Tecnologias Para Back-End

Prof. JUNIO FIGUEIRÊDO

JUNIOINF@GMAIL.COM

AULA 06 – CADASTRO DE PACIENTES

Cadastro de Pacientes

Spring


Novo perfil
Médicos
Pacientes
Consultas
Paciente
Nome completo
CPF
Contatos
E-mail
Telefone ou celular
Endereço
Logradouro
Número
Complemento
Cidade
UF
CEP
Concluir cadastro
Cancelar

Spring

Roteiro - Revisão

Spring Initializr: Baixando as Dependencias

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

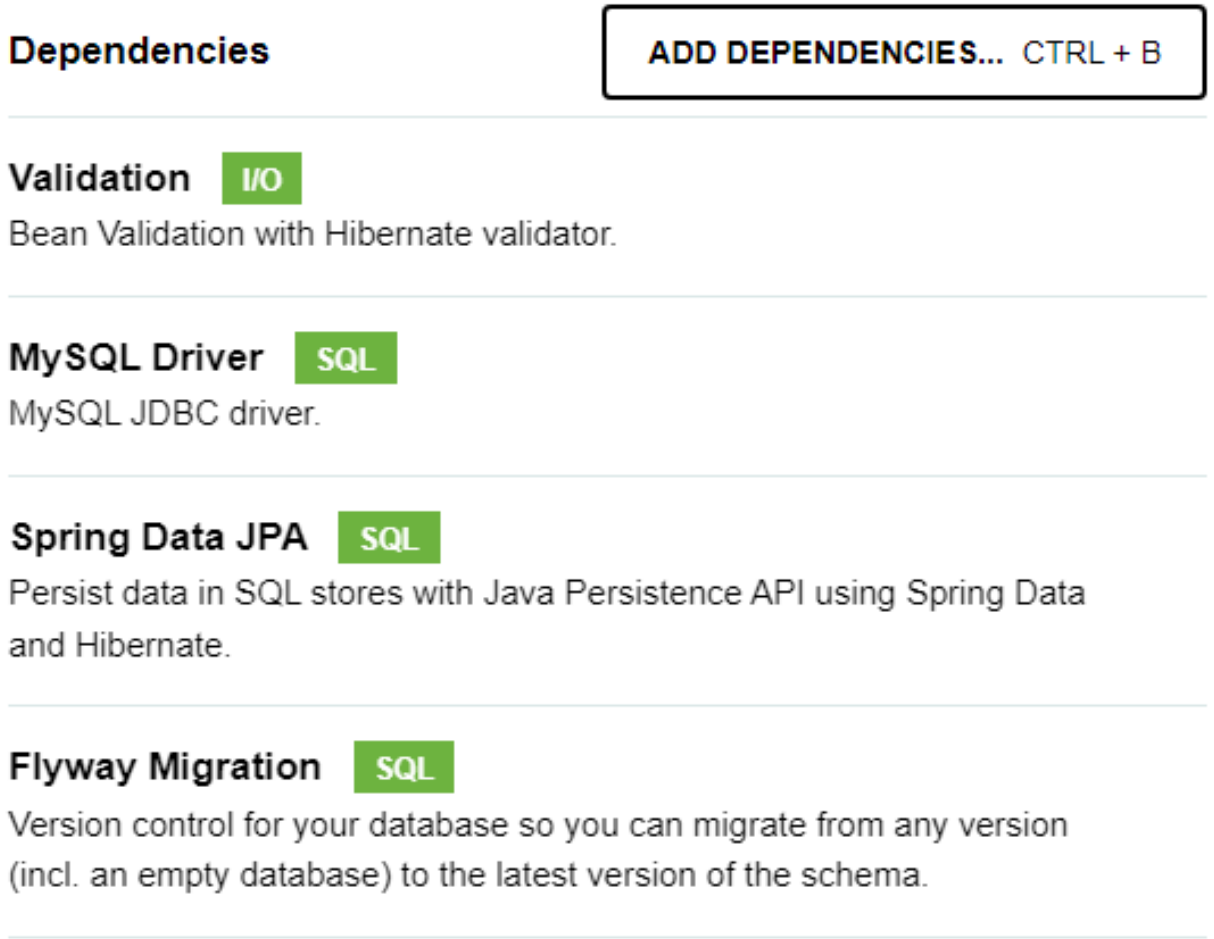
Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring

Roteiro - Revisão

Spring Initializr: Baixando as Dependencias



Spring

- **DTO** é uma classe referênte aos campos que devem aparecer na página web (ou ferramente de teste, como Insomnia).
- **JPA** é uma classe referêntes aos campos que serão mapeados como tabelas no banco de dados.

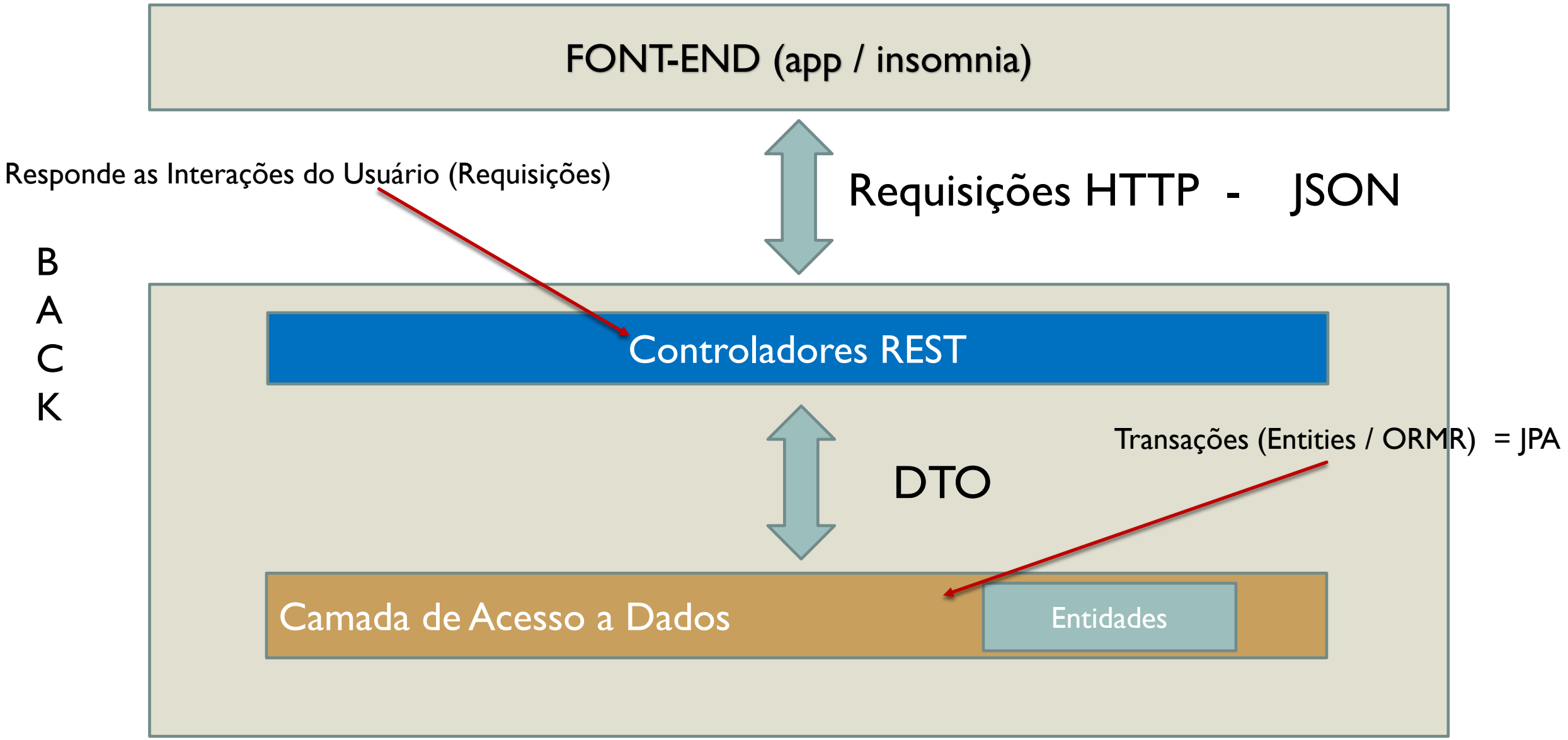
Roteiro - Revisão

Pra quê usar DTO?

- Projeção de dados
  - Segurança
  - Economia de tráfego
  - Flexibilidade: permite que a API trafegue mais de uma representação dos dados
    - Para preencher uma combobox: { id: number, nome: string }
    - Para um relatório detalhado: { id: number, nome: string, salario: number, email: string, telefones: string[ ] }

Spring

Roteiro - Revisão



Roteiro - Revisão

- Você precisará criar uma classe Controller chamada de PacienteController.
- Lembre-se de inserir as anotações:
  - **@RestController** – define uma classe que contém métodos para uma API RESTful. Isso não é uma aplicação tradicional, e sim uma API REST.
  - **@RequestMapping** – mapeia requisições REST.
  - A requisição é cadastrar

- Criar um DTO na pasta Paciente:

```
public record DadosCadastroPaciente(
        String nome,
        String email,
        String telefone,
        String cpf,
        DadosEndereco endereco
) {
```

Roteiro - Revisão

- Já o DTO DadosEndereco será o mesmo utilizado na funcionalidade de cadastro de médicos.

- Lembrar de realizar as devidas Validações.

```
public record DadosCadastroPaciente(
        @NotBlank String nome,
        @NotBlank @Email String email,
        @NotBlank String telefone,
```

Roteiro - Revisão

- Validações continuação.

- @NotBlank @Pattern(regexp = "\\d{3}\\.?\\d{3}\\.?\\d{3}\\-?\\d{2}")
  String cpf,

```
@NotNull @Valid DadosEndereco endereco
```

- Na sequência, precisa criar um repository

```
public interface PacienteRepository extends JpaRepository<Paciente, Long> {
}
```

Spring

- O Repository realiza operações individuais de acesso ao bd.
- Depois, precisará alterar as classes Controller:

```
@Autowired
private PacienteRepository repository;
```

- Estamos avisando ao Spring que esse novo atributo será instanciado por você(Spring). Isso chamamos de Injeção de dependência

- E, por fim, vai precisar criar uma migration (Atenção! Lembre-se de parar o projeto antes de criar a migration!):

```
create table pacientes(
    id int not null auto_increment,
    nome varchar(100) not null,
    email varchar(100) not null unique,
    cpf varchar(14) not null unique,
    telefone varchar(20) not null,
    logradouro varchar(100) not null,
    bairro varchar(100) not null,
    cep varchar(9) not null,
    complemento varchar(100),
    numero varchar(20),
    uf char(2) not null,
    cidade varchar(100) not null,
    primary key(id)
);
```

- O atributos de DadosCadastroPaciente são os mesmos do Entity Paciente
- Para chamar a classe Paciente de Entity, precisamos adicionar as anotações da JPA para transformar isso em uma entidade.

```
@Getter
@EqualsAndHashCode(of = "id")
@NoArgsConstructor
@AllArgsConstructor
@Entity(name = "Paciente")
@Table(name = "pacientes")
```

Spring