Week 13 – Programming Web – Glauco – 29/10/2024

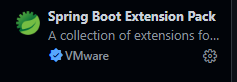
Iniciação com Java com o framework Spring Boot

O Spring Boot vai fazer o backend substituindo o Dbjson

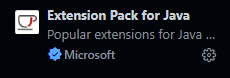
Vamos criar um projeto spring boot usando o spring data e spring framework;

Primeiramente vamos ao Vscode e baixamos as extensões para poder usar sem complicações o Spring Boot

1º

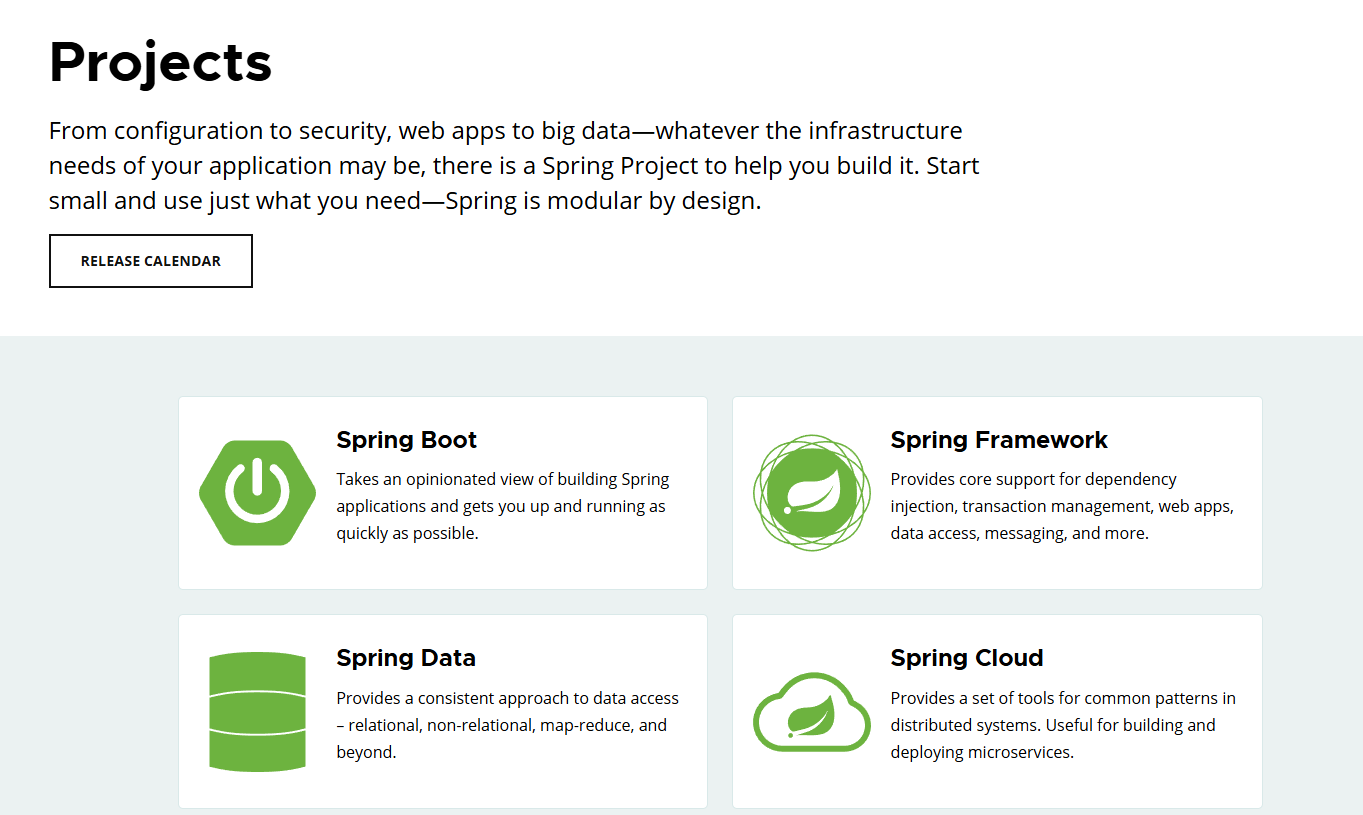


2º



Vamos ao site do SpringBoot, onde encontramos todas as dependências que o SpringBoot têm:

<https://spring.io/projects>

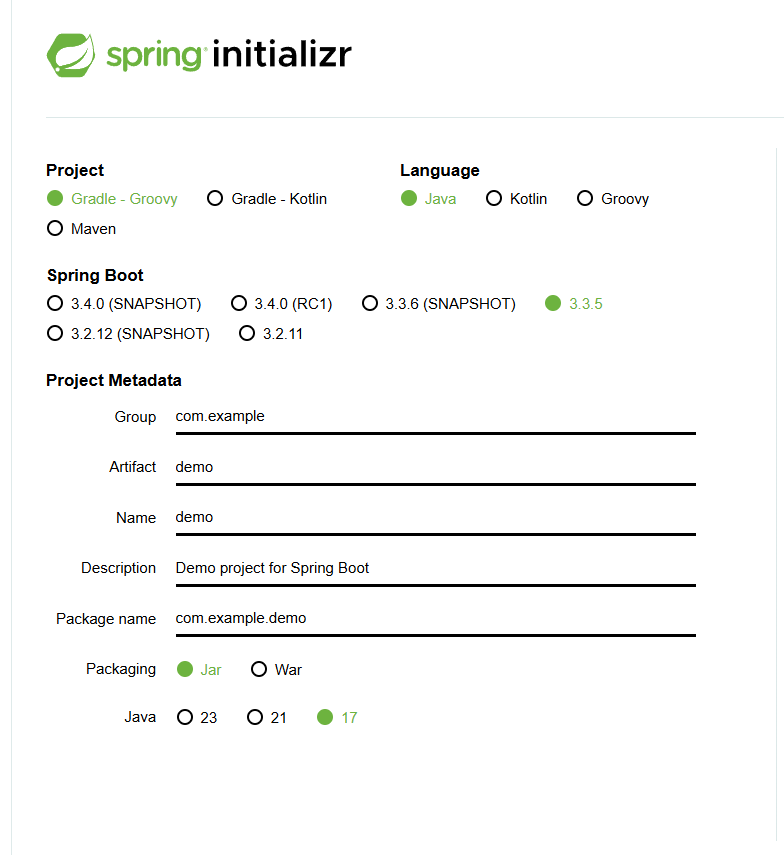


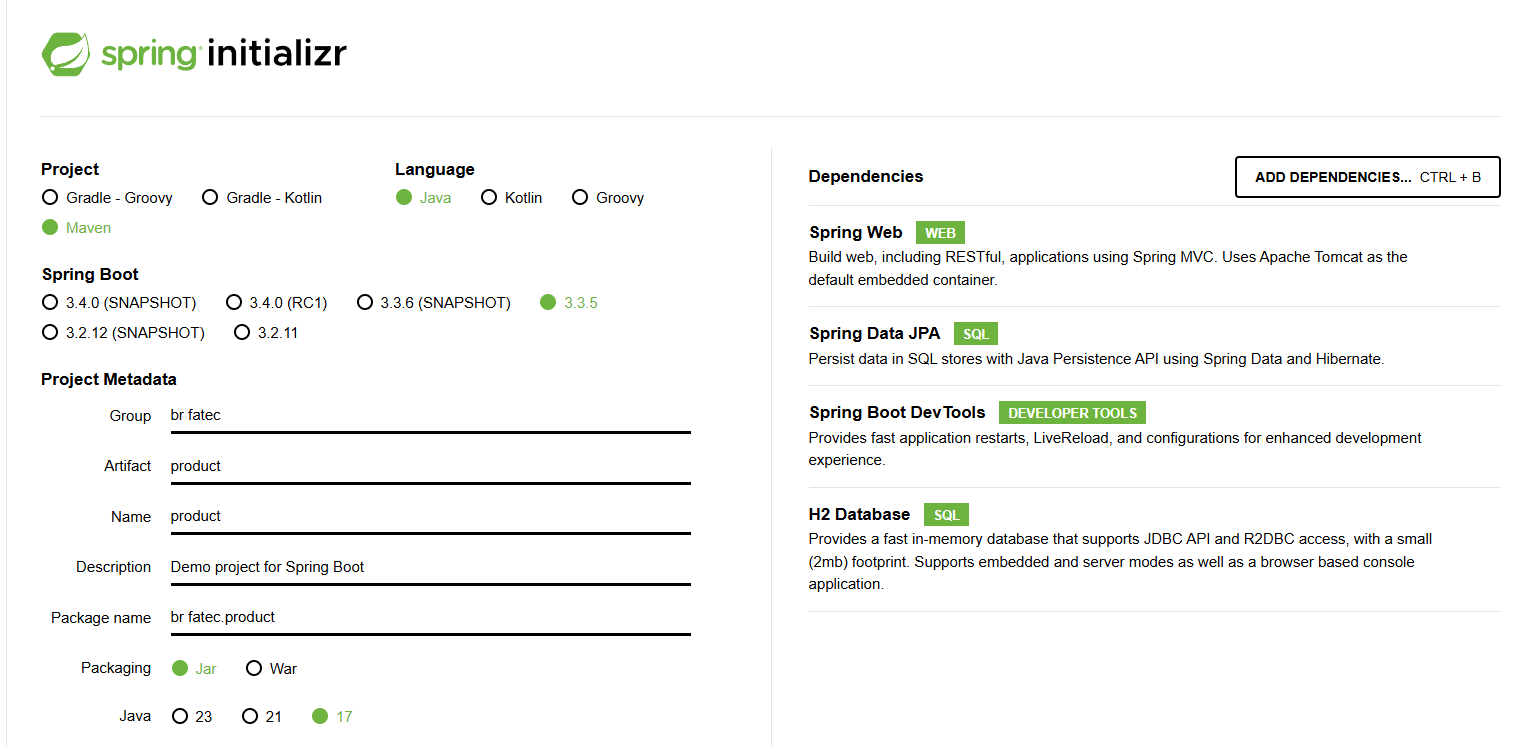


No caso neste momento usaremos mais o Sping Boot, Framework e Data;

