

POO2

Encontro 03 – Conceitos de Spring Boot

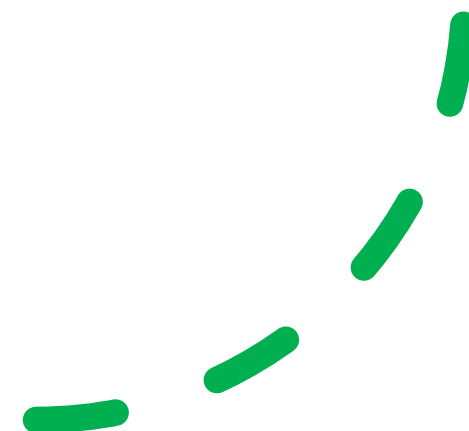
Professor Luiz Augusto Rodrigues

luiz.a.rodrigues@cogna.com.br



Sumário

- Introdução ao Spring Boot
- Arquitetura
- Configuração
- Estrutura Revisitada
- Referências



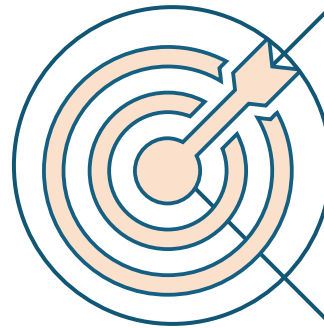
Informações Importantes



Nenhum material será enviado via e-mail. Os materiais serão disponibilizados no **AVA** e no Google Drive da Disciplina.



Dúvidas, questionamentos, entre outros deverão ser realizados pelo **e-mail** e pelo grupo de **Whatsapp** da disciplina.



A Disciplina de Programação Orientada a Objetos, a partir deste slide, será referenciada pela sigla **POO2**.

Informações Importantes

Grupo de Whatsapp da Disciplina:

<https://tinyurl.com/mw2xdkd7>



POO2

**Instruções
Importantíssimas**



Instruções Importantíssimas

Para este semestre, na disciplina de POO, utilizaremos OBRIGATORIAMENTE as seguintes ferramentas:

- Sistema Operacional Windows 10, Linux ou MacOS.
- Linguagem de Programação Java.
- JDK 17 ou superior
 - https://download.oracle.com/java/17/archive/jdk-17.0.6_windows-x64_bin.exe
- VS Code.

Instruções Importantíssimas

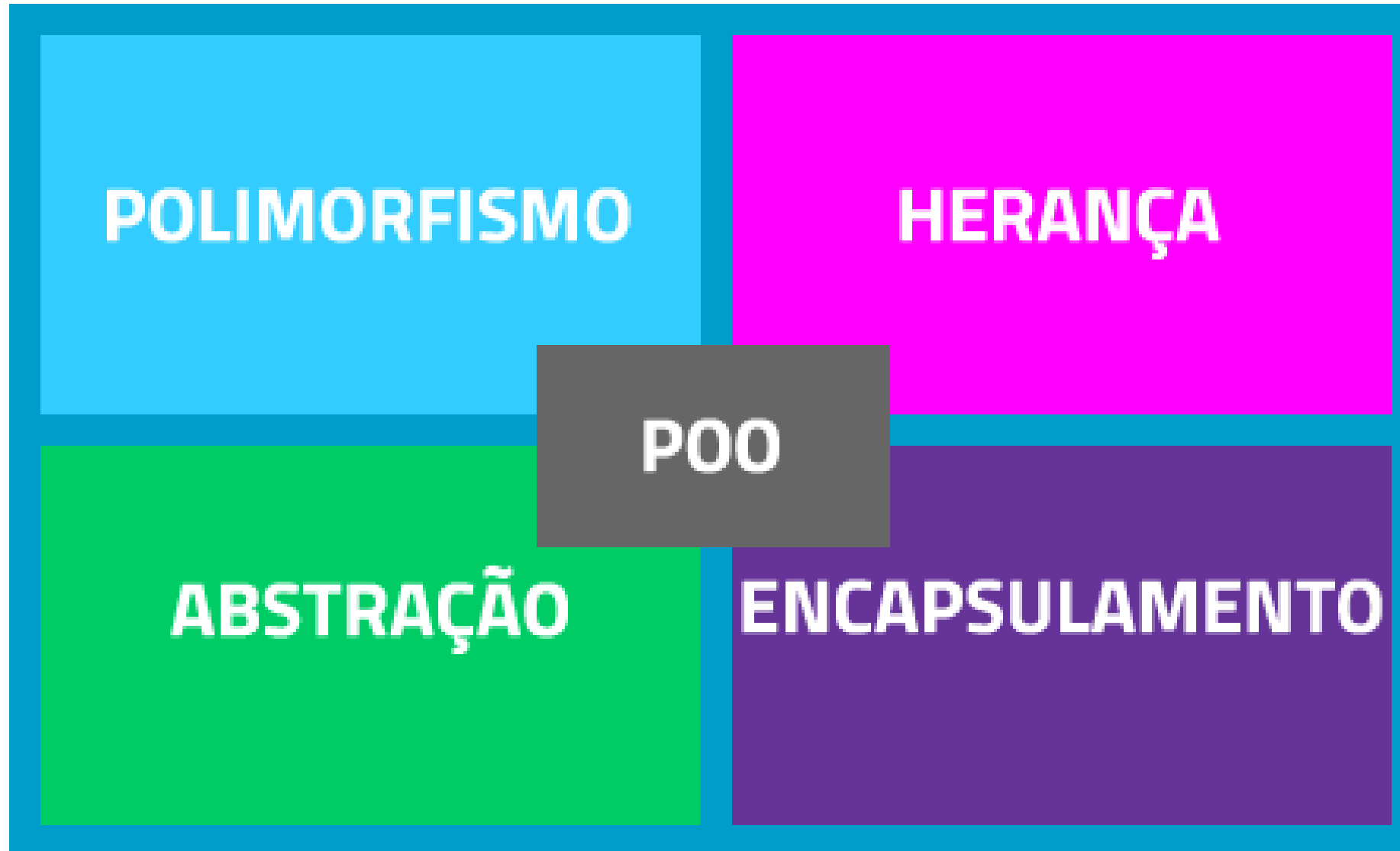
- Git 2.39.2 ou superior
 - <https://github.com/git-for-windows/git/releases/download/v2.39.2.windows.1/Git-2.39.2-64-bit.exe>
- Github
 - <https://github.com/>
 - <https://desktop.github.com/>
- Tutorial em Git e Github
 - Tutorial Git e Github 2022 – Introdução prática para iniciantes
 - Canal DevSuperior
 - https://youtu.be/_hZf1teRFNg

Quem for utilizar Linux ou MacOSx, fale comigo após a aula, ou pelo grupo do WhatsApp, para instruções de instalação.

Instruções Importantíssimas

- Os relatórios não serão utilizados nessa disciplina, por se tratar de uma disciplina 90% prática.
- No lugar, teremos Listas de Atividades Formativas (Atividade do Professor), que deverão ser respondidas e justificadas, para fixação do conteúdo discutido nas aulas.
- As atividades de aula estarão disponíveis Google Drive da turma.
- O tutorial do Git e Github é obrigatório.

Pilares de POO



POO2

Conceitos de Spring Boot

Introdução



Spring Boot é um framework open-source que faz parte do ecossistema Spring, amplamente utilizado para o desenvolvimento de aplicações Java.

Ele foi criado com o objetivo de simplificar o processo de construção de aplicações Spring, eliminando grande parte da configuração manual e da complexidade que, historicamente, estava associada ao desenvolvimento em Java.

Com o Spring Boot, você pode criar aplicações robustas e prontas para produção rapidamente, utilizando configurações pré-estabelecidas que seguem as melhores práticas da indústria.

O desenvolvimento de aplicações Java tradicionais frequentemente envolvia muita configuração manual, desde a configuração de servidores até a definição de diversos arquivos XML. Isso tornava o processo mais demorado e propenso a erros.

Spring Boot surgiu como uma resposta a essas dificuldades, oferecendo:

- Configuração Automática;
- Iniciação Rápida;
- *Standalone Applications*;
- Facilidade de Integração.

Configuração Automática

O Spring Boot é capaz de configurar automaticamente muitos aspectos da aplicação com base nas dependências encontradas no projeto. Isso significa que, ao adicionar uma dependência, como um banco de dados, o Spring Boot configura automaticamente os componentes necessários para que esse banco de dados funcione na aplicação.

Iniciação Rápida

Com o Spring Boot, é possível iniciar uma nova aplicação em poucos minutos. Ele fornece um conjunto de “starters” (conjuntos de dependências pré-configuradas) que permitem adicionar funcionalidades específicas, como segurança, acesso a banco de dados, e desenvolvimento web, de forma simples e direta.

Standalone Applications

As aplicações Spring Boot podem ser executadas como aplicações *standalone*, sem a necessidade de um servidor de aplicação externo. Isso é possível porque o Spring Boot embute servidores como *Tomcat* ou *Jetty* diretamente na aplicação, facilitando o processo de *deployment*.

Facilidade de Integração

O Spring Boot facilita a integração com outras tecnologias e frameworks, permitindo que você construa aplicações complexas e escaláveis sem precisar se preocupar com a infraestrutura.

- 1. Opiniated Defaults:** Spring Boot segue o princípio de “convenção sobre configuração”, onde escolhas padrão são feitas para você, permitindo que você se concentre na lógica de negócios em vez de em detalhes de configuração.
- 2. Microservices:** Embora possa ser usado para criar qualquer tipo de aplicação, Spring Boot é especialmente popular no desenvolvimento de microservices, devido à sua capacidade de criar serviços independentes e facilmente implantáveis.
- 3. Comunidade e Suporte:** Spring Boot tem uma grande comunidade de desenvolvedores, o que significa que há uma vasta quantidade de recursos, tutoriais e suporte disponíveis para quem adota a tecnologia.

