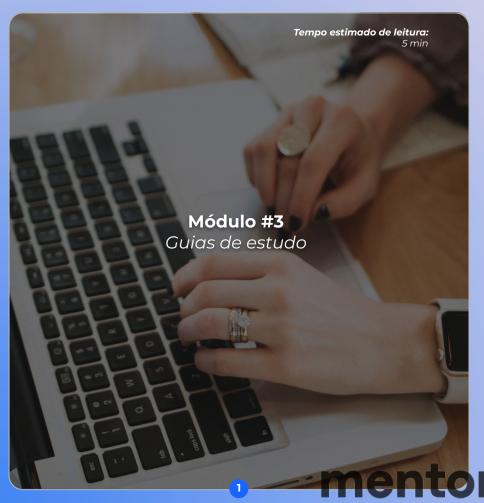
mentorama.



Introdução

No módulo 3 estudamos sobre como podemos empacotar nossos projetos Java para distribuí-los.

Dessa forma poderemos fazer com que outras pessoas sejam capazes de executar nossos projetos e também acessar programas web que nós desenvolvemos

Build

Quando falamos em **build** ou "**construção**" nos referimos ao processo de transformar o código em um arquivo executável que pode ser aberto em qualquer sistema operacional que ofereça suporte.

Em Java, as aplicações podem ser construídas para os formatos **JAR** e **WAR**







mentorama.



JAR

Java **Ar**chive é um arquivo compactado que contém todas as classes e demais arquivos necessários para um programa Java ser executado.

WAR

Web Application **Ar**chive é semelhante ao arquivos *JAR*, no entanto usado para empacotamento de aplicações web que contém arquivos HTML e demais recursos usados para as aplicações.

Dependências

Dependências se referem a códigos prontos que podem ser adicionados aos projetos.

Esses códigos são chamados de bibliotecas e elas podem ser usadas para estender um projeto com funcionalidades criadas por outros programadores. Podemos adicionar dependências aos nossos projetos Java de duas formas:

- Manualmente, por meio de referências explícitas aos pacotes JAR de bibliotecas que baixamos;
- De forma automatizada usando um gerenciador de dependências, como o Maven (https://maven.apache.org/) ou Gradle (https://gradle.org/)

Maven

O **Maven** é o mais conhecido gerenciador de dependências e automatizador do processo de **build** de projetos Java. Ele tem 4 principais objetivos:

- Tornar o processo de build fácil
- Prover um sistema de build uniforme
- Prover informações de qualidade do projeto
- Encorajar melhores práticas de desenvolvimento

3





mentorama.

O repositório de dependências do Maven (https://mvnrepository.com/) é muito abrangente e possui mais de 30 milhões de bibliotecas que podem ser adicionadas em nossos projetos.

Como exemplo de uma bibliotecas há o Gson, uma biblioteca do Google para parsing de JSON:

https://mvnrepository.com/artifact/com.google.code.gson/gson

Arquivo pom.xml

O arquivo pom.xml (Project Object Model) é o principal arquivo que define que o projeto Java tem suporte ao Maven.

Ele define as configurações do projeto como versão do Java, nome do projeto, autor, dependências, plugins e especificações de build.

O arquivo sempre ficará na raiz do projeto Java e o Maven sempre irá buscar por esse arquivo quando alguma tarefa for realizada. mentoramo

```
<?xml version="1.0" encoding="UTF-8"?>
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot
    <artifactId>spring-boot-starter-parent</artifactId>
     <version>2.7.7-SNAPSHOT
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>br.com.mentorama
  <artifactId>alunos-api</artifactId>
  <version>0.0.1-SNAPSHOT
 <name>alunos-api</name>
  <description>Demo project for Spring Boot</description>
  properties>
    <iava.version>17</java.version>
  </properties>
  <dependencies>
     <dependency>
       <groupId>org.springframework.boot
       <artifactId>spring-boot-starter-web</artifactId>
     </dependency>
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-test</artifactId>
       <scope>test</scope>
    </dependency>
  </dependencies>
```



No arquivo mencionado anteriormente, especificamos algumas propriedades:

- modelVersion: define a versão do formato do arquivo pom.xml;
- parent: define que o projeto em questão irá herdar as propriedades do projeto "pai", que no caso é o "spring-boot-starter-parent";
- groupld: identifica a organização responsável pelo projeto. Por convenção se utiliza o domínio da organização invertido (mentorama.com.br ficará br.com.mentorama);
- artifactId: é o identificador do projeto dentro do grupo (ou organização);
- version: versão do projeto;
- name: o nome do projeto;
- description: uma breve descrição sobre o projeto;
- properties: define algumas propriedades do projeto usadas na compilação, como a versão do Java;
- dependencies: define quais dependências serão usadas no projeto. Podemos adicionar dependências a partir do MVNRepository (https://mvnrepository.com/)

Repositórios remotos

Podemos especificar qual será o repositório usado para o Maven baixar as dependências. Esse cenário é comum em empresas que controlam quais dependências podem ser usadas em seus projetos internos. Para fazer isso, basta adicionar a tag "**repositories**" e especificar os endereços de cada repositório disponível: