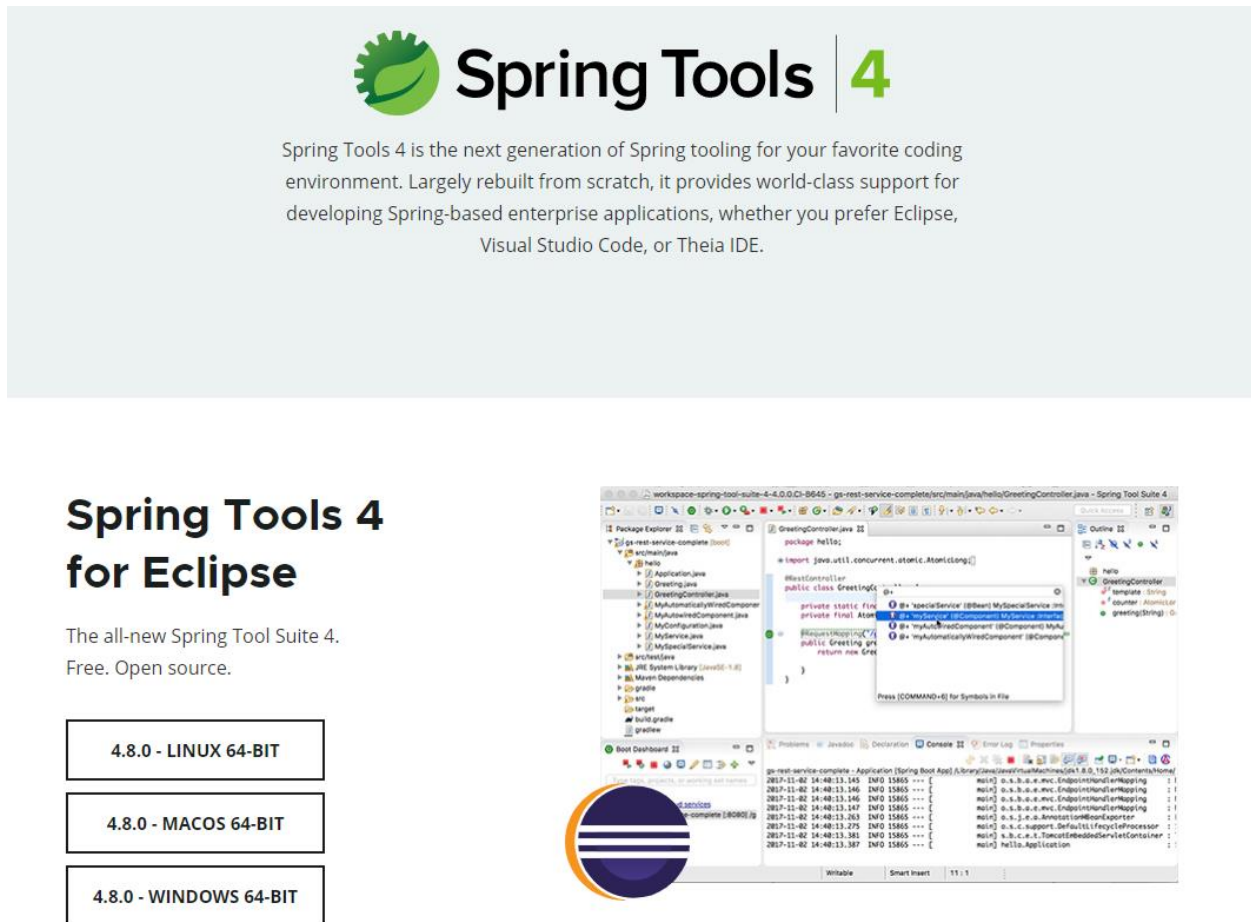


## API CLIENTES

Vamos utilizar a IDE Spring Tools, versão 4, criada pelo grupo Spring.

Para isso, acesse o link [spring.io/tools](https://spring.io/tools)

Será apresentada uma página semelhante à indicada:



The image shows a promotional banner for Spring Tools 4. The banner features the Spring logo (a green gear with a leaf) and the text "Spring Tools | 4". Below this, it states: "Spring Tools 4 is the next generation of Spring tooling for your favorite coding environment. Largely rebuilt from scratch, it provides world-class support for developing Spring-based enterprise applications, whether you prefer Eclipse, Visual Studio Code, or Theia IDE." Below the banner is a screenshot of the Spring Tools 4 for Eclipse download page. The page has the heading "Spring Tools 4 for Eclipse" and the subtext "The all-new Spring Tool Suite 4. Free. Open source." There are three download buttons: "4.8.0 - LINUX 64-BIT", "4.8.0 - MACOS 64-BIT", and "4.8.0 - WINDOWS 64-BIT". To the right of the buttons is a screenshot of the Spring Tools 4 IDE interface, showing a code editor with a Java file, a Package Explorer on the left, and a Console window at the bottom.

Spring Tools é uma IDE baseada no Eclipse que oferece recursos adequados para o desenvolvimento de aplicações Web com diversos recursos Spring para a plataforma Java.

Selecione a plataforma desejada e efetue o download

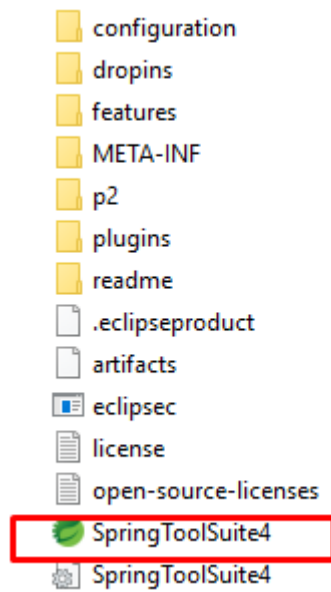
Descompacte o arquivo disponível.

Um dos recursos oferecidos pelo Spring Tools é o Spring Boot que permite facilitar o desenvolvimento de aplicações web com Java.

Uma dessas facilidades é a utilização do Maven, que é uma ferramenta que fornece aos desenvolvedores uma forma de automatizar e padronizar a construção e publicação de aplicações web Java com o uso de um arquivo no formato xml, denominado de pom.xml. Esse arquivo permite definir as bibliotecas para atender as necessidades de uma determinada aplicação.

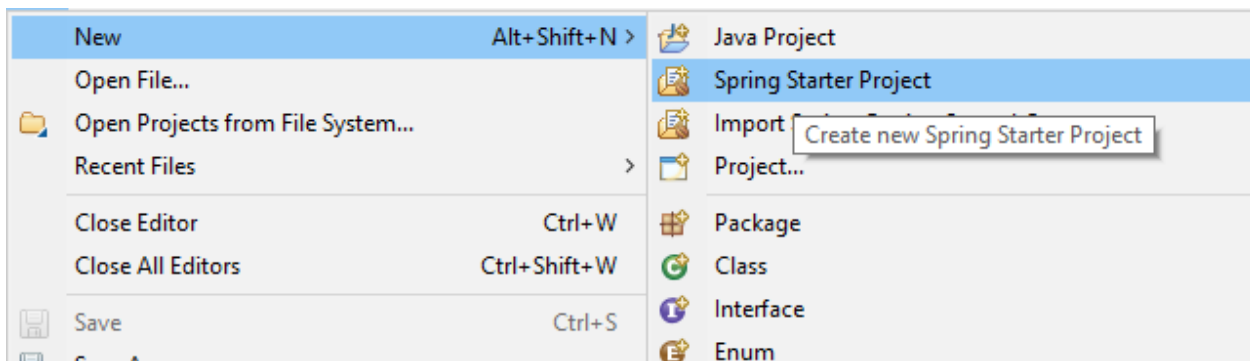
Assim, ao se criar um projeto com o Spring Boot, o desenvolvedor seleciona os recursos necessários, os quais incorporarão diversos arquivos (bibliotecas) com versões adequadas. Essas dependências são acessadas em locais específicos na Internet e baixadas diretamente para a aplicação, ficando referenciadas no arquivo pom.xml. Qualquer mudança em uma dessas dependências no arquivo pom.xml, a aplicação é automaticamente atualizada. Assim, o Spring Boot minimiza o trabalho do desenvolvedor, pois antes dele um desenvolvedor tinha que pesquisar cada biblioteca na Internet, efetuar seu download, inseri-la no projeto e analisar se é ou não compatível com as demais bibliotecas existentes no projeto.

Após a descompactação do arquivo do Spring Tools, será criada a pasta com a estrutura dessa aplicação:

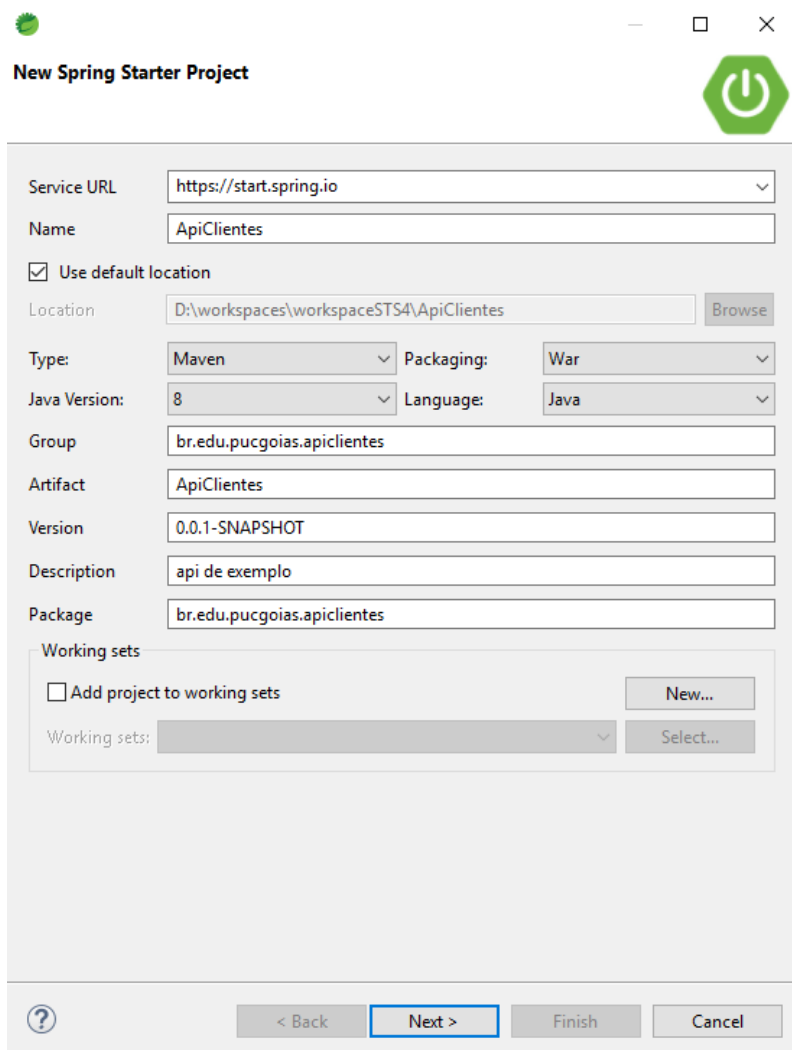


Acesse o programa em destaque na figura anterior.

Para criar um projeto, selecione File / New, como indicado:



Será apresentada a seguinte janela:



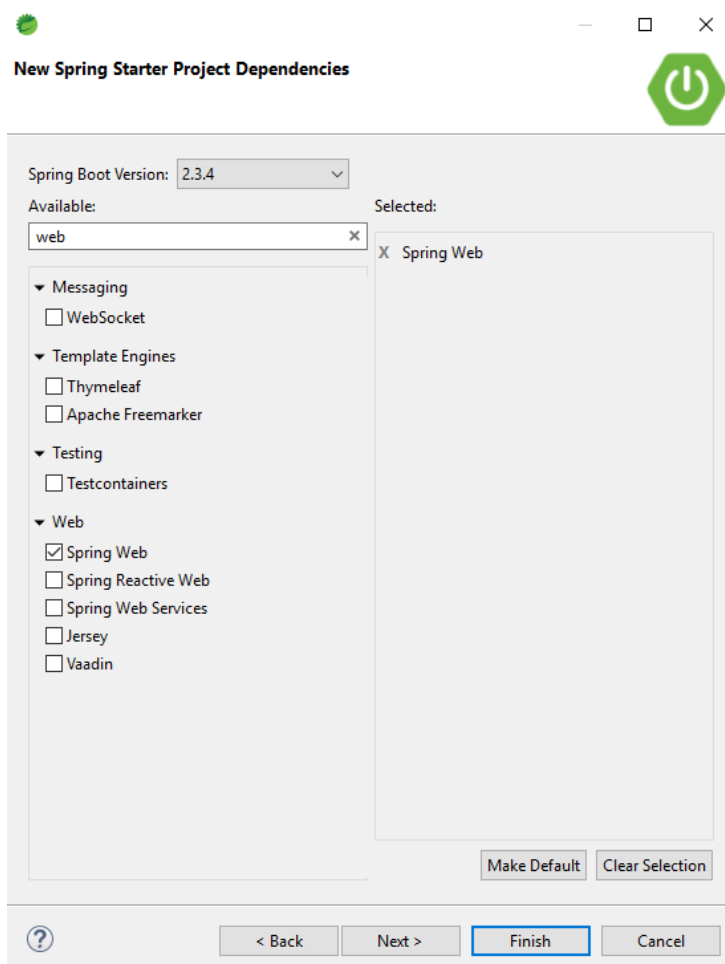
The image shows a 'New Spring Starter Project' dialog box. At the top left is a green circular icon with a white gear. At the top right are standard window controls (minimize, maximize, close) and a green hexagonal icon with a white power symbol. The dialog contains the following fields and controls:

- Service URL:** A dropdown menu with 'https://start.spring.io' selected.
- Name:** A text field containing 'ApiClientes'.
- Use default location:** A checked checkbox.
- Location:** A text field containing 'D:\workspaces\workspaceSTS4\ApiClientes' and a 'Browse' button.
- Type:** A dropdown menu with 'Maven' selected.
- Packaging:** A dropdown menu with 'War' selected.
- Java Version:** A dropdown menu with '8' selected.
- Language:** A dropdown menu with 'Java' selected.
- Group:** A text field containing 'br.edu.pucgoias.apiclientes'.
- Artifact:** A text field containing 'ApiClientes'.
- Version:** A text field containing '0.0.1-SNAPSHOT'.
- Description:** A text field containing 'api de exemplo'.
- Package:** A text field containing 'br.edu.pucgoias.apiclientes'.
- Working sets:** A section containing:
  - A checkbox labeled 'Add project to working sets'.
  - A 'New...' button.
  - A 'Working sets:' dropdown menu.
  - A 'Select...' button.

At the bottom, there is a row of buttons: a help icon (?), '< Back', 'Next >' (highlighted with a blue border), 'Finish', and 'Cancel'.

Crie o projeto ApiClientes preenchendo e selecionando informações como está descrito na figura anterior

Confirme em Next.

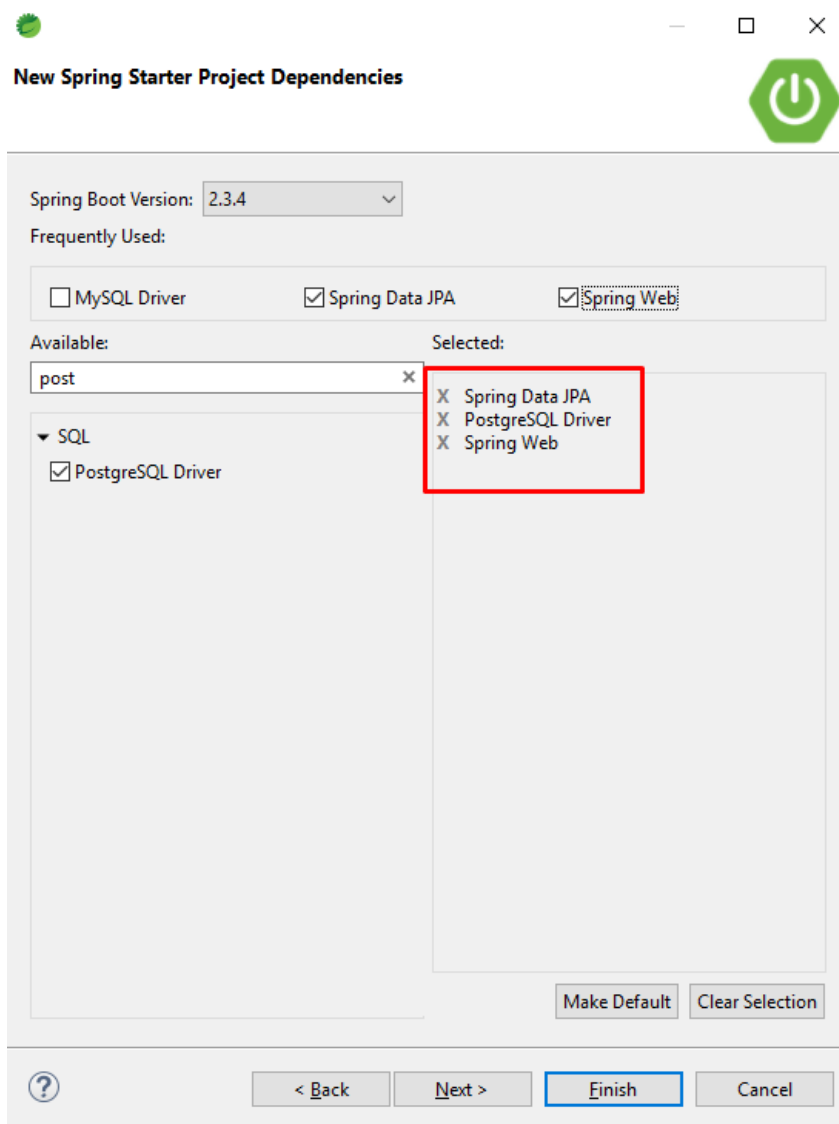


A figura anterior permite seleccionar as dependências que serão utilizadas na aplicação.

Basta informar no campo Available palavras para pesquisa.

No exemplo da figura anterior, informou-se web.

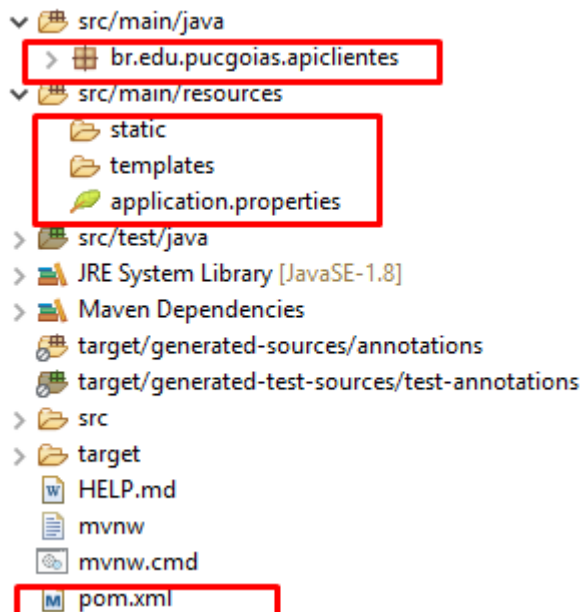
Selecione Spring web, deixando essa opção em destaque no lado direito da janela.



Selecione as demais opções: Spring Data JPA e PostgreSQL Driver, como está destacado na figura anterior. Essas serão as dependências que serão utilizadas pela aplicação.

finish

O projeto será criado com a seguinte estrutura:



O pacote criado possui dois arquivos da aplicação:

- > ApiClientesApplication.java
- > ServletInitializer.java

A classe ApiClientesApplication.java é o programa responsável pela execução da aplicação.

As pastas static e templates ficam dentro do pacote resources da aplicação.

É na pasta static onde se armazena os arquivos css, javascript, imagens e outros.

É na pasta templates onde se armazena as páginas HTML da aplicação, quando esta possui front-end.

O arquivo application.properties armazena configurações da aplicação, principalmente as configurações de acesso ao banco de dados, o qual será ajustado posteriormente para a conexão com o banco de dados da aplicação.

Em destaque na figura está o arquivo pom.xml

Com a criação do projeto, o arquivo pom.xml é gerado com um conteúdo que possui as dependências criadas e as informações que foram introduzidas para a criação do projeto. Observe a sua estrutura a seguir:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>2.2.6.RELEASE</version>
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>br.edu.pucgoias.apiclientes</groupId>
12  <artifactId>ApiClientes</artifactId>
13  <version>0.0.1-SNAPSHOT</version>
14  <packaging>war</packaging>
15  <name>ApiClientes</name>
16  <description>api de exemplo</description>
17
18  <properties>
19    <java.version>1.8</java.version>
20  </properties>
21
```

```

22- <dependencies>
23-   <dependency>
24-     <groupId>org.springframework.boot</groupId>
25-     <artifactId>spring-boot-starter-data-jpa</artifactId>
26-   </dependency>
27-   <dependency>
28-     <groupId>org.springframework.boot</groupId>
29-     <artifactId>spring-boot-starter-web</artifactId>
30-   </dependency>
31-
32-   <dependency>
33-     <groupId>org.postgresql</groupId>
34-     <artifactId>postgresql</artifactId>
35-     <scope>runtime</scope>
36-   </dependency>
37-   <dependency>
38-     <groupId>org.springframework.boot</groupId>
39-     <artifactId>spring-boot-starter-tomcat</artifactId>
40-     <scope>provided</scope>
41-   </dependency>
42-   <dependency>
43-     <groupId>org.springframework.boot</groupId>
44-     <artifactId>spring-boot-starter-test</artifactId>
45-     <scope>test</scope>
46-     <exclusions>
47-       <exclusion>
48-         <groupId>org.junit.vintage</groupId>
49-         <artifactId>junit-vintage-engine</artifactId>
50-       </exclusion>
51-     </exclusions>
52-   </dependency>
53- </dependencies>
54-
55- <build>
56-   <plugins>
57-     <plugin>
58-       <groupId>org.springframework.boot</groupId>
59-       <artifactId>spring-boot-maven-plugin</artifactId>
60-     </plugin>
61-   </plugins>
62- </build>
63-
64- </project>

```

Observe que a versão do Spring Boot está em destaque, pois foi alterada para 2.2.6. Assim, faça essa alteração no arquivo pom.xml do seu projeto. Vamos trabalhar com uma versão um pouco inferior do que a atual.

Abra o arquivo application.properties e ajuste o seu conteúdo para o indicado a seguir:

```

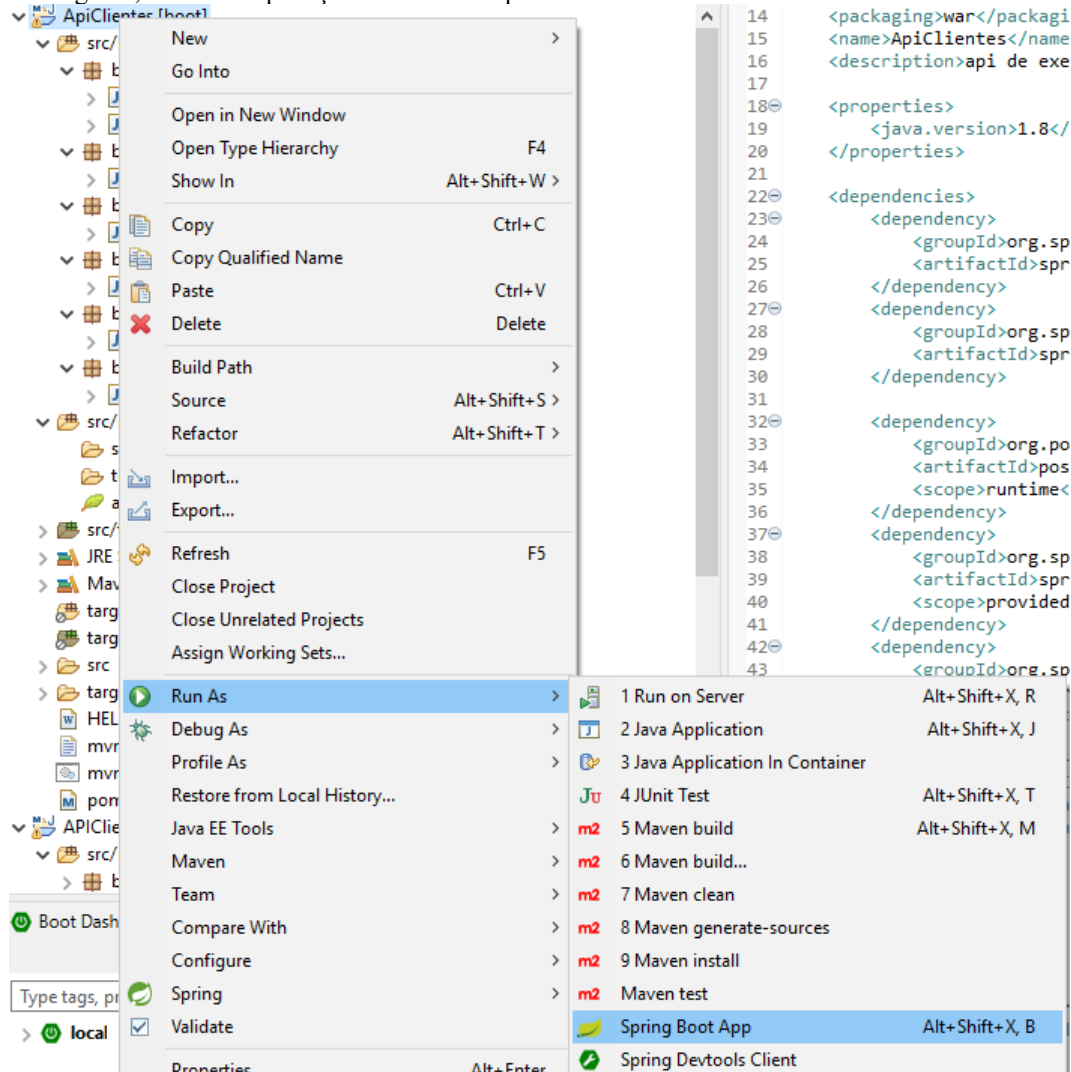
1  server.port=${port:8282}
2
3  spring.datasource.url=jdbc:postgresql://localhost:5432/clientes?createDatabaseIfNotExist=true&useSSL=false
4  spring.datasource.username=postgres
5  spring.datasource.password=postgres
6  spring.datasource.driver-class-name=org.postgresql.Driver
7
8  # =====
9  # = JPA / HIBERNATE
10 # =====
11 # mostra ou não o log de cada script sql
12
13 spring.jpa.show-sql=true
14
15 # Hibernate ddl auto (create, create-drop, update): with "create-drop"
16 # o esquema do banco de dados será automaticamente criado para iniciar a aplicação
17
18 spring.jpa.hibernate.ddl-auto=update
19
20 # permite o Hibernate gerar SQL otimizado
21
22 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

```

Note que se destaca a configuração de um banco de dados PostgreSQL. Cada uma das linhas em destaque possuem uma sintética explicação.

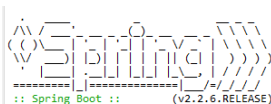
Crie o banco de dados no PostgreSQL.

Em seguida, execute a aplicação utilizando os passos:



The screenshot shows an IDE interface with three main panels. On the left is the 'Project Explorer' showing a project named 'ApiClientes' with a 'src' directory containing 'main' and 'test' subdirectories. The middle panel is the 'Code Editor' showing a Java file with a 'main' method. The right panel is the 'Run' configuration menu, which is open and shows a list of run configurations. The 'Run As' option is selected, and a sub-menu is open showing various options: '1 Run on Server', '2 Java Application', '3 Java Application In Container', '4 JUnit Test', '5 Maven build', '6 Maven build...', '7 Maven clean', '8 Maven generate-sources', '9 Maven install', 'Maven test', 'Spring Boot App', and 'Spring Devtools Client'. The 'Spring Boot App' option is highlighted.





```
2020-10-24 05:53:44.354 INFO 12824 --- [main] b.e.p.a.ApiClientesApplication : Starting ApiClientesApplication on DESKTOP-SFV614G with PID 12824 (D:\workspaces\workspaceST54\
2020-10-24 05:53:44.358 INFO 12824 --- [main] b.e.p.a.ApiClientesApplication : No active profile set, falling back to default profiles: default
2020-10-24 05:53:45.040 INFO 12824 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2020-10-24 05:53:45.097 INFO 12824 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 49ms. Found 1 JPA repository interfaces.
2020-10-24 05:53:45.618 INFO 12824 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8282 (http)
2020-10-24 05:53:45.629 INFO 12824 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-10-24 05:53:45.630 INFO 12824 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.33]
2020-10-24 05:53:45.810 INFO 12824 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-10-24 05:53:45.811 INFO 12824 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1404 ms
2020-10-24 05:53:45.987 INFO 12824 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH0000204: Processing PersistenceUnitInfo [name: default]
2020-10-24 05:53:46.044 INFO 12824 --- [main] org.hibernate.Version : HHH0000412: Hibernate ORM core version 5.4.12.Final
2020-10-24 05:53:46.142 INFO 12824 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2020-10-24 05:53:46.226 INFO 12824 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2020-10-24 05:53:46.345 INFO 12824 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2020-10-24 05:53:46.359 INFO 12824 --- [main] org.hibernate.dialect.Dialect : HHH0000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
2020-10-24 05:53:47.868 INFO 12824 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH0000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.int
2020-10-24 05:53:47.876 INFO 12824 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2020-10-24 05:53:48.437 WARN 12824 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed dur
2020-10-24 05:53:48.617 INFO 12824 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-10-24 05:53:48.897 INFO 12824 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8282 (http) with context path ''
2020-10-24 05:53:48.900 INFO 12824 --- [main] b.e.p.a.ApiClientesApplication : Started ApiClientesApplication in 4.891 seconds (JVM running for 5.799)
```

A aplicação “subiu” sem problemas, pois não apresentou nenhum erro. O Spring Boot destaca alguns recursos que foram ativados quando a aplicação foi ativada.

crie o pacote `br.edu.pucgoias.apiclientes.model`

crie a classe `Cliente.java`

```
1 package com.algaworks.osworks.domain.model;
2
3 import javax.persistence.Column;
4
11 @Entity
12 @Table(name = "tb_cliente")
13 public class Cliente {
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     @Column(name = "codigo")
18     private Long id;
19
20     @NotBlank
21     @Size(max = 60)
22     private String nome;
23
24     @NotBlank
25     @Email
26     @Size(max = 255)
27     private String email;
28
29     @NotBlank
30     @Size(max = 20)
31     @Column(name = "fone")
32     private String telefone;
33
34     public Long getId() {
35         return id;
36     }
37
38     public void setId(Long id) {
39         this.id = id;
40     }
41
42     public String getNome() {
43         return nome;
44     }
45
46     public void setNome(String nome) {
47         this.nome = nome;
48     }
49
50     public String getEmail() {
51         return email;
52     }
53
54     public void setEmail(String email) {
55         this.email = email;
56     }
57 }
```

```

58
59 public String getTelefone() {
60     return telefone;
61 }
62
63 public void setTelefone(String telefone) {
64     this.telefone = telefone;
65 }
66
67 @Override
68 public int hashCode() {
69     final int prime = 31;
70     int result = 1;
71     result = prime * result + ((id == null) ? 0 : id.hashCode());
72     return result;
73 }
74
75 @Override
76 public boolean equals(Object obj) {
77     if (this == obj)
78         return true;
79     if (obj == null)
80         return false;
81     if (getClass() != obj.getClass())
82         return false;
83     Cliente other = (Cliente) obj;
84     if (id == null) {
85         if (other.id != null)
86             return false;
87     } else if (!id.equals(other.id))
88         return false;
89     return true;
90 }
91
92 }

```

@Entity é uma annotation que permite criar uma tabela no banco de dados com as informações da classe Cliente. Opcionalmente, pode-se utilizar a annotation @Table para informar um nome para a tabela correspondente no banco de dados.

A annotation @Id estabelece que o referido atributo será uma chave primária.

@GeneratedValue permite configurar como a chave primária será criada. No exemplo, utiliza-se ENTITY, indicando que será gerada automaticamente pelo banco de dados. Essa é a melhor opção entre as disponíveis.

@Column permite referenciar o nome da coluna, caso o nome do atributo seja diferente.

@NotBlank estabelece que a String é válida desde que não seja nula e seu comprimento seja maior que zero.

Há a opção: @NotBlank(message="Nome deve ser informado")

Há também o @NotEmpty e @NotNull.

@Size permite estabelecer o tamanho máximo de uma determinada String. Opcionalmente, pode-se ter algo do tipo:

@Size(min=4, max=7)

HashCode() e Equals() são procedimentos que são criados pela opção source do menu do Eclipse, como são criados os métodos get/set. HasCode permite que um determinado código hashing seja criado na tentativa de se criar um identificador único para um objeto. O Equals permite fazer comparações entre dois objetos se ambos são iguais, incluindo todos os seus atributos. São mais utilizados em collections.

crie o pacote br.edu.pucgoias.apiclientes.repository

crie a interface ClienteRepository.java

```

1 package br.edu.pucgoias.apiclientes.repository;
2
3
4 import java.util.List;
5
6 import org.springframework.data.jpa.repository.JpaRepository;
7 import org.springframework.stereotype.Repository;
8
9 import br.edu.pucgoias.apiclientes.model.Cliente;
10
11
12 @Repository
13 public interface ClienteRepository extends JpaRepository<Cliente, Long> {
14
15     List<Cliente> findByName(String nome);
16     List<Cliente> findByNameContaining(String nome);
17     Cliente findByEmail(String email);
18
19 }

```