Curso

Aplicações JAVA com SPRING BOOT

Prof. Msc. Antonio B. C. Sampaio Jr
ENGENHEIRO DE SOFTWARE & PROFESSOR

@abctreinamentos
@amazoncodebr

www.abctreinamentos.com.br
www.amazoncode.com.br

CONTEÚDO PROGRAMÁTICO

- UNIDADE 1 – INTRODUÇÃO
- UNIDADE 2 – FUNDAMENTOS DO SPRING BOOT
- UNIDADE 3 – PERSISTÊNCIA DE DADOS NO SPRING BOOT
- UNIDADE 4 – PROJETO WEB NO SPRING BOOT
- UNIDADE 5 – PROJETO REST API NO SPRING BOOT
- UNIDADE 6 – PROJETO REST API NO SPRING BOOT COM REACTJS

CONTEÚDO PROGRAMÁTICO

- UNIDADE 7 – PROJETO REST API NO SPRING BOOT COM THYMELEAF
- UNIDADE 8 – PROJETO REST API NO SPRING BOOT COM MICROSSERVIÇOS
- UNIDADE 9 – PROJETO FINAL
- EXTRAS

PROJETOS DO CURSO

- 1º Projeto Spring Boot – Impressão de Mensagens
- 2° Projeto Spring Boot – Impressão de Mensagens na WEB
- 3° Projeto Spring Boot – Aplicação Servidor Público
- 4° Projeto Spring Boot – Aplicação Servidor Público na WEB
- 5° Projeto Spring Boot – Aplicação Servidor Público no SGBD MYSQL
- 6° Projeto Spring Boot – Aplicação Servidor Público no MONGO DB

PROJETOS DO CURSO

- 7° Projeto Spring Boot – Aplicação Servidor Público WEB
- 8° Projeto Spring Boot – Aplicação Servidor Público REST API e MySQL
- 9° Projeto Spring Boot – Aplicação Servidor Público REST API com REACT
- 10° Projeto Spring Boot – Aplicação Servidor Público/Curso REST API – Monolítico
- 11° Projeto Spring Boot – Aplicação Servidor Público REST API - Microsserviços

UNIDADE 7

PROJETO REST API NO SPRING BOOT COM THYMELEAF

REST API COM THYMELEAF

ARQUITETURA MONOLÍTICA

PROJETO PRÁTICO

10° Projeto Spring Boot – Aplicação Servidor Público/Curso REST API com THYMELEAF

REST API COM THYMELAEF

REST API COM THYMELEAF

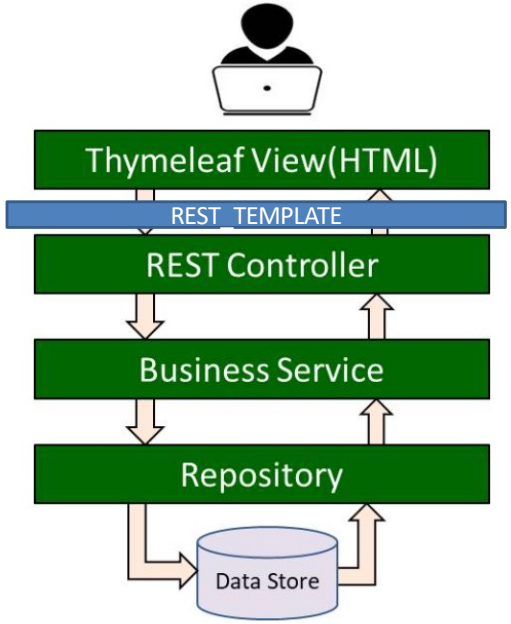
- **Incompatibilidade**
  - O Thymeleaf é uma biblioteca de renderização de templates para aplicações web, enquanto uma API REST é um estilo arquitetural para a construção de serviços web.

  - Para a criação de uma API REST no Spring Boot, faz-se necessário a utilização da anotação @RestController para criar endpoints RESTful. Contudo, essa anotação não é compatível com o Thymeleaf. Isso ocorre porque as respostas são geradas como dados serializados ao invés de HTML dinâmico.

  - Por isso, o Thymeleaf não é a escolha mais comum para renderização direta de respostas de API REST.

REST API COM THYMELEAF

- **Qual a Solução?**
  - Criar uma camada intermediária que se comunique com as páginas Thymeleaf e com os serviços da API REST, fazendo uso da classe RestTemplate.



https://www.kindsonthegenius.com/crud-tutorial-with-spring-h2-thymeleaf-bootstrap-jquery-and-mysql-step-by-step-procedure/

REST TEMPLATE

- **Definição**
  - O RestTemplate é uma classe do Spring Framework que fornece uma maneira conveniente de interagir com APIs RESTful no lado do cliente. Ele simplifica o processo de fazer solicitações HTTP, converter as respostas em objetos Java e lidar com os diferentes métodos HTTP, como GET, POST, PUT, DELETE, entre outros.

```
@Configuration
public class AppConfig {
  @Bean
  public RestTemplate restTemplate() {
     return new RestTemplate();
  }
}
```

REST TEMPLATE

- **Injeção de Dependência**
  - O RestTemplate será injetado no código que irá definir os endpoints específicos que serão manipulados pelas páginas Thymeleaf.

```
@Controller
public class AppClient {
  @Autowired
  private RestTemplate restTemplate;
  /***** API – END POINT ESPECÍFICO ******/
  @GetMapping("/listagemServidores")
  public String getServidores(Model model) {
     ResponseEntity<List<ServidorPublico>> response = restTemplate.exchange(
     "http://localhost:8080/listarServidores", HttpMethod.GET, null,
     new ParameterizedTypeReference<List<ServidorPublico>>() {});
     List<ServidorPublico> servidoresEncontrados = response.getBody();
     model.addAttribute("servidorespublicos", servidoresEncontrados);

     // Retorne a view Thymeleaf para renderização
     return "servidorpublico/servidorespublicos";
}
```

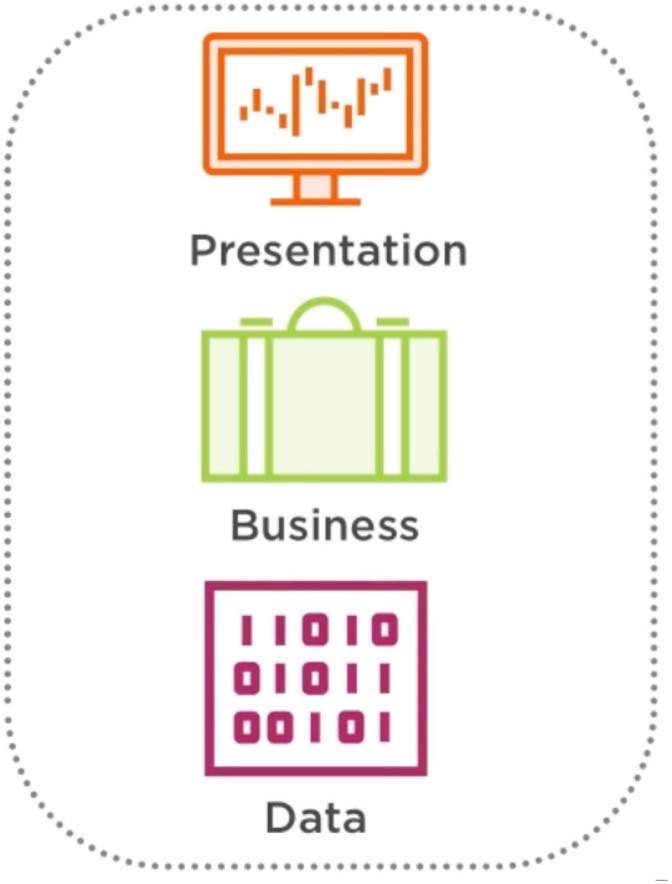
END-POINT THYMELEAF

END-POINT REST

ARQUITETURA MONOLÍTICA

ARQUITETURA MONOLÍTICA

- **Definição**
  - É um tipo de arquitetura de software em que todo o aplicativo é construído como um único componente, geralmente em um único processo ou servidor. Nesse tipo de sistema, todas as funcionalidades, camadas e componentes são empacotados juntos e executados como uma única unidade.



ARQUITETURA MONOLÍTICA

- **Principais Características**
  - Arquitetura Centralizada - Todo o código-fonte e a lógica de negócios residem em um único aplicativo, normalmente em um único repositório de código.

