

**Guia completo** para montar um projeto CRUD com **Spring Boot**, banco **H2 persistente** e interface HTML básica, tudo pronto para rodar no **VSCode**.



### REQUISITOS de EXTENSIONS no VSCode:



#### Extension Pack for Java

Microsoft [microsoft.com](#) | ⚡ 35,539,245 | ★★★★☆(82)

Popular extensions for Java development that provides Java IntelliSense, debugging, testing, ...

[Disable](#) | [Uninstall](#)

[Switch to Pre-Release Version](#)

Auto Update



#### Spring Boot Extension Pack

VMware [vmware.com](#) | ⚡ 3,282,611 | ★★★★★(20)

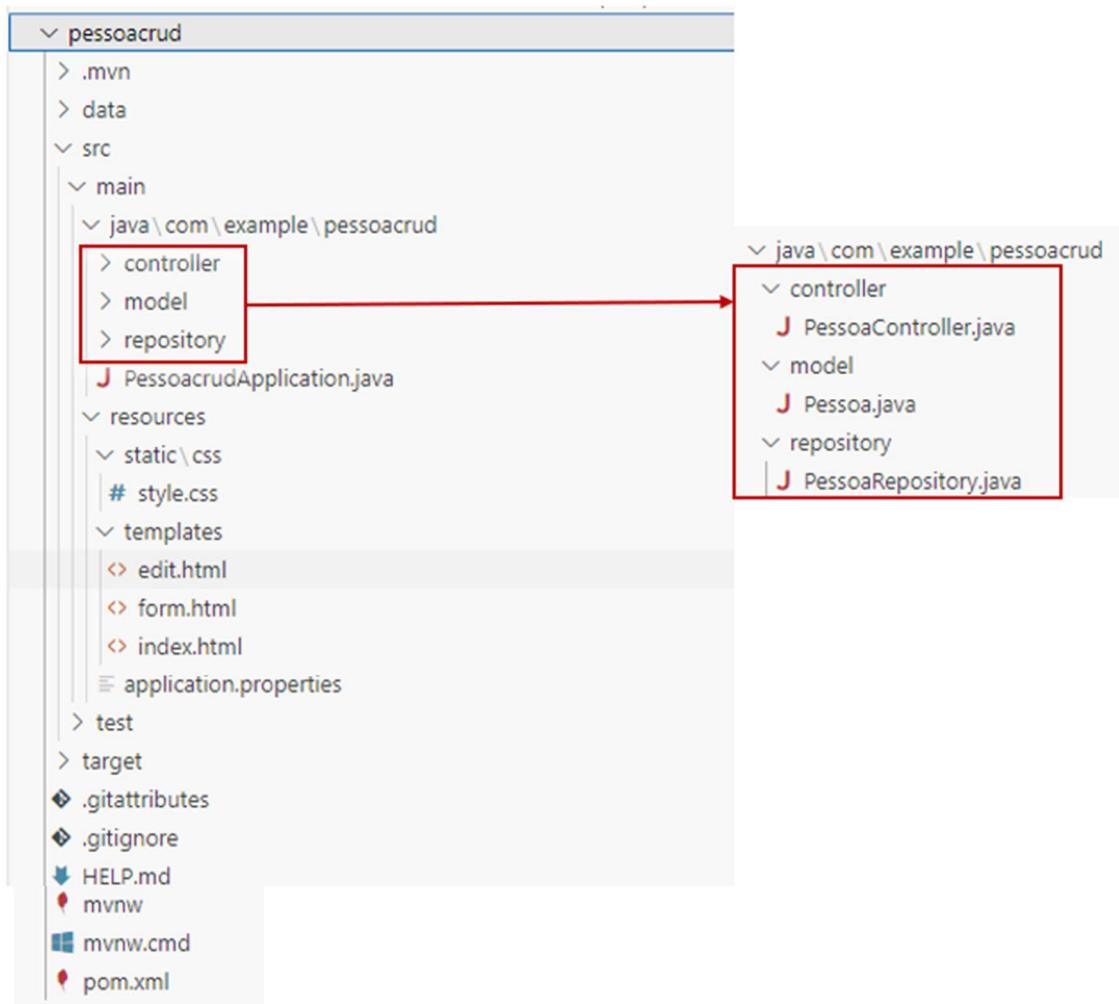
A collection of extensions for developing Spring Boot applications

[Disable](#) | [Uninstall](#)

[Switch to Pre-Release Version](#)

Auto Update

### 📁 Estrutura de pastas esperada



## Passo a passo com Spring Initializr

---

### 1. Acesse o site

👉 <https://start.spring.io/> (ANEXO A)

### 2. Configure os campos:

- **Project:** Maven
- **Language:** Java
- **Spring Boot:** 3.x (ex: 3.2.0)
- **Group:** com.example
- **Artifact:** pessoacrud
- **Name:** pessoacrud
- **Package name:** com.example.pessoacrud
- **Packaging:** Jar

- **Java:** 17 (ou 21, se quiser)

 **3. Adicione as dependências:**

- Spring Web
- Spring Data JPA
- H2 Database
- Thymeleaf

 **4. Clique em → Generate**

Ele vai baixar um **ZIP** pronto. Extraia onde quiser.

 **5. Abra a pasta no VSCode.**

---

 **Configuração do banco (application.properties)**

Abra src/main/resources/application.properties e configure:

```
spring.application.name=pessoacrud
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.datasource.url=jdbc:h2:file:./data/pessoa-db
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Isso garante que o banco H2 seja **persistente** no disco (não apenas memória).

---

 **Criando a entidade Pessoa**

Em src/main/java/com/example/pessoacrud/model/Pessoa.java:

```
package com.example.pessoacrud.model;

import jakarta.persistence.*;

@Entity
public class Pessoa {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nome;
    private int idade;
    private String cargo;

    // Getters e setters
```

```

public Long getId() { return id; }
public void setId(Long id) { this.id = id; }

public String getNome() { return nome; }
public void setNome(String nome) { this.nome = nome; }

public int getIdade() { return idade; }
public void setIdade(int idade) { this.idade = idade; }

public String getCargo() { return cargo; }
public void setCargo(String cargo) { this.cargo = cargo; }
}

```

### **Repositório PessoaRepository**

Em src/main/java/com/example/pessoacrud/repository/PessoaRepository.java:

```

package com.example.pessoacrud.repository;

import com.example.pessoacrud.model.Pessoa;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PessoaRepository extends JpaRepository<Pessoa, Long> {}

```

### **Controlador PessoaController**

Em src/main/java/com/example/pessoacrud/controller/PessoaController.java:

```

package com.example.pessoacrud.controller;

import com.example.pessoacrud.model.Pessoa;
import com.example.pessoacrud.repository.PessoaRepository;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class PessoaController {
    private final PessoaRepository repository;

    public PessoaController(PessoaRepository repository) {
        this.repository = repository;
    }

    @GetMapping("/")
    public String index(Model model) {
        model.addAttribute("pessoas", repository.findAll());
        return "index";
    }

    @GetMapping("/nova")

```

```

public String novaPessoa(Model model) {
    model.addAttribute("pessoa", new Pessoa());
    return "form";
}

{@PostMapping("/salvar")
public String salvarPessoa(@ModelAttribute Pessoa pessoa) {
    repository.save(pessoa);
    return "redirect:/";
}

{@GetMapping("/editar/{id}")
public String editarPessoa(@PathVariable Long id, Model model) {
    model.addAttribute("pessoa", repository.findById(id).orElseThrow());
    return "edit";
}

{@PostMapping("/atualizar/{id}")
public String atualizarPessoa(@PathVariable Long id, @ModelAttribute Pessoa pessoa) {
    pessoa.setId(id);
    repository.save(pessoa);
    return "redirect:/";
}

{@GetMapping("/deletar/{id}")
public String deletarPessoa(@PathVariable Long id) {
    repository.deleteById(id);
    return "redirect:/";
}
}

```

## 💻 Templates HTML (Thymeleaf)

👉 Coloque estes arquivos em src/main/resources/templates/

---

### index.html

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Lista de Pessoas</title>
    <link rel="stylesheet" th:href="@{/css/style.css}">
</head>
<body>
    <h1>Pessoas</h1>
    <a href="/nova">Adicionar Pessoa</a>
    <table border="1">
        <tr>
            <th>ID</th><th>Nome</th><th>Idade</th><th>Cargo</th><th>Ações</th>
        </tr>

```

```

<tr th:each="pessoa : ${pessoas}">
    <td th:text="${pessoa.id}"></td>
    <td th:text="${pessoa.nome}"></td>
    <td th:text="${pessoa.idade}"></td>
    <td th:text="${pessoa.cargo}"></td>
    <td>
        <a th:href="@{/editar/{id}(id=${pessoa.id})}">Editar</a> |
        <a th:href="@{/deletar/{id}(id=${pessoa.id})}">Deletar</a>
    </td>
</tr>
</table>
</body>
</html>