Spring Boot & POO

Spring Boot não só simplifica o desenvolvimento, mas também promove boas práticas de Programação Orientada a Objetos (POO). Ao organizar o código em diferentes camadas (controladores, serviços, repositórios), ele encoraja a separação de responsabilidades, um princípio chave da POO. Além disso, ele facilita a implementação de conceitos como injeção de dependências, que promove o acoplamento fraco e a reutilização de código.

Em resumo, Spring Boot é uma escolha poderosa e eficiente para desenvolvedores que desejam construir aplicações Java modernas, escaláveis e bem estruturadas, utilizando os princípios da POO. Ele não apenas acelera o desenvolvimento, mas também garante que a aplicação siga boas práticas desde o início.

POO2

Conceitos de Spring Boot

Arquitetura do Spring Boot



A arquitetura do Spring Boot é projetada para seguir uma estrutura modular e orientada a componentes, facilitando o desenvolvimento de aplicações Java robustas e escaláveis.

A base dessa arquitetura está na divisão clara de responsabilidades em diferentes camadas, permitindo que o código seja organizado de forma eficiente e fácil de manter.

Camada de Apresentação (Controller)

- Função: Lida com as requisições HTTP recebidas pela aplicação. Os controladores processam essas requisições, invocam a lógica de negócios necessária e retornam as respostas adequadas ao cliente.
- Anotação Principal: @RestController, @GetMapping, @PostMapping.

Camada de Serviço (Service)

- Função: Contém a lógica de negócios da aplicação. Os serviços coordenam as operações necessárias entre diferentes repositórios e/ou outras lógicas de negócio.
- Anotação Principal: @Service.

Camada de Acesso a Dados (Repository)

- Função: Interage diretamente com o banco de dados, realizando operações de CRUD (Create, Read, Update, Delete). Essa camada utiliza JPA (Java Persistence API) para mapear objetos Java para tabelas no banco de dados.
- Anotação Principal: @Repository, @Entity.

Inversão de Controle (IoC)

Inversão de Controle (IoC) é um princípio fundamental em frameworks como o Spring Boot, que inverte a responsabilidade pela criação e gerenciamento de objetos dentro de uma aplicação.

Tradicionalmente, em programação orientada a objetos, o código de uma classe instanciaria diretamente suas dependências, criando um forte acoplamento entre os componentes. Isso tornava o código mais difícil de manter e testar.

Inversão de Controle (IoC)

Com IoC, essa responsabilidade é invertida: em vez de uma classe instanciar suas dependências, o framework Spring gerencia a criação e a injeção desses objetos (conhecidos como beans).

A configuração das dependências é feita externamente, geralmente por meio de anotações ou arquivos de configuração.

O Spring Boot, por exemplo, automaticamente injeta essas dependências onde necessário, permitindo que o desenvolvedor se concentre na lógica de negócios, sem se preocupar com a criação e o gerenciamento manual dos objetos.

Benefícios da IoC

- Desacoplamento: Reduz o acoplamento entre componentes, facilitando a manutenção e evolução do código.
- Reutilização: Facilita a reutilização de código, uma vez que os componentes podem ser facilmente substituídos ou atualizados.
- Testabilidade: Torna o código mais fácil de testar, pois as dependências podem ser facilmente simuladas ou substituídas em testes unitários.

Injeção de Dependências (DI)

Injeção de Dependências (DI) é um padrão de design usado para implementar o princípio da Inversão de Controle (IoC). Em vez de as classes criarem suas próprias dependências internamente, essas dependências são fornecidas a elas externamente, geralmente pelo framework que está gerenciando o ciclo de vida dos objetos.

No contexto do Spring Boot, DI permite que o framework injete automaticamente as dependências necessárias em uma classe sem que esta precise instanciá-las manualmente. Isso é feito através de anotações como `@Autowired`, `@Inject`, ou `@Resource`, que indicam ao Spring que a dependência deve ser injetada.

Injeção de Dependências (DI)

Como Funciona

- Definição de Beans: As classes que representam as dependências são configuradas como beans no contexto da aplicação Spring.
- Injeção Automática: Quando uma classe requer um bean, o Spring identifica a dependência necessária e a injeta automaticamente no ponto apropriado, seja no construtor, em métodos setters, ou diretamente nos campos.

Benefícios da DI

- Redução de Acoplamento: As classes não precisam saber como suas dependências são criadas, promovendo um acoplamento mais fraco entre os componentes.
- Facilidade de Teste: Como as dependências são injetadas, é fácil substituir essas dependências por versões simuladas durante testes unitários.
- Flexibilidade: Facilita a mudança de implementações de dependências sem modificar o código das classes que as utilizam.

POO2

Conceitos de Spring Boot

Configuração e Estrutura

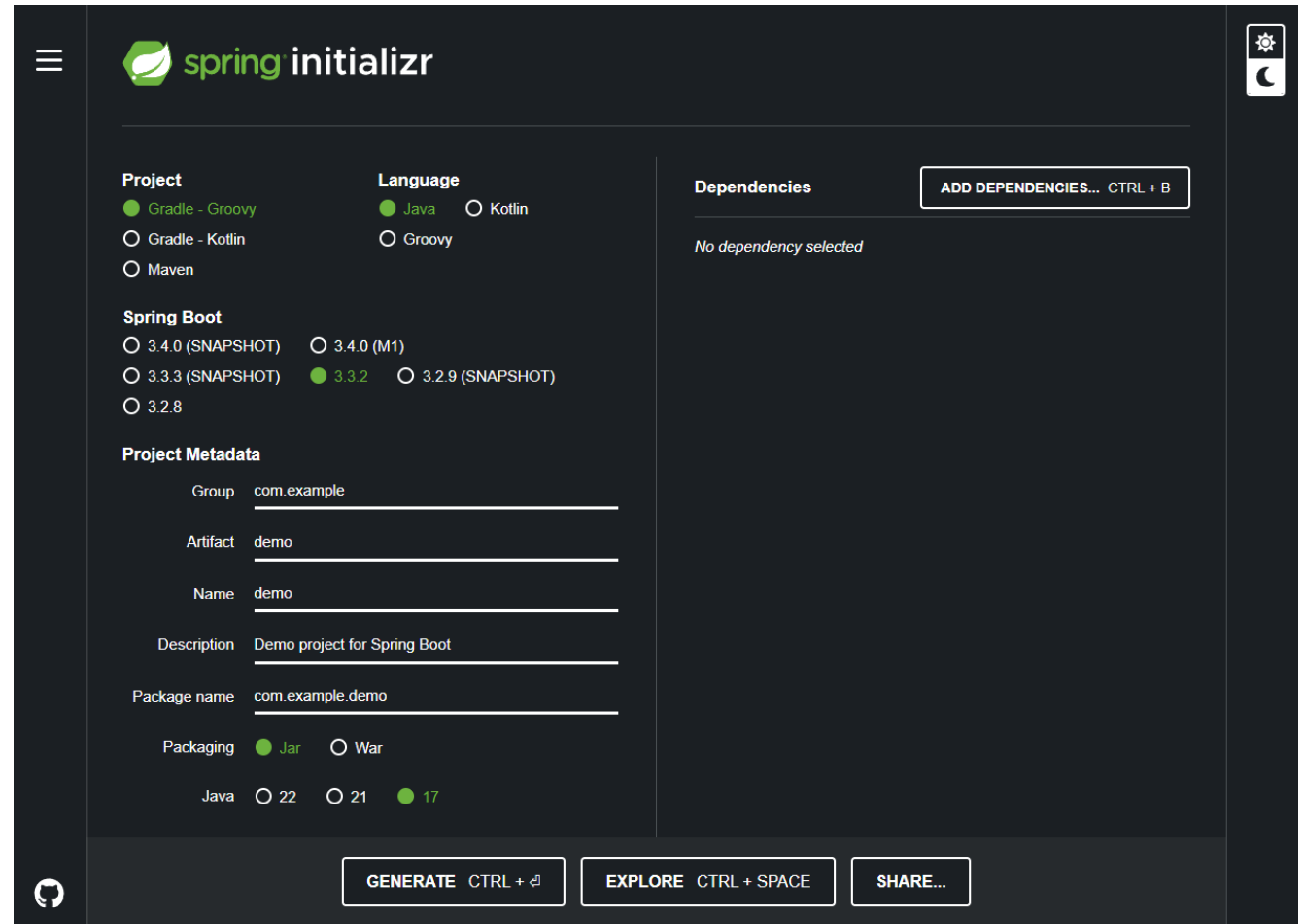


Criando um projeto Spring Boot

O ***Spring Initializr*** é uma ferramenta que permite a criação de projetos Spring Boot rapidamente, selecionando as dependências e gerando a estrutura inicial do projeto.

Acesse a seguinte url:

<https://start.spring.io/>



The screenshot shows the Spring Initializr web application interface. The header includes the Spring logo and the text "spring initializr". The main content area is divided into several sections:

- Project:** Radio buttons for ☒ Gradle - Groovy, ☐ Gradle - Kotlin, and ☐ Maven.
- Language:** Radio buttons for ☒ Java, ☐ Kotlin, and ☐ Groovy.
- Spring Boot:** Radio buttons for ☐ 3.4.0 (SNAPSHOT), ☐ 3.4.0 (M1), ☐ 3.3.3 (SNAPSHOT), ☒ 3.3.2, and ☐ 3.2.9 (SNAPSHOT). There is also a ☐ 3.2.8 option.
- Project Metadata:** Text input fields for Group (com.example), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), and Package name (com.example.demo).
- Packaging:** Radio buttons for ☒ Jar and ☐ War.
- Java:** Radio buttons for ☐ 22, ☐ 21, and ☒ 17.
- Dependencies:** A section with a button "ADD DEPENDENCIES... CTRL + B" and the text "No dependency selected".

At the bottom, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

Criando um projeto Spring Boot

Project: Maven

Language: Java

Spring Boot: 3.3.2

Group: poo2

Artifact: estoque

Name: estoque

Description: API do Estoque

Package Name:poo2.estoque

Packaging: Jar

Java: 21



The image shows the Spring Initializr web interface, which is used to generate a Spring Boot project. The interface is dark-themed and contains several sections for configuring the project:

- Project:** Radio buttons for `Gradle - Groovy`, `Gradle - Kotlin`, and `Maven` (selected).
- Language:** Radio buttons for `Java` (selected), `Kotlin`, and `Groovy`.
- Spring Boot:** Radio buttons for versions `3.4.0 (SNAPSHOT)`, `3.4.0 (M1)`, `3.3.3 (SNAPSHOT)`, `3.3.2` (selected), `3.2.9 (SNAPSHOT)`, and `3.2.8`.
- Project Metadata:** Text input fields for `Group` (poo2), `Artifact` (estoque), `Name` (estoque), `Description` (API do Estoque), and `Package name` (poo2.estoque).
- Packaging:** Radio buttons for `Jar` (selected) and `War`.
- Java:** Radio buttons for versions `22`, `21` (selected), and `17`.
- Dependencies:** A section with the text "No dependency selected" and a button "ADD DEPENDENCIES... CTRL + B".

At the bottom of the interface, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

Criando um projeto Spring Boot



The image shows the Spring Initializr web form. It is divided into several sections: Project, Language, Spring Boot, Project Metadata, Dependencies, and Action buttons. The Project section has radio buttons for Gradle - Groovy, Gradle - Kotlin, and Maven (selected). The Language section has radio buttons for Java (selected), Kotlin, and Groovy. The Spring Boot section has radio buttons for 3.4.0 (SNAPSHOT), 3.4.0 (M1), 3.3.3 (SNAPSHOT), 3.3.2 (selected), and 3.2.9 (SNAPSHOT). The Project Metadata section has input fields for Group (poo2), Artifact (estoque), Name (estoque), Description (API do Estoque), and Package name (poo2.estoque). The Packaging section has radio buttons for Jar (selected) and War. The Java version section has radio buttons for 22, 21 (selected), and 17. The Dependencies section has a list of dependencies: Spring Web (WEB), Spring Data JPA (SQL), and H2 Database (SQL). Each dependency has a minus button to remove it. At the bottom, there are three buttons: GENERATE (CTRL + G), EXPLORE (CTRL + SPACE), and SHARE... The top right of the form has a settings icon and a dark mode toggle.

Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ Maven

Language

☒ Java

☐ Kotlin

☐ Groovy

Spring Boot

☐ 3.4.0 (SNAPSHOT)

☐ 3.4.0 (M1)

☐ 3.3.3 (SNAPSHOT)

☒ 3.3.2

☐ 3.2.9 (SNAPSHOT)

