



# Desenvolvimento Java com Spring Boot



- Não é um framework apenas;
- É conjunto de projetos que resolvem várias situações do cotidiano de um programador,
- Ajuda a criar aplicações Java com simplicidade e flexibilidade;
- Ele foi pensado para que nossas aplicações pudessem focar mais na regra de negócio e menos na infraestrutura.



# SPRING x SPRING BOOT



- O ecossistema Spring não é apenas o Spring Framework.
- Mas, o Spring Framework é apenas um, dentre o conjunto de projetos, que o Spring possui.
- Ele é o projeto do Spring que serve de base para todos os outros, talvez por isso haja essa pequena confusão.





# SPRING x SPRING BOOT



- Enquanto os componentes do Spring eram simples, sua configuração era extremamente complexa e cheia de XMLs.
- Depois de um tempo, a partir da versão 3.0, a configuração pôde ser feita através de código Java.



# SPRING x SPRING BOOT



- Talvez a maior revolução e o maior acerto dos projetos Spring, foi o Spring Boot.
- Com ele você alcança um novo paradigma para desenvolver aplicações Spring com pouco esforço.
- O Spring Boot não gera código! Ele simplesmente analisa o projeto e automaticamente o configura.

# SPRING BOOT



- A documentação do Spring boot pode ser encontrada em: <https://goo.gl/Z62B4C>
- Ela também está disponível em pdf, epub e outros formatos, para conferir os formatos disponíveis: <https://goo.gl/oDVEQE>



# SPRING SECURITY

- Spring Security torna bem simples a parte de autenticação;
- Com algumas poucas configurações já podemos ter uma autenticação via banco de dados ou mesmo por memória;
- Através das permissões que atribuímos aos usuários autenticados, podemos proteger as requisições web (como as telas do nosso sistema, por exemplo), a simples invocação de um método e até a instância de um objeto.



# HIBERNATE

- Realiza o mapeamento objeto-relacional (ORM) e é a base para a java persistence API (JPA);
- Abstrai o seu código SQL e toda a camada JDBC;
- Mais que isso, ele vai gerar o SQL que serve para um determinado banco de dados, já que cada banco fala um "dialeto" diferente dessa linguagem.





# THYMELEAF

- Interpretador de Java Server Pages (JSP) que cuida de toda interface entre o código java e o HTML;
- Teremos um código HTML misturado com alguns atributos do Thymeleaf, que após processado, será gerado apenas o HTML para ser renderizado no browser do cliente.



# THYMELEAF

- O Thymeleaf não é um projeto Spring, mas uma biblioteca que foi criada a facilitar a criação da camada de view com uma forte integração com o Spring, sendo assim uma boa alternativa ao JSP;
- O principal objetivo do Thymeleaf é prover uma forma elegante e bem formatada para criarmos nossas páginas.
- Documentação: <https://goo.gl/g5yBub>

# ALGUMAS TAGS ÚTEIS

- `th:if`
- `th:each`
- `th:href`
- `th:object`
- `th:action`
- `${T(caminho_enum).values()}` - pega valores de enum
- `th:attr`
- `th:field`
- `<!--/*/ <th:block`  
`th:include="caminho_do_arquivo_a_ser_incluido ::`  
`head"></th:block> /*/-->`

# EXEMPLO DE USO

- `th:if="${variavel_do_controler}"`
- `th:each="nova variavel : ${variavel_do_controler}"`
- `th:href="@{/caminho}"`
- `th:object="${atributo_do_controller}"`
- `th:action="@{/url_destino}"`
- `th:attr="data-target='#'+${noticia.id}"`
- `th:field="*{variavel_a_receber_valores}"`





- O Maven é uma ferramenta muito importante no dia a dia do desenvolvedor Java;
- Com ele nós conseguimos automatizar uma série de tarefas
  - Build da aplicação;
  - Gerenciamento de dependências;
  - etc.



# INTELLIJ IDEA

- IntelliJ IDEA é um Java IDE por JetBrains, que oferece suporte para todos os desenvolvedores que querem trabalhar com Frameworks, serviços corporativos e dispositivos móveis.
- Para programação Web o IntelliJ IDEA possui várias ferramentas que irão ajudar, usando ferramentas como Spring MVC , Web Services , JSF e Struts além de incluir assistência a códigos de programação HTML5, CSS3, SASS , JavaScript e Node.js.



- `$ git clone -b branch url_do_repositorio`
  - Baixa o repositório na branch escolhida, caso não haja mais de uma branch a parte “-b branch” não é necessária.
- `$ git add .`
  - Coloca seu arquivo na lista de modificados (Index)
- `$ git commit -m “Mensagem do commit”`
  - Comita seus arquivos do Index para o servidor local
- `$ git push origin master`
  - Envia seus arquivos para o repositório remoto
- `$ git pull`
  - Baixa as modificações do repositório remoto
- `$ git checkout .`
  - Reverte suas modificações para o master

# O QUE SERÁ O NOSSO PROJETO

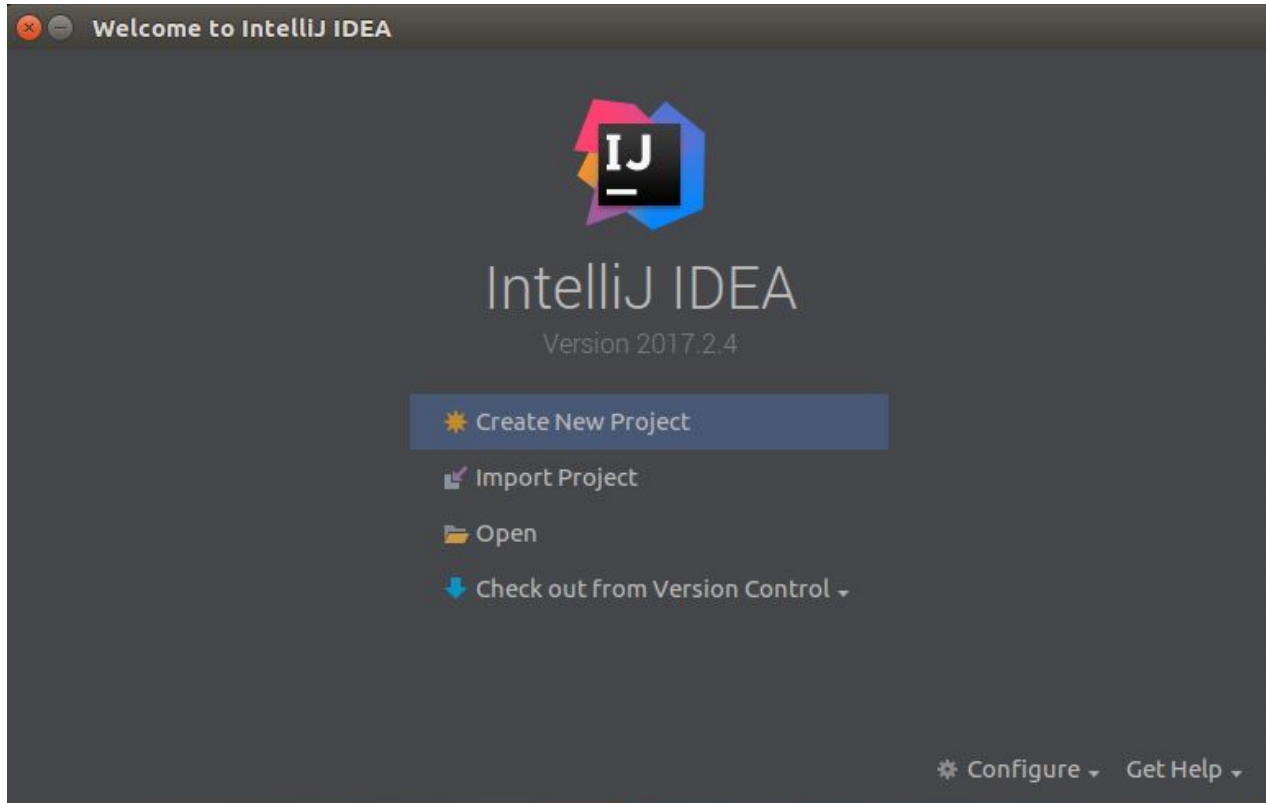
Bora colocar a mão na massa?

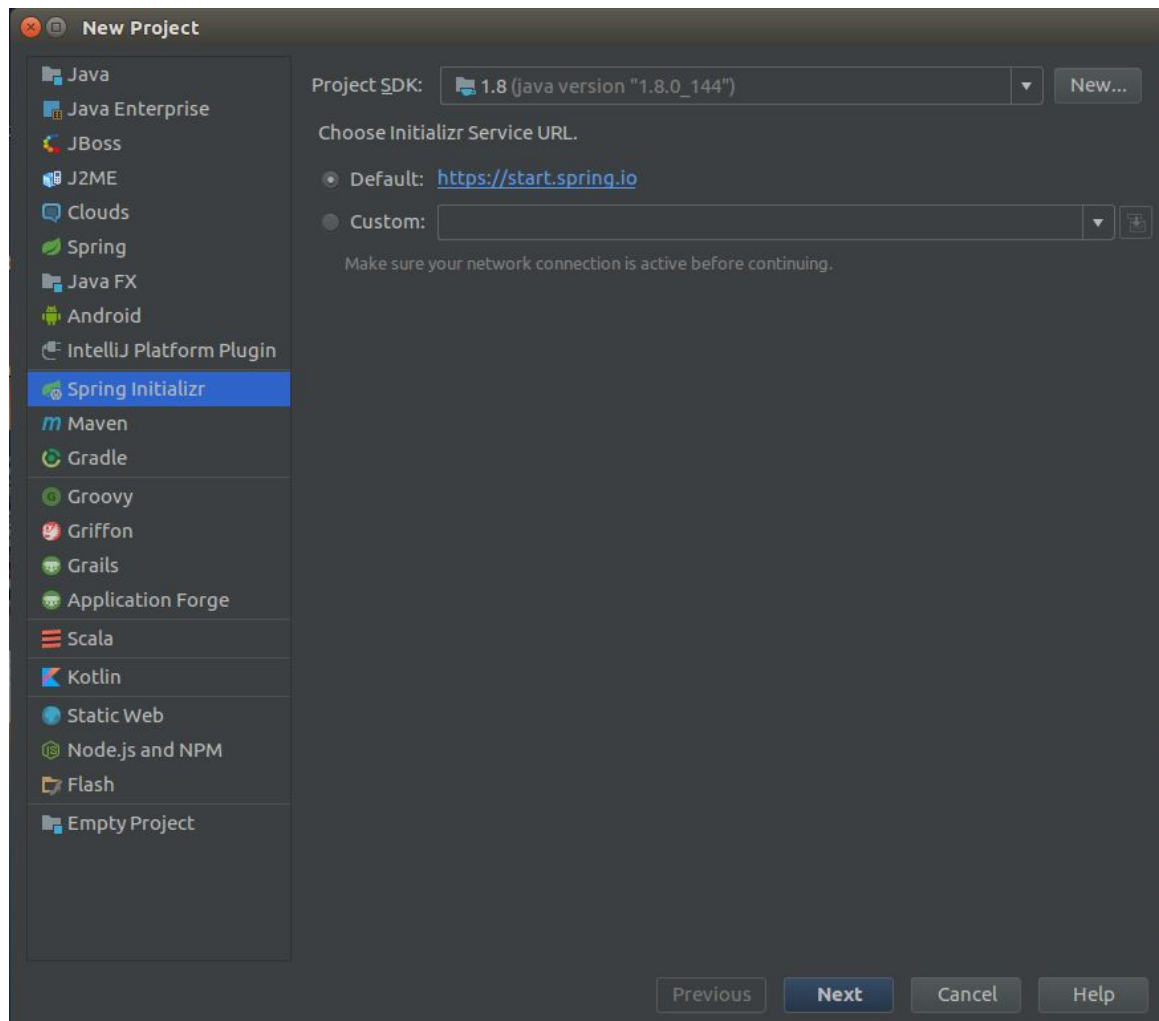
Vamos criar uma aplicação simples, do zero e passo a passo para você ver como o Spring Boot, o Spring MVC, o Spring Data JPA, o Spring Security e o Thymeleaf funcionam juntos, e para isso vamos usar o IntelliJ IDEA e o Maven.



Nossa aplicação será um sistema médico com algumas funcionalidades para exemplificar o funcionamento de uma aplicação web.



# CRIANDO O PROJETO





  **New Project**

**Project Metadata**

Group:

Artifact:

Type:  ▼

Language:  ▼

Packaging:  ▼

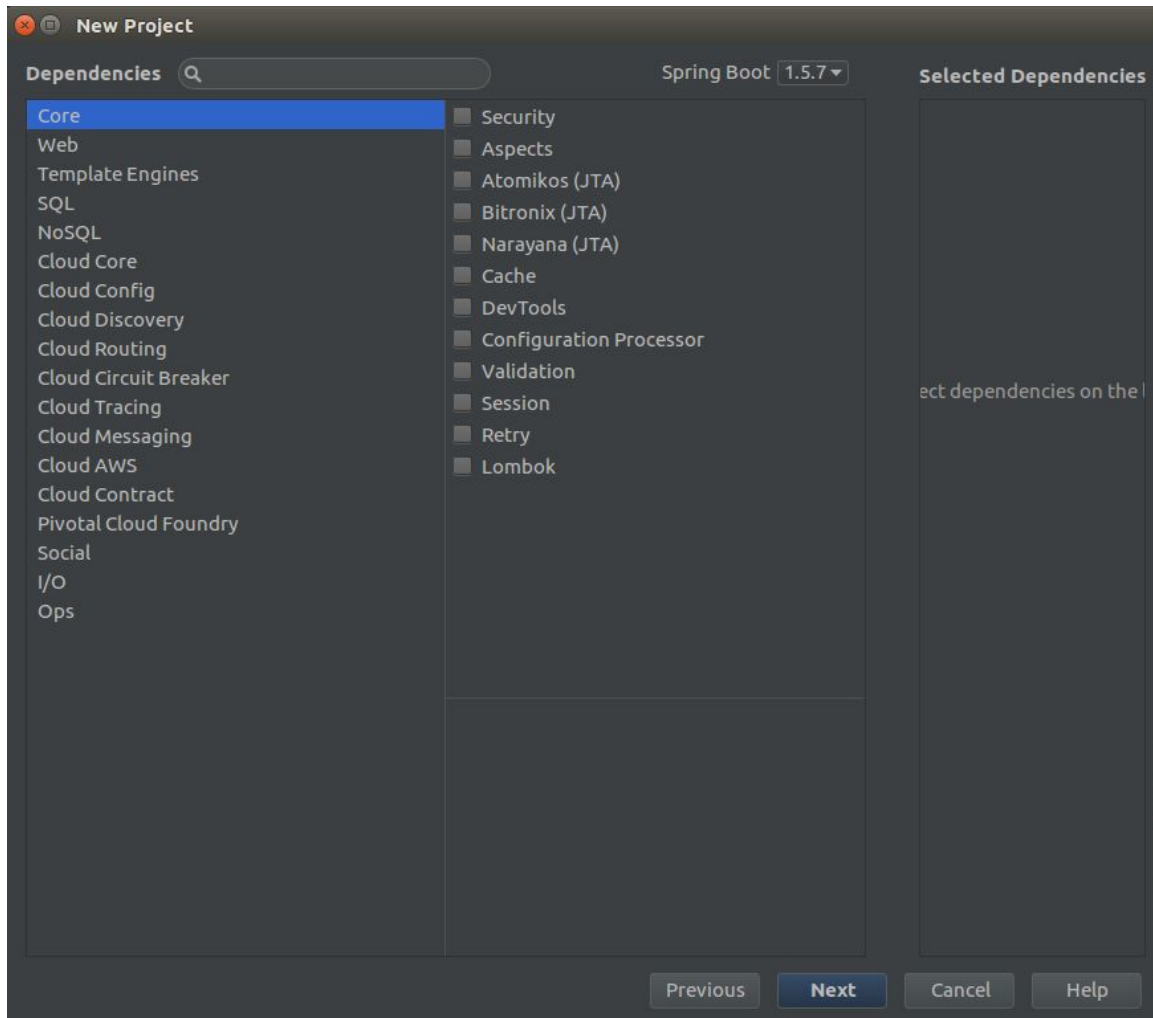
Java Version:  ▼

Version:

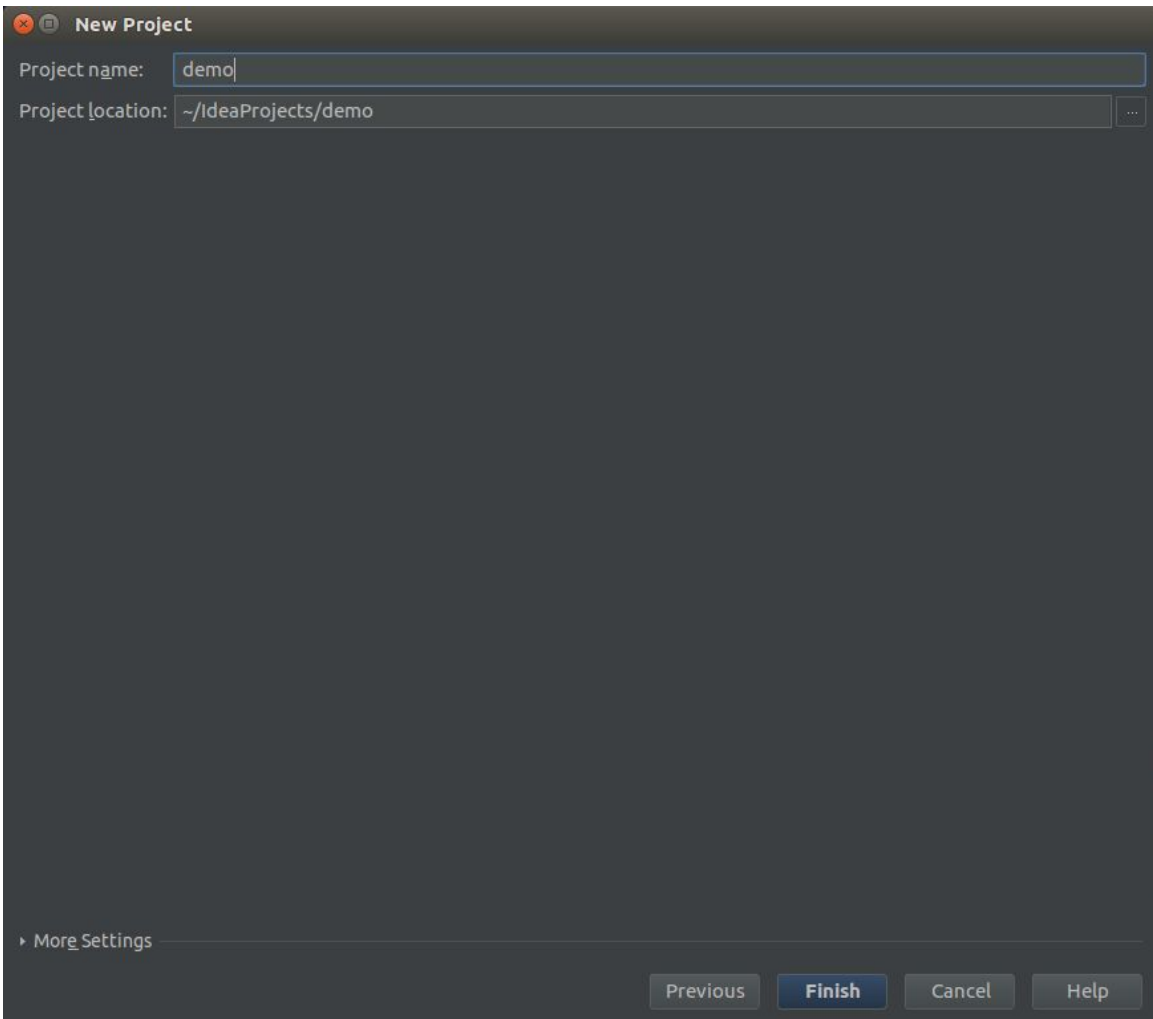
Name:

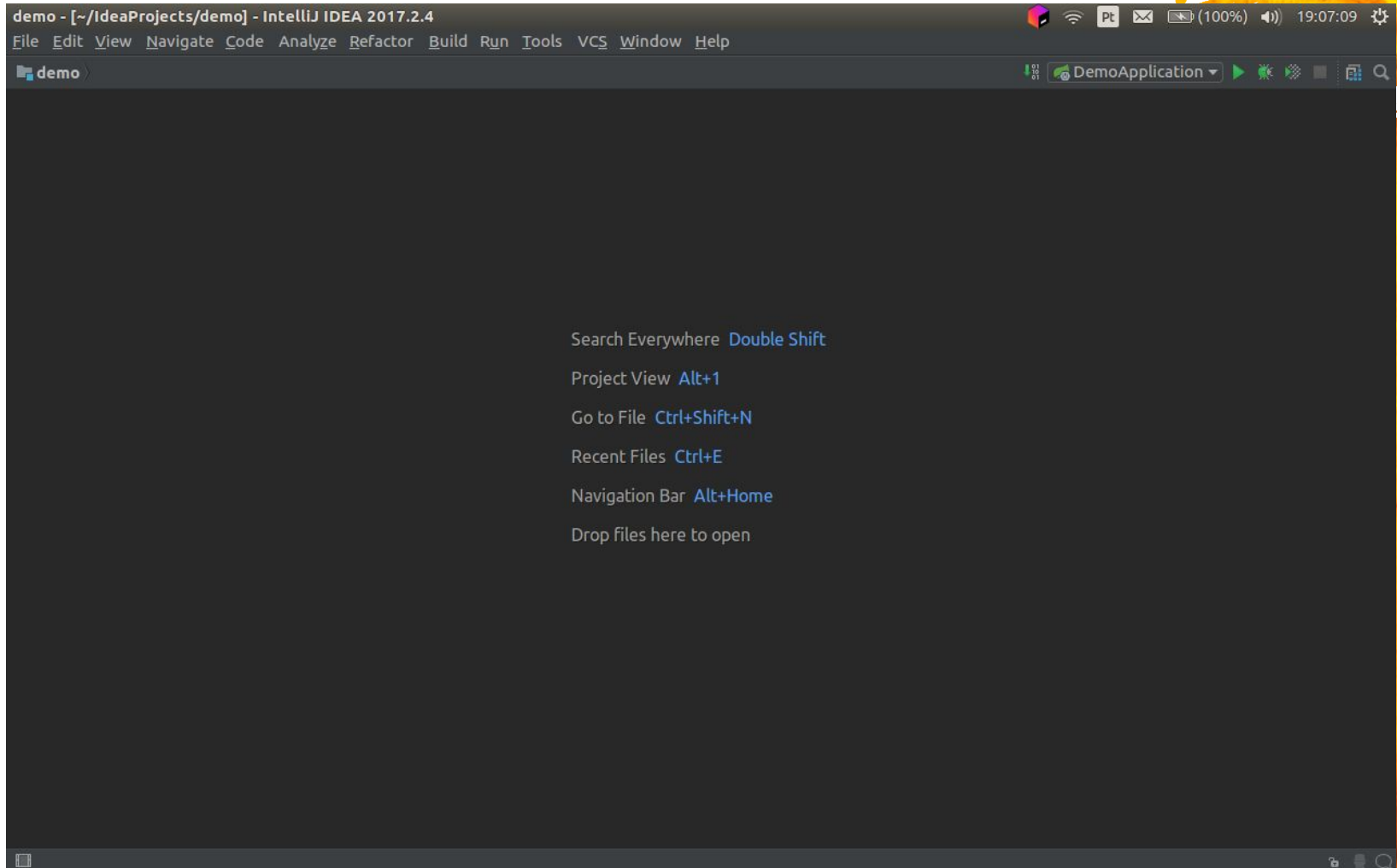
Description:

Package:









# CONFIGURAÇÕES

- Realizaremos algumas modificações no pom.xml e adicionaremos algumas configurações no application.properties

# ARQUIVO DE CONFIGURAÇÃO

- Grande parte das configurações do spring boot podem ser feitas através do arquivo **application.properties**.
- Alguns exemplos de configuração de arquivos foram vistos no slide anterior, mas a referência completa de todas as configurações que podem ser feitas neste arquivo pode ser encontrada em: <https://goo.gl/taVRtF>



# MODELO (Model)

Um modelo é uma representação da estrutura lógica fundamental dos dados em uma aplicação de software e a classe de alto nível associada a ele. Este modelo de objeto não contém nenhuma informação sobre a interface do usuário.



# EXEMPLO

@Entity

```
public class Paciente {
```

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)

```
    private Integer id;
```

    @NotEmpty

    @Size(max = 14)

    @Column(unique = true)

```
    private String cpf; }
```



# NOSSO PRIMEIRO MODEL

Criaremos nosso primeiro model.



# MAPEAMENTO DE REQUISIÇÕES

Mapeamento de requisições são as url que serão disponibilizadas para serem utilizadas pelo usuário final da aplicação.

- @RequestMapping, @GetMapping, @PostMapping

Pode ser passado uma única url para mapeamento ou várias. Ex:

- @RequestMapping("/url")
- @RequestMapping({"url1", "url2", "url2"})



# CONTROLADOR (Controller)

Classe responsável por receber a requisição do browser e retornar a view (arquivo a ser renderizado pelo browser).

@Controller

@RequestMapping({"/", "/index", "/index.html"})

public class HomeController {

    @GetMapping

        public String home() {

            return "index";

        }

}

# VISÃO (View)

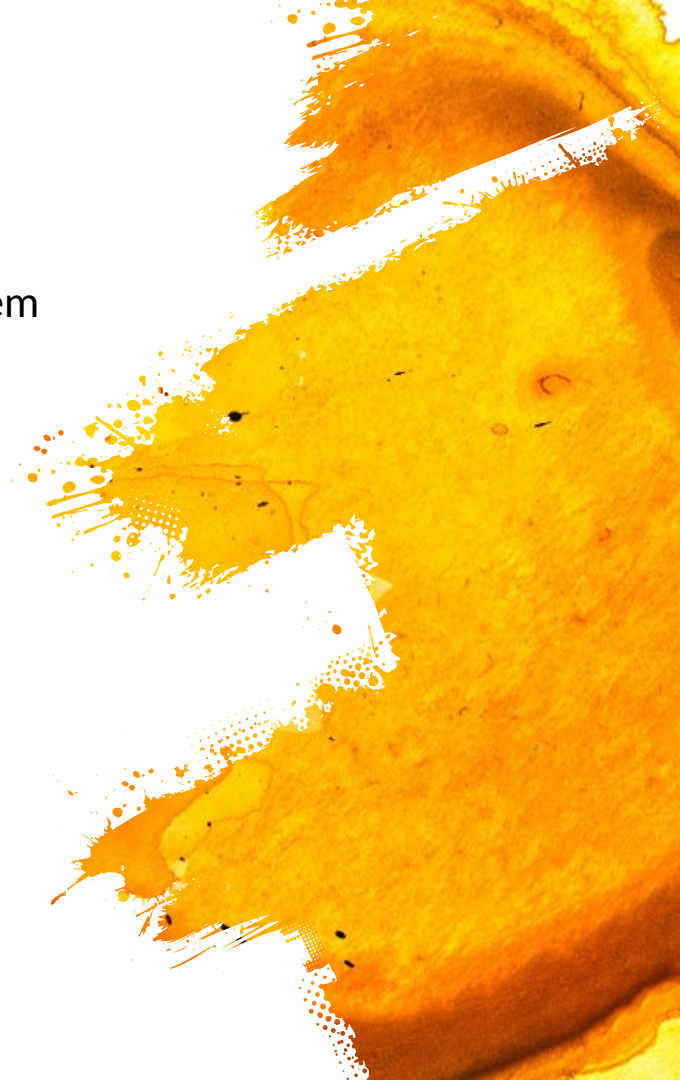
Em poucas palavras a view é a página HTML que será renderizada no browser.

No spring boot todos os arquivos html devem ficar dentro da pasta ***templates*** em ***resources***, e os outros arquivos estáticos, tais como imagens, fontes, js, css, dentre outros devem ficar na pasta ***static*** em ***resources***.

# A PRIMEIRA INTERFACE

Todos os arquivos estáticos estarão disponível em <https://goo.gl/fXghMp>.

Agora vamos criar a nossa primeira interface.



# RODANDO O PROJETO PELA PRIMEIRA VEZ

Para que possamos executar nosso projeto, vamos criar um controller que indicará qual será a home do nosso projeto.



# CRUD

CRUD é um acrônimo para as 4 operações básicas em uma base de dados relacional ou em interface para utilizadores p criação, consulta, atualização e destruição de dados. O significado do acrônimo é: **Create, Read, Update e Delete.**

No spring boot a parte responsável por fazer o CRUD são interfaces que estendem o `JpaRepository<Classe, Tipo_de_Identificador>`

Além das operações definidas pela interface também podemos definir nossas próprias queries com palavras chaves e definindo o tipo de retorno. Ex:

```
List<Usuario> findAllByRole(Role role)
```

A query acima retorna uma lista de usuarios com a role passada.

A documentação do spring data jpa pode ser encontrada em: <https://goo.gl/mQVoT1>

Na documentação você pode achar mais informações de como montar suas próprias queries e outras coisas.

# NOSSO PRIMEIRO REPOSITORY

Criaremos nosso primeiro repository.



# SEGURANÇA

- Para a segurança da nossa aplicação vamos utilizar o `spring security`.
- A documentação do `spring security` pode ser encontrada em: <https://goo.gl/uMPv1S>.
- O `spring security` pode ser implementado com vários tipos de meios de segurança, tais como pelo banco de dados, JWT, OAuth, dentre outros.
- Nesse projeto iremos utilizar com por banco e para codificar a senha utilizaremos o algoritmo Bcrypt, que já está implementado pelo método `BCryptPasswordEncoder()`.



# SEGURANÇA

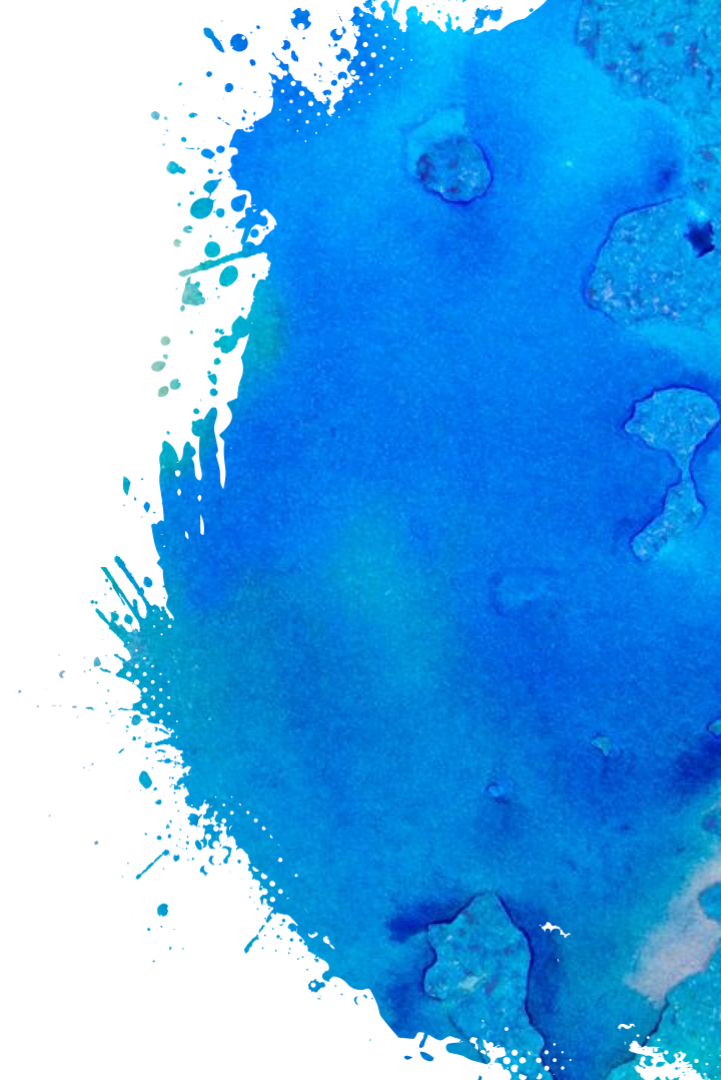
Para adicionar segurança ao projeto basta adicionar ao pom.xml as seguintes linhas:

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-security</artifactId>  
</dependency>
```

Após isso executem a aplicação

# LOGIN

Vamos configurar a parte de login da nossa aplicação



# REGISTRO/CADASTRO

Vamos configurar a parte de registro da nossa aplicação

