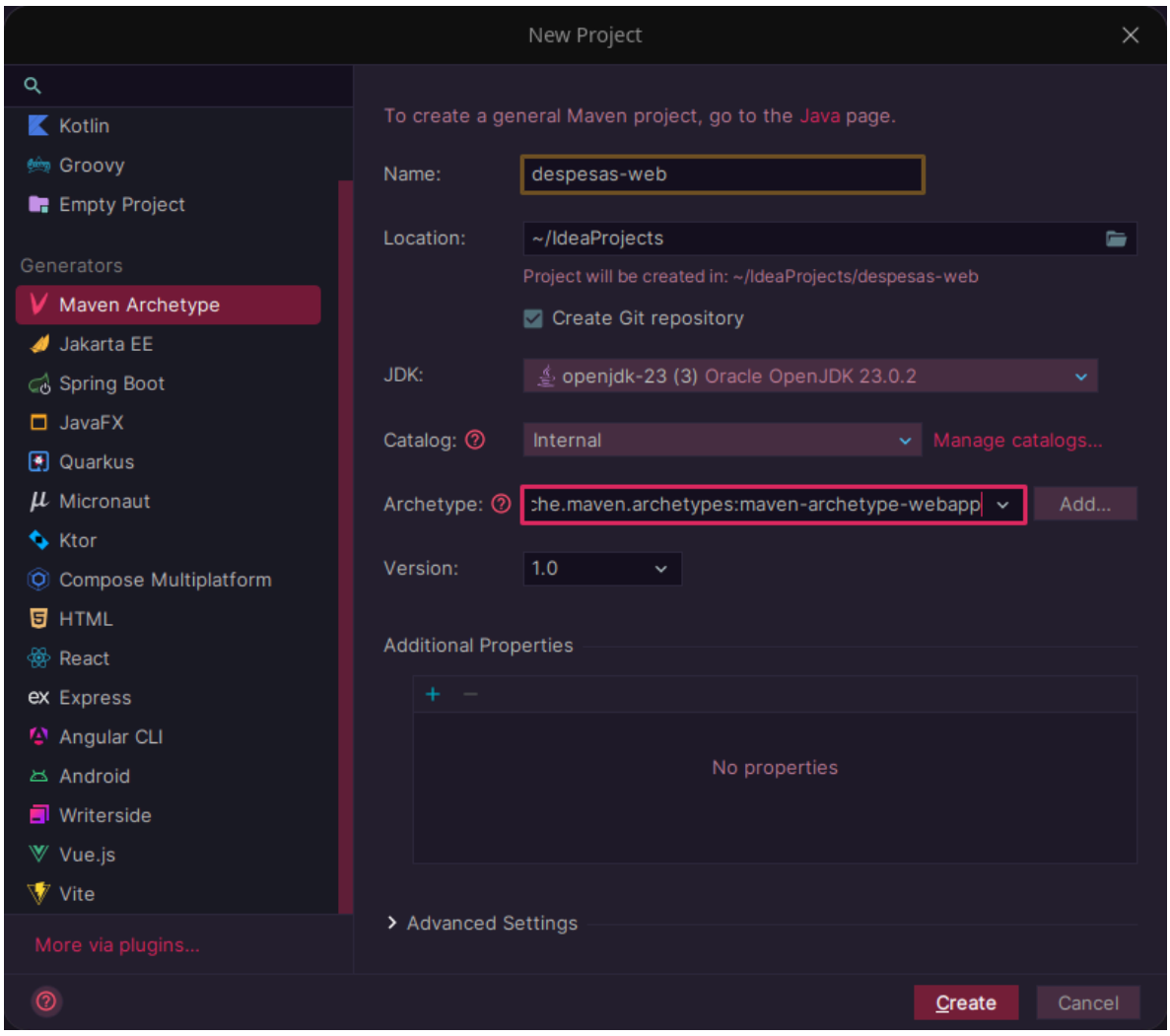
Table of Contents

Learn Servlet - Module 1

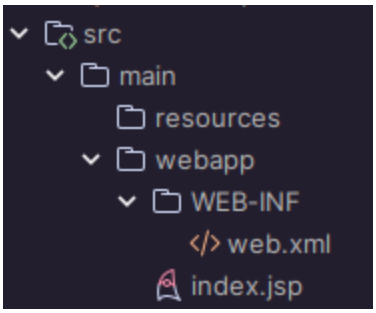
Criando o projeto Maven Archetype no IntelliJ



Tela de criação de projeto Maven no IntelliJ, mostrando a seleção do archetype e outras configurações.

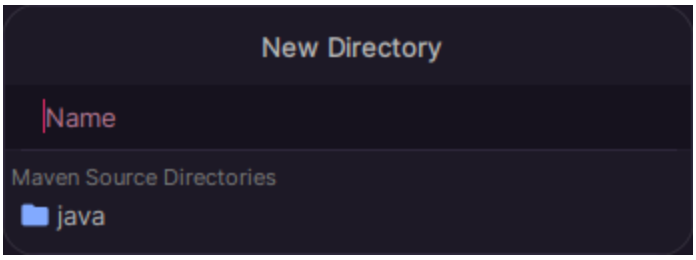
Estrutura de Pastas Criada

A estrutura padrão de pastas criada pelo Maven Archetype para um web app inclui a pasta `WEB-INF`, onde ficam as configurações do aplicativo web.



Estrutura de pastas padrão do Maven Archetype, destacando a pasta WEB-INF.

No entanto, ainda precisamos criar uma pasta adicional para armazenar o código da aplicação. Para isso, clique com o `BOTÃO` + `DIREITO` + `DO` + `MOUSE` na pasta `main`, selecione `New > Directory` e digite `java`. Provavelmente, aparecerá um popup para inserir o nome da pasta, e a IDE pode sugerir automaticamente o nome correto. Pressione Enter para confirmar.



Criação da pasta 'java' dentro da estrutura do projeto.

Entendendo a estrutura

O Maven Archetype para um web app cria uma estrutura de pastas organizada para facilitar o desenvolvimento. Aqui está uma explicação das principais pastas:

- **src/main/java**: Onde você coloca o código-fonte Java da aplicação. Esta pasta é criada manualmente, como descrito anteriormente.
- **src/main/resources**: Armazena recursos como arquivos de configuração que são usados pelo código Java.
- **src/main/webapp**: Contém os arquivos da aplicação web, como páginas JSP, arquivos HTML, CSS, JavaScript, etc.
- **src/main/webapp/WEB-INF**: Esta pasta é crucial para a configuração do aplicativo web. Ela contém:

    - **web.xml**: O arquivo de configuração do servlet, onde você define servlets, filtros, listeners, etc.
    - Outras configurações específicas do servidor ou da aplicação.
- **src/test/java**: Onde você coloca os testes unitários para o seu código Java.
- **pom.xml**: O arquivo de configuração do Maven, onde você define as dependências, plugins, e outras configurações do projeto.

Essa estrutura ajuda a separar o código-fonte, recursos, e configurações, tornando o projeto mais organizado e fácil de gerenciar.

```
src
├── main
│   ├── java
│   ├── resources
│   └── webapp
│       └── WEB-INF
│           └── web.xml
└── test
    └── java
pom.xml
```

Tudo precisa de um servidor

Para fazermos o nosso **web app** em java funcionar é preciso que utilizemos um servidor, existem dezenas de servidores, eles variam em seus tipos como mulheres variam, tem as mais morenas as mais loiras, etc.

Neste caso eles variam dois aspectos - para nossa abordagem iremos utilizar somente esses dois aspectos:

- Tem os servidores mais gordinhas, aquelas que vem com gordura a mais, ou seja, funções do Java EE a mais, coisa que só é preciso normalmente quando tu quer algo mais robusto
- Tem os mais slim, que são os aquelas que se adequa ao que queremos tendo em vista que não possuem todas as dezenas de coisas que o Java EE oferece, ou seja são

menos robustos

Vamos utilizar o Jet

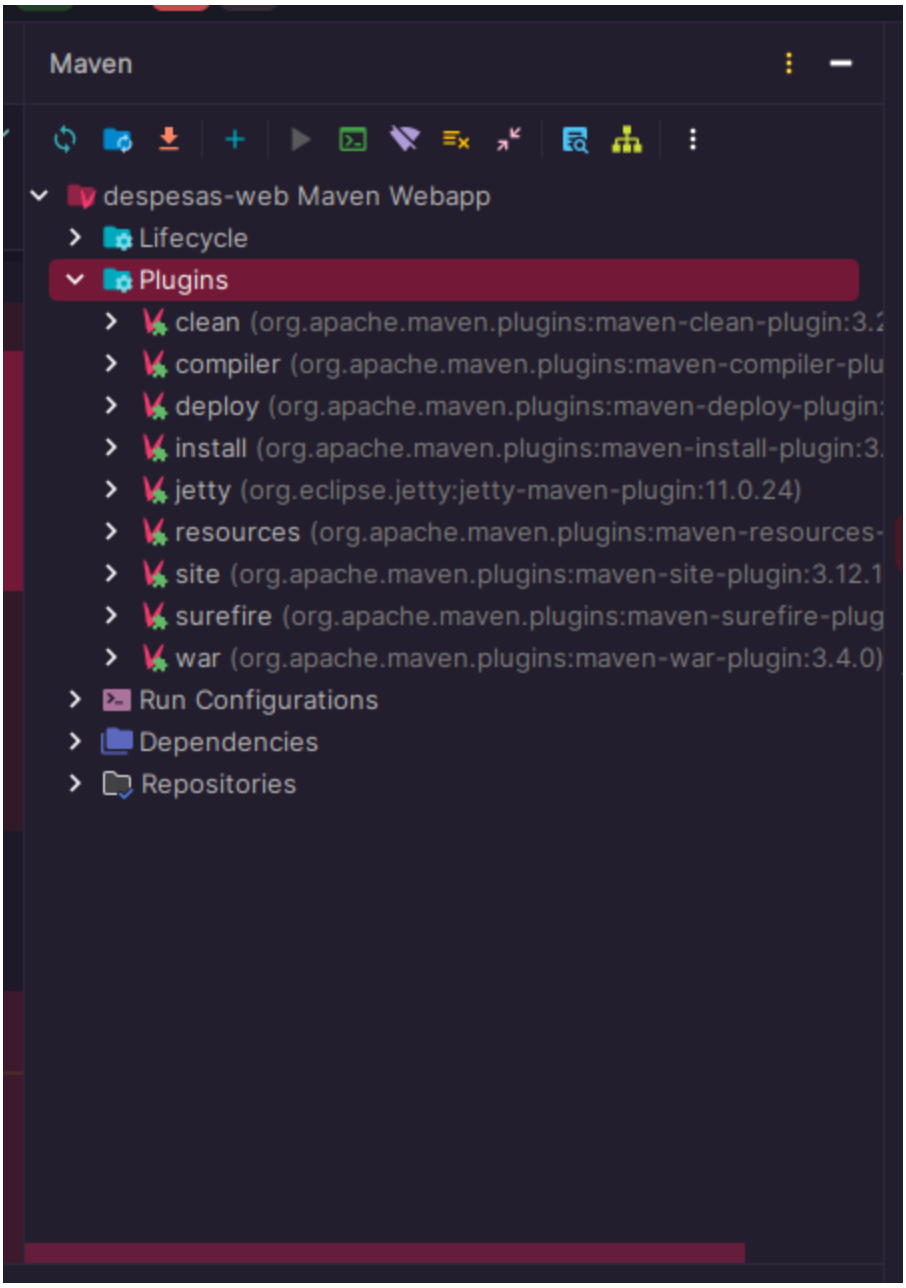
Instalando o plugin do Jet no Intellij

Adicione estas linhas dentro do `pom.xml` para fazermos o uso do plugin do Jet:

```
<build>
    <finalName>despesas-web</finalName>

    <!--
https://mvnrepository.com/artifact/org.eclipse.jetty/jetty-maven-
plugin -->
    <plugins>
        <plugin>
            <groupId>org.eclipse.jetty</groupId>
            <artifactId>jetty-maven-plugin</artifactId>
            <version>11.0.24</version>
        </plugin>
    </plugins>
</build>
```

Ao atualizar as dependencias do Maven, com `CTRL` + `SHIFT` + `O`



1741812889774.png

Ao apertar em `Run` vá ao navegador e acesse `http://localhost:8080/`:

Hello World!

1741812998167.png

> ⚠ A porta padrão que o Jetty usa é a `8080`, mas caso estiver em uso é possivel mudar para que ele acesse outra porta

Definindo contextos

Para definirmos os contextos, seria como definir as aplicações, podemos fazer com está configuração no `pom.xml`:

```
<plugins>
  <plugin>
    <groupId>org.eclipse.jetty</groupId>
    <artifactId>jetty-maven-plugin</artifactId>
    <version>11.0.24</version>
    <configuration>
      <webApp>
        <!--    aqui vai o nome do caminho do contexto    -->
        <contextPath>
          /expenses-web
        </contextPath>
      </webApp>
    </configuration>
```

```
    </plugin>
  </plugins>
```