```

### **form.html**

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Nova Pessoa</title>
    <link rel="stylesheet" th:href="@{/css/style.css}">
</head>
<body>
    <h1>Nova Pessoa</h1>
    <form th:action="@{/salvar}" method="post">
        Nome: <input type="text" name="nome" /><br/>
        Idade: <input type="number" name="idade" /><br/>
        Cargo: <input type="text" name="cargo" /><br/>
        <button type="submit">Salvar</button>
    </form>
    <a href="/">Voltar</a>
</body>
</html>

```

### **edit.html**

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Editar Pessoa</title>
    <link rel="stylesheet" th:href="@{/css/style.css}">
</head>
<body>
    <h1>Editar Pessoa</h1>
    <form th:action="@{/atualizar/{id}(id=${pessoa.id})}" method="post">
        Nome: <input type="text" name="nome" th:value="${pessoa.nome}" /><br/>
        Idade: <input type="number" name="idade" th:value="${pessoa.idade}" /><br/>
        Cargo: <input type="text" name="cargo" th:value="${pessoa.cargo}" /><br/>
        <button type="submit">Atualizar</button>
    </form>
    <a href="/">Voltar</a>

```

```
</body>  
</html>
```

 **O Spring Boot serve automaticamente tudo que estiver em:**

- **src/main/resources/static/ → para arquivos estáticos como CSS, JS, imagens.**
- 

 **Exemplo de arquivo CSS**

**style.css**

```
body {  
    font-family: Arial, sans-serif;  
    background-color: #f5f5f5;  
    margin: 20px;  
}  
  
h1 {  
    color: #333;  
}  
  
table {  
    width: 100%;  
    border-collapse: collapse;  
    margin-top: 20px;  
}  
  
table, th, td {  
    border: 1px solid #ddd;  
}  
  
th, td {  
    padding: 10px;  
    text-align: left;  
}  
  
button {  
    background-color: #4CAF50;  
    color: white;  
    border: none;  
    padding: 8px 12px;  
    cursor: pointer;  
}  
  
button:hover {  
    background-color: #45a049;  
}
```

 **Rodando no VSCode**

**1. No terminal:** (onde estiver o arquivo “mvnw.cmd” digitar via opção (Abrir em terminal integrado)):

**mvnw.cmd spring-boot:run**

**2. Acesse:**

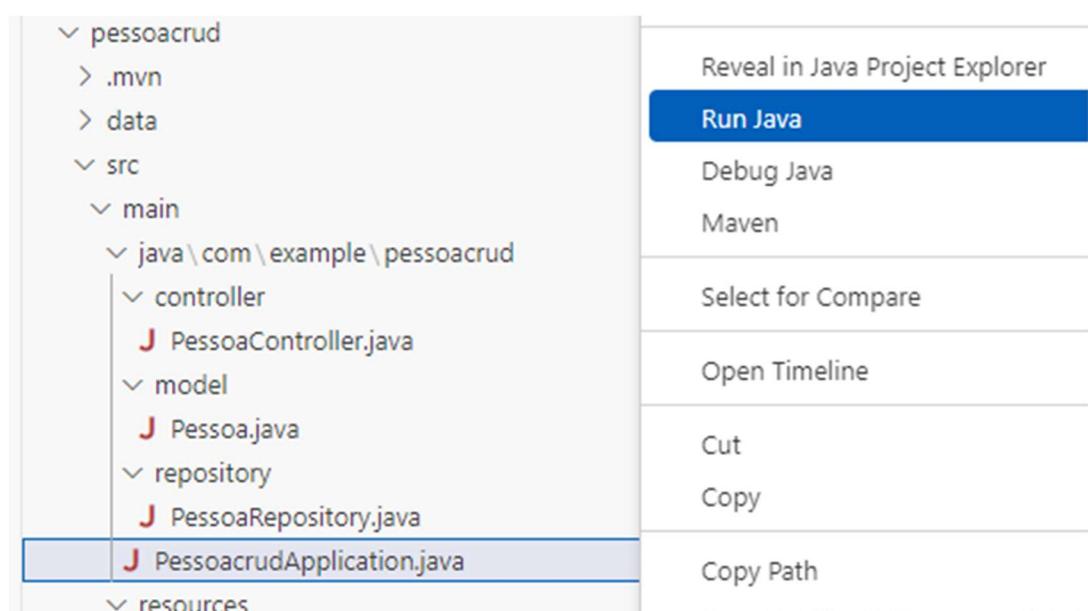
- **App:** <http://localhost:8080>
- **H2 Console:** <http://localhost:8080/h2-console> **(ANEXO B)**  
**Use:**

JDBC URL: jdbc:h2:file:./data/pessoa-db

User: sa

Password: (em branco)

Se preferir, também pode abrir o projeto no **VSCode**, abrir o painel de extensões, instalar a extensão **Spring Boot Extension Pack** e usar o botão **"Run"** que aparece em cima do método main() no PessoacrudApplication.java.



## CARREGAMENTO VIA VSCode:

A screenshot of the VSCode interface showing the 'PessoacrudApplication.java' file open in the editor. The code defines a main method that starts a Spring Boot application. The terminal at the bottom shows the application's startup logs, including database connection details and autowiring information. The logs indicate that the H2 console is available at '/h2-console', the database is available at 'jdbc:h2:file:./data/pessoa-db', and the application is started on port 8080. The logs also show the autowiring of beans such as 'PessoaController', 'Pessoa', 'PessoaRepository', and 'PessoacrudApplication'. The logs conclude with the completion of initialization in 2 ms and a Hibernate query for selecting data from the 'pessoa' table.

## Visão pelo navegador (READ):

localhost:8080

## Pessoas

[Adicionar Pessoa](#)

ID	Nome	Idade	Cargo	Ações
1	Cristiano Almeida	49	Desenvolvedor back end	<a href="#">Editar   Deletar</a>
2	Katia Flávia	42	Desenvolvedor front end	<a href="#">Editar   Deletar</a>
35	Maria D. Barbosa	35	Analista Senior	<a href="#">Editar   Deletar</a>

### CREATE:

Nova Pessoa

localhost:8080/nova

## Nova Pessoa

Nome:

Idade:

Cargo:

[Salvar](#)

[Voltar](#)

### UPDATE:

Editar Pessoa

localhost:8080/editar/2

## Editar Pessoa

Nome:

Idade:

Cargo:

[Atualizar](#)

[Voltar](#)

### DELETE:

## Pessoas

[Adicionar Pessoa](#)

ID	Nome	Idade	Cargo	Ações
1	Cristiano Almeida	49	Desenvolvedor back end	<a href="#">Editar   Deletar</a>
2	Katia Flávia	42	Desenvolvedor front end	<a href="#">Editar   Deletar</a>
35	Maria D. Barbosa	35	Analista Senior	<a href="#">Editar   Deletar</a>

localhost:8080/deletar/35

## ANEXO A:

The screenshot shows the Spring Initializer web interface at [start.spring.io](http://start.spring.io). The configuration is set for a Maven-based Spring Boot project using Java 17. The 'Project' section includes Gradle and Kotlin options. The 'Language' section is set to Java. Under 'Spring Boot', version 3.4.5 is selected. The 'Project Metadata' section shows Group as com.example, Artifact as pessoadcrud, and Name as pessoadcrud. The 'Dependencies' section includes Spring Web (selected), Spring Data JPA, and H2 Database. The 'Thymeleaf' section is also visible. At the bottom, there are 'GENERATE' and 'EXPLORE' buttons.

## ANEXO B: <http://localhost:8080/h2-console>

Português (Brasil) Preferências Tools Ajuda

### Login

Configuração ativa: Generic H2 (Embedded)

Nome da configuração: Generic H2 (Embedded) Gravar Remover

Classe com o driver: org.h2.Driver

JDBC URL: jdbc:h2:file:/data/pessoa-db

Usuário: sa

Senha:

**Conectar Testar conexão**

H2 - Inicial Pesquisar guias

localhost:8080/h2-console/login.do?sessionid=6182aa3699a979443f473cb832143cb8

Número máximo de linhas: 1000 | Auto commit | Executar comando | Executar selecionado | Auto complete | Limpar | Comando SQL: SELECT \* FROM PESSOA

PESSOA INFORMATION\_SCHEMA Usuários H2 2.3.232 (2024-08-11)

**Comandos importantes**

?	Mostrar esta página de ajuda
!	Mostrar o histórico de comandos
Ctrl+Enter	Executar o comando SQL corrente
Shift+Enter	Executes the SQL statement defined by the text selection
Ctrl+Space	Auto complete
Esc	Fechar conexão à Base de Dados

H2 Terminal

localhost:8080/h2-console/login.do?sessionid=6182aa3699a979443f473cb832143cb8

Número máximo de linhas: 1000 | Auto commit | Número máximo de linhas: 1000 | Auto complete | Desligado | Auto seleção | Ligado | ?

jdbc:h2:file:./data/pessoa-db | Executar comando | Executar selecionado | Auto complete | Limpar | Comando SQL:

SELECT \* FROM PESSOA|

PESSOA | INFORMATION\_SCHEMA | Usuários | H2 2.3.232 (2024-08-11)

SELECT \* FROM PESSOA;

ID	CARGO	IDADE	NOME
1	Desenvolvedor back end	49	Cristiano Almeida
2	Desenvolvedor front end	42	Katia Flávia
35	Analista Senior	35	Maria D. Barbosa

(3 linhas, 25 ms)

Alterar