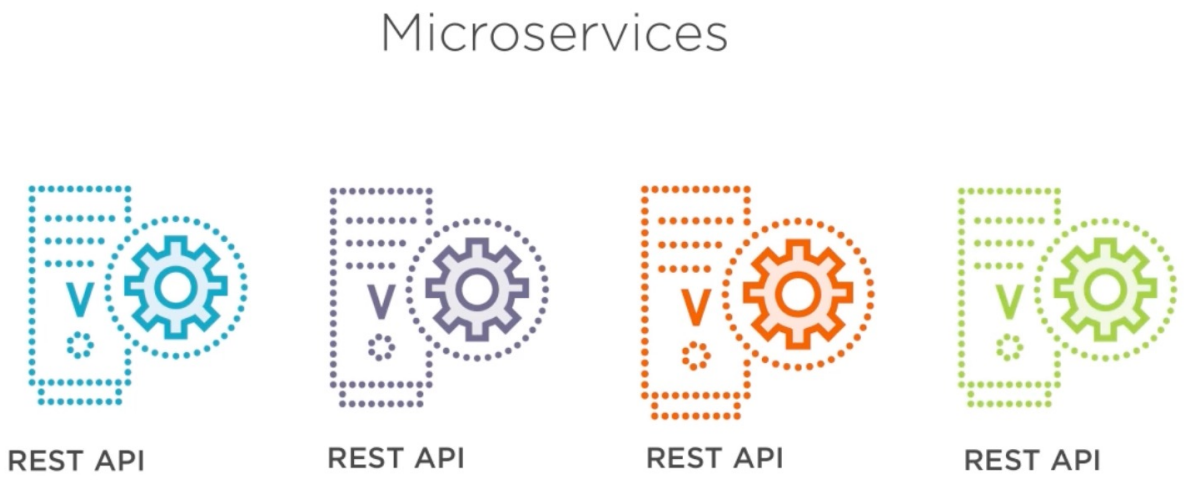
  - Implantação Única – O aplicativo é implantado como um único pacote em um único servidor ou ambiente de execução.

  - Acoplamento forte - As diferentes partes do aplicativo estão intimamente acopladas, o que significa que uma alteração em uma parte pode afetar outras partes do sistema.
  - Escalabilidade Limitada - A capacidade de escalabilidade é limitada, pois o sistema é dimensionado como um todo, em vez de componentes individuais.

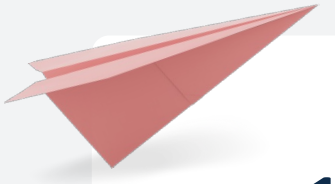
ARQUITETURA MONOLÍTICA

- **Principais Características**
  - Dificuldade de Manutenção - Devido ao acoplamento forte e à natureza monolítica, a manutenção e a atualização do sistema podem ser complexas e demoradas.

- **Alternativa**

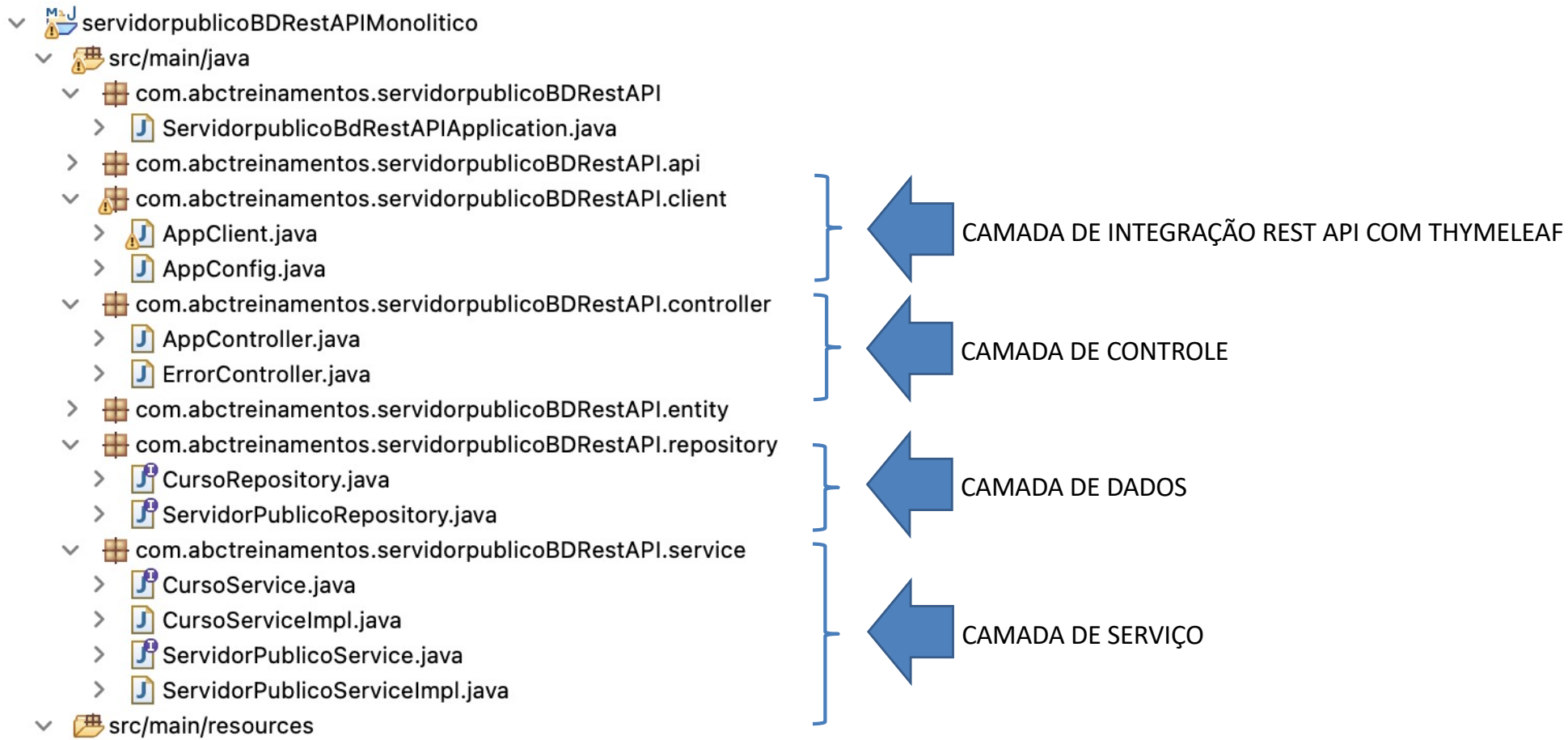


PROJETO PRÁTICO

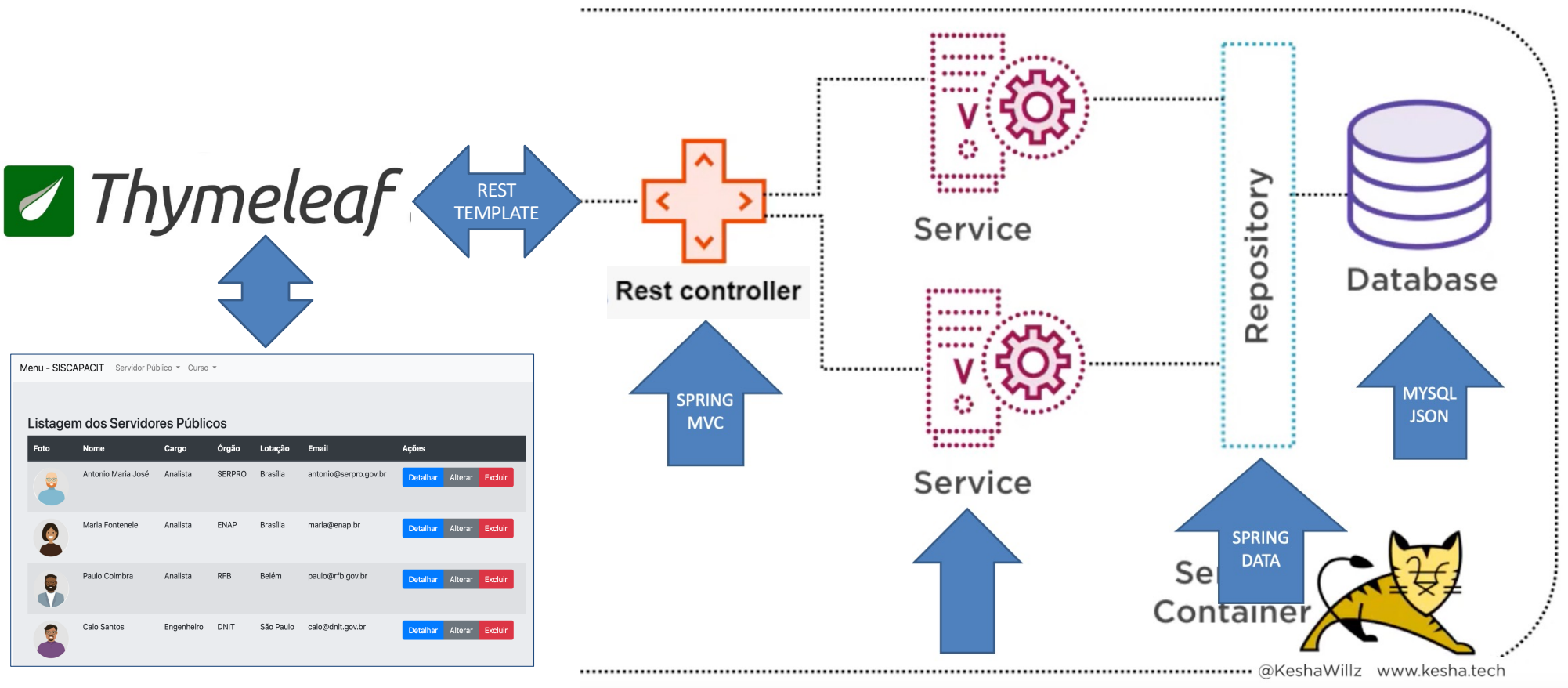


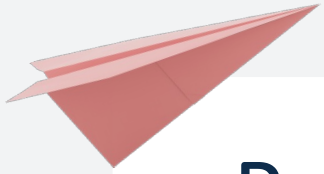
10° Projeto Spring Boot – Aplicação Servidor Público/Curso REST API com THYMELEAF

ARQUITETURA MONOLÍTICA



ARQUITETURA THYMELEAF