☐ 3.2.8

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging

☒ Jar

☐ War

Java

☐ 22

☒ 21

☐ 17

Dependencies

Spring Web WEB

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

Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

ADD DEPENDENCIES... CTRL + B

GENERATE CTRL + G

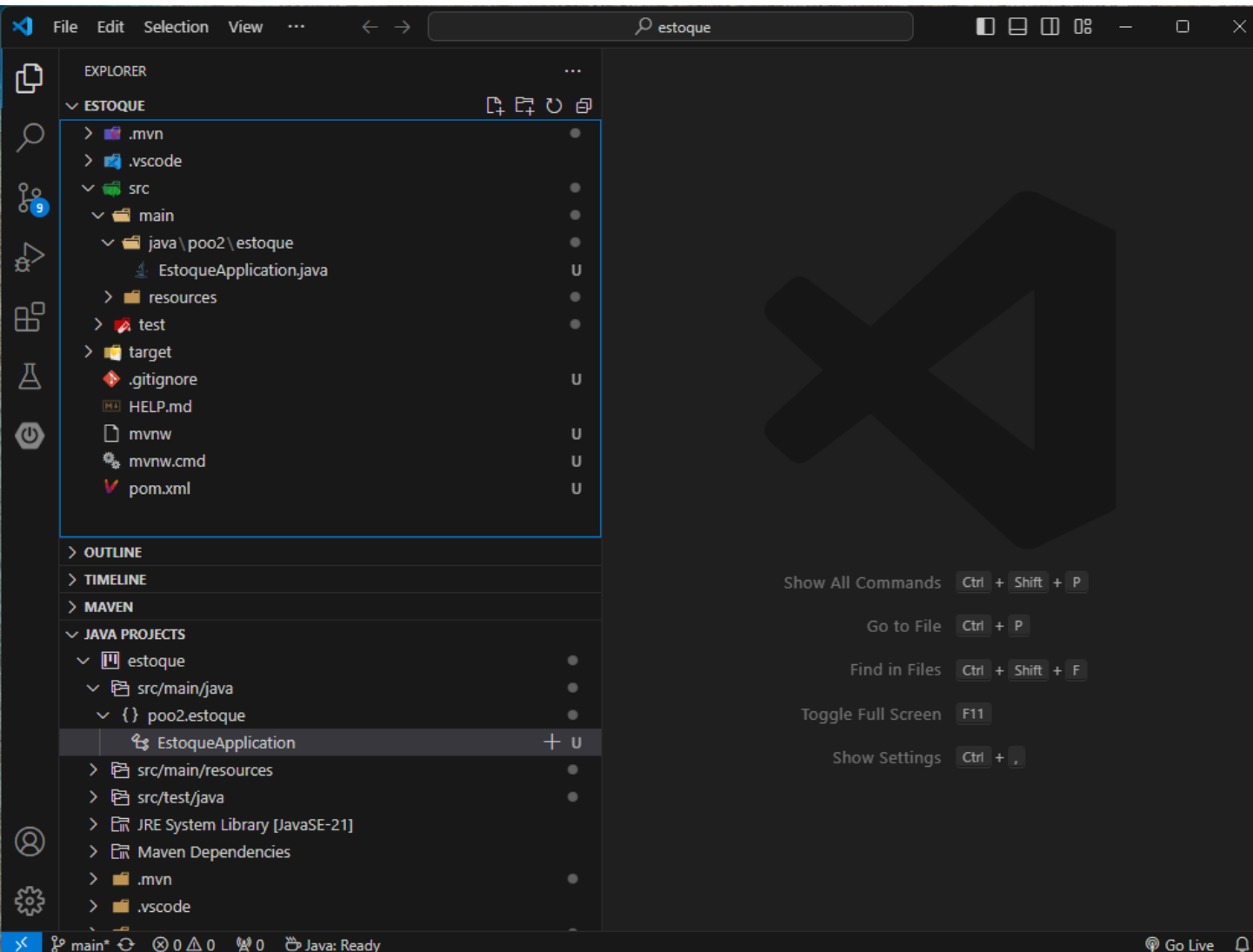
EXPLORE CTRL + SPACE

SHARE...

Dependencies:
Spring Web
Spring Data JPA
H2 Database

GENERATE
Download arquivo .zip

Abrindo o projeto Spring Boot



Após download do arquivo zip, descompactar a pasta, e abrir pelo VS Code.

ATENÇÃO

- Antes de abrir, verificar se as extensões foram instaladas adequadamente.
- Extension Pack for Java;
- Spring Boot Extension Pack;

Arquivos principais

Um projeto Spring Boot é estruturado de maneira a facilitar o desenvolvimento rápido e eficiente de aplicações.

A seguir destacamos os principais arquivos e pastas encontrados em um projeto Spring Boot, e suas respectivas funções.

Esses arquivos e pastas formam a espinha dorsal de um projeto Spring Boot e são essenciais para o desenvolvimento, configuração, e operação da aplicação.

Arquivos principais

Pasta src/main/java – contém o código Java da aplicação. É onde encontramos as principais classes e pacotes do projeto.

- **Application.java**

- Arquivo principal que contém o método main. É o ponto de entrada da aplicação e é anotado com `@SpringBootApplication`, que inclui configurações para auto-configuração, varredura de componentes e configuração do Spring.

- **Pastas Controller, Service, Repository**

- Contêm a lógica da aplicação, incluindo controladores para endpoints REST, serviços para lógica de negócios, e repositórios para acesso a dados.

Pasta src/main/resources – contém arquivos de configuração e outros recursos não-Java utilizados pela aplicação.

- **application.properties ou application.yml**

- Arquivo de configuração principal da aplicação. Define propriedades como a configuração do banco de dados, parâmetros de aplicação, e configurações de logging.

- **Pasta static**

- Contém arquivos estáticos, como arquivos CSS, JavaScript e imagens, que são servidos diretamente pelo servidor web integrado.

Arquivos principais

Pasta src/main/resources – contém arquivos de configuração e outros recursos não-Java utilizados pela aplicação.

- **Pasta templates**

- Contém templates HTML para renderização de páginas se você estiver usando Thymeleaf ou outro mecanismo de template.

- **application-dev.properties / application-prod.properties**

- Arquivos de configuração específicos para diferentes perfis (dev, prod, etc.). Permitem a configuração de propriedades específicas para ambientes distintos.

