**

**Sigle Page Application (SPA)**: significa você codificar menos no server-side e mais no client-side, ou seja, estará quase toda no cliente, sendo que assim que o usuário acesso o site a aplicação e seus templates são armazenados no lado cliente, evitando os refresh de tela. ([Leia](https://blog.locaweb.com.br/artigos/desenvolvimento-artigos/o-que-e-single-page-application/))

**ReactJS**: biblioteca JavaScript mais popular para construir aplicações SPA. Usado por Facebook, etc.. Ela divide sua tela em componentes, muito simples de usar. Seu código (React Native) pode ser compilado automaticamente para iOS e Android.

**Criando o Projeto:**

Para criar o projeto, ir no diretório GIT onde o será criado – no terminal de comando digitar o comando abaixo:

Npx create-react-app front-web –template typescript –use-npm (sendo que “front-web” é o nome do projeto)

Após o projeto criado, entrar na pasta “front-web” e digitar “code .”. Isso abrirá o projeto no VS Code.

**Rodando a Aplicação:** npm start

**Gerenciador de Rotas do React**: npm install react-router-dom

Também é necessário instalar os tipos da biblioteca de rotas: npm install @types/react-router-dom

as