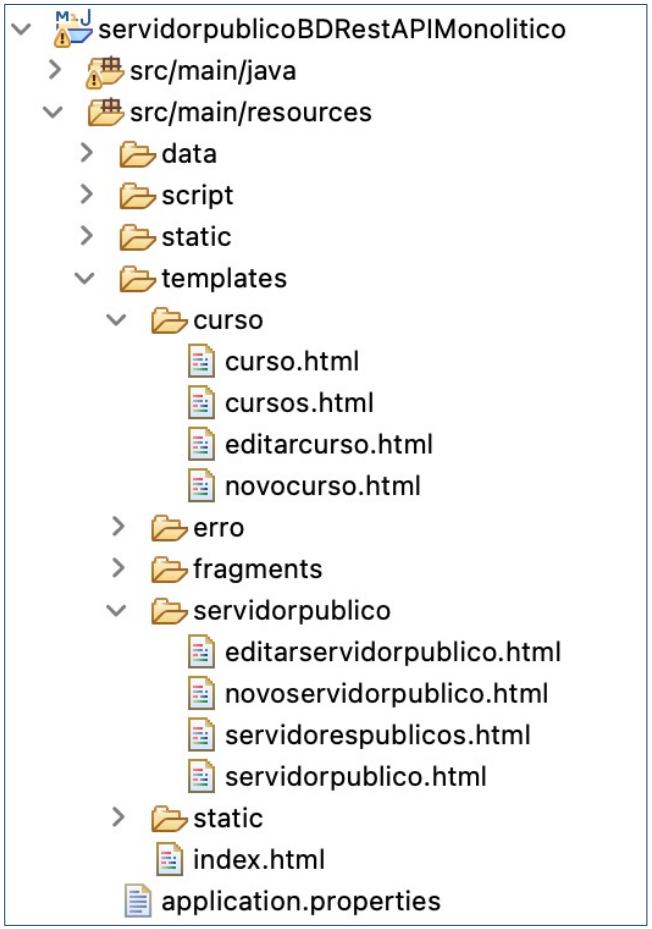




Passos:

- Copiar o projeto anterior (APIRest)
- Criar as novas classes/interfaces para representar a entidade Curso
- Criar um Cliente Thymeleaf
- Criar todas as páginas HTML de Curso

ARQUITETURA DO PROJETO - FRONTEND



ARQUITETURA DO PROJETO - BACKEND