Pasta `src/test/java` – contém testes unitários e de integração para o código Java da aplicação.

- **Testes Unitários e de Integração**

- Arquivos de teste que garantem que o código funciona corretamente. Usualmente escritos com JUnit e/ou Mockito, e podem incluir testes para controladores, serviços e repositórios.

Arquivos principais

pom.xml ou **build.gradle** – arquivo de Configuração de Build, que contém as dependências do projeto e configurações de build para Maven (pom.xml) ou Gradle (build.gradle).

- **pom.xml (para Maven)**

- Define as dependências do projeto, plugins de build, e configurações do Maven.

- **build.gradle (para Gradle)**

- Define as dependências do projeto, plugins de build e configurações do Gradle.

Observação: neste projeto, utilizaremos Maven.

Arquivos principais

README.md – arquivo de documentação que fornece uma visão geral do projeto, instruções de configuração e uso, e outras informações relevantes para desenvolvedores.

.gitignore – arquivo que especifica quais arquivos e pastas devem ser ignorados pelo Git, evitando que arquivos temporários, compilados ou de configuração local sejam versionados.

Dockerfile (opcional) – arquivo para criar uma imagem Docker da aplicação, facilitando o deployment em containers.

docker-compose.yml (opcional) – arquivo para definir e executar multi-containers Docker applications, útil para configurar ambientes de desenvolvimento e testes complexos.

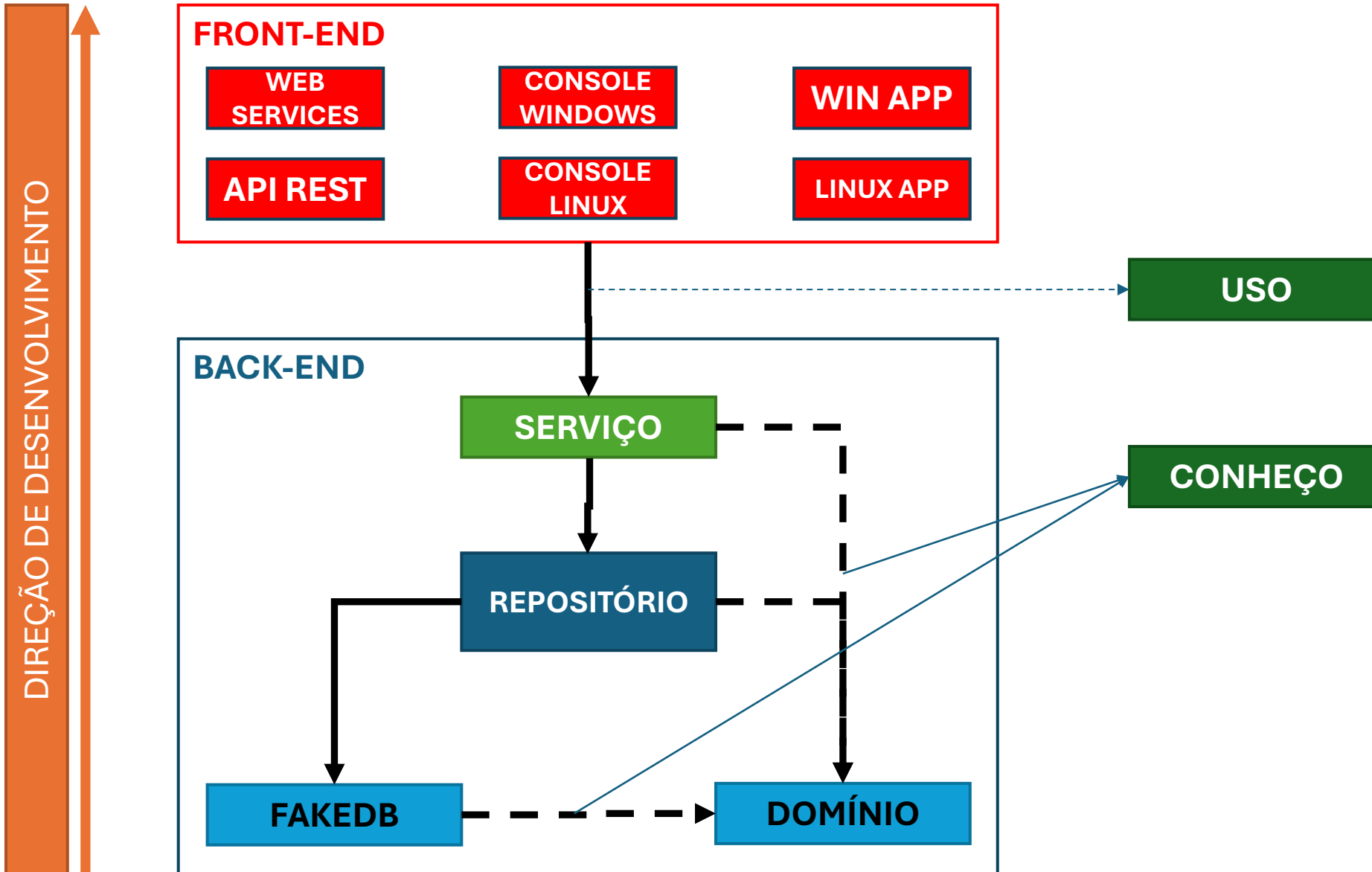
POO2

Conceitos de Spring Boot

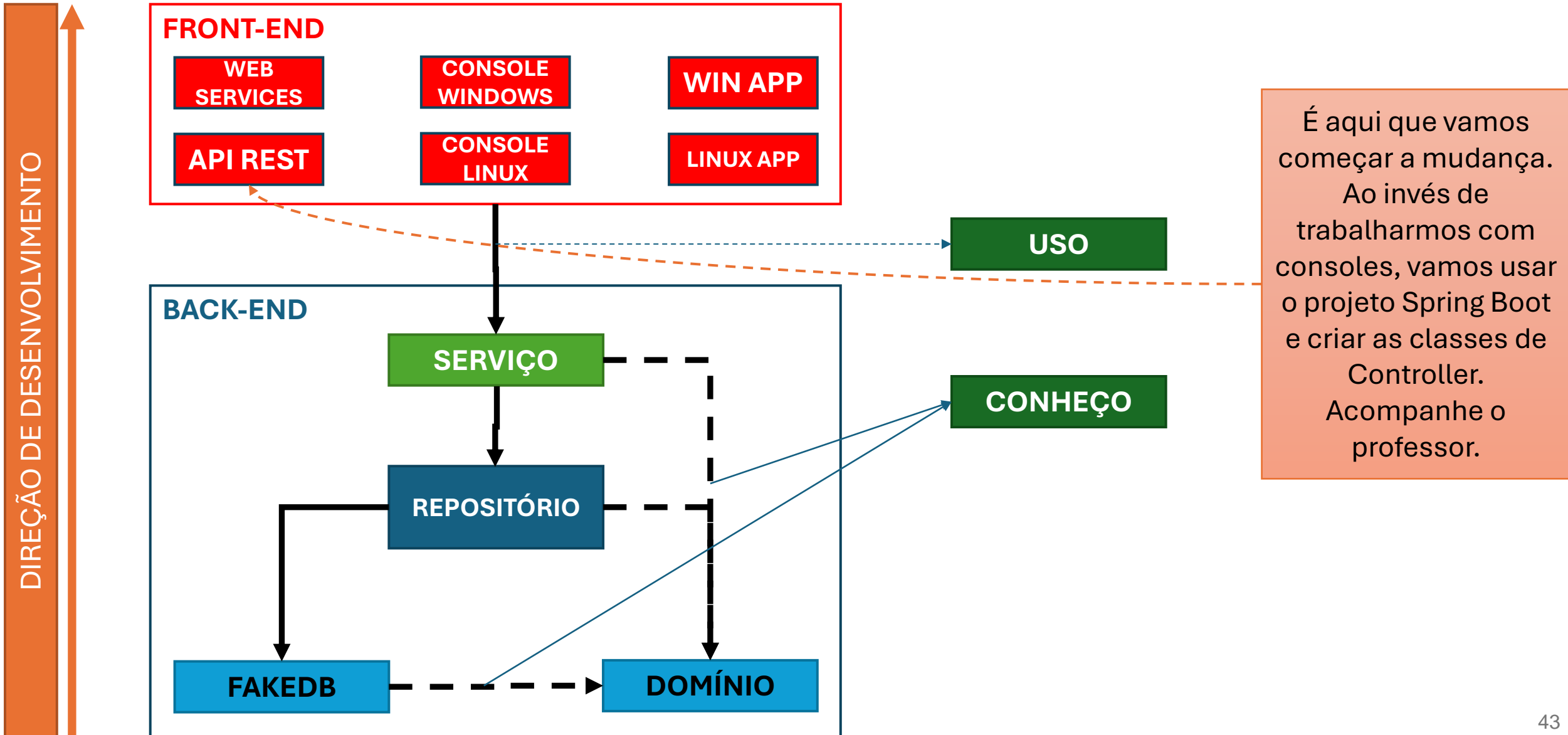
Estrutura Revisitada



Estrutura de Projeto



Estrutura de Projeto



POO2

Conceitos do Spring Boot

Referências



Referências

Documentação Oficial do Spring Boot:

[Spring Boot Documentation]

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

Tutoriais:

[Baeldung] – Spring Boot

<https://www.baeldung.com/spring-boot>

Spring Boot Guides

<https://spring.io/guides>

Dúvidas?





Dúvidas, críticas e sugestões, entre em contato
através do e-mail do professor:

luiz.a.rodriques@cogna.com.br