Agora Vamos ao site Spring Initalzr 🡪 <https://start.spring.io/>

Onde eu vou dar o formato do meu projeto, as dependências que ele tem a linguagem;



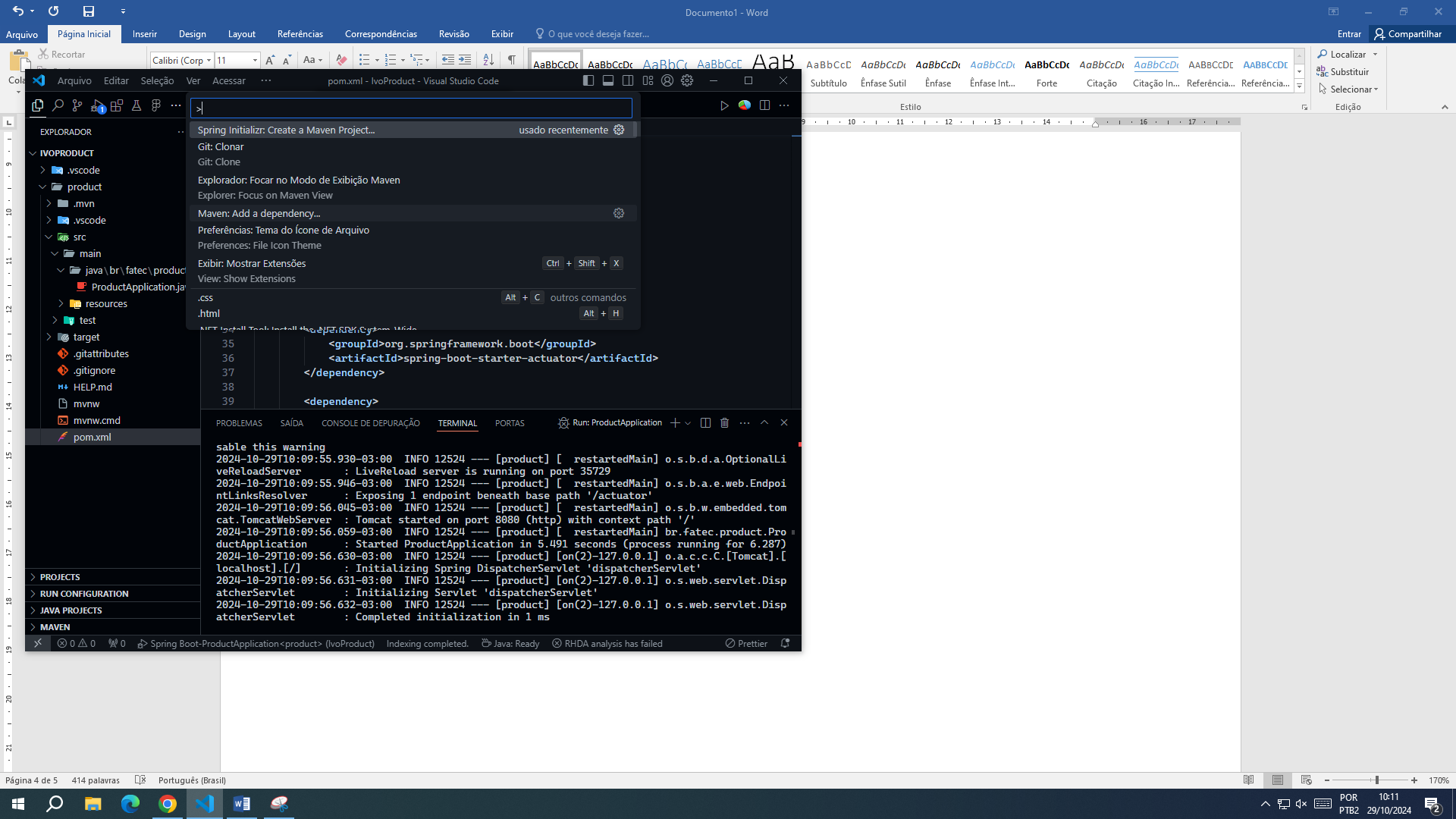


Colocado as dependências e as informações de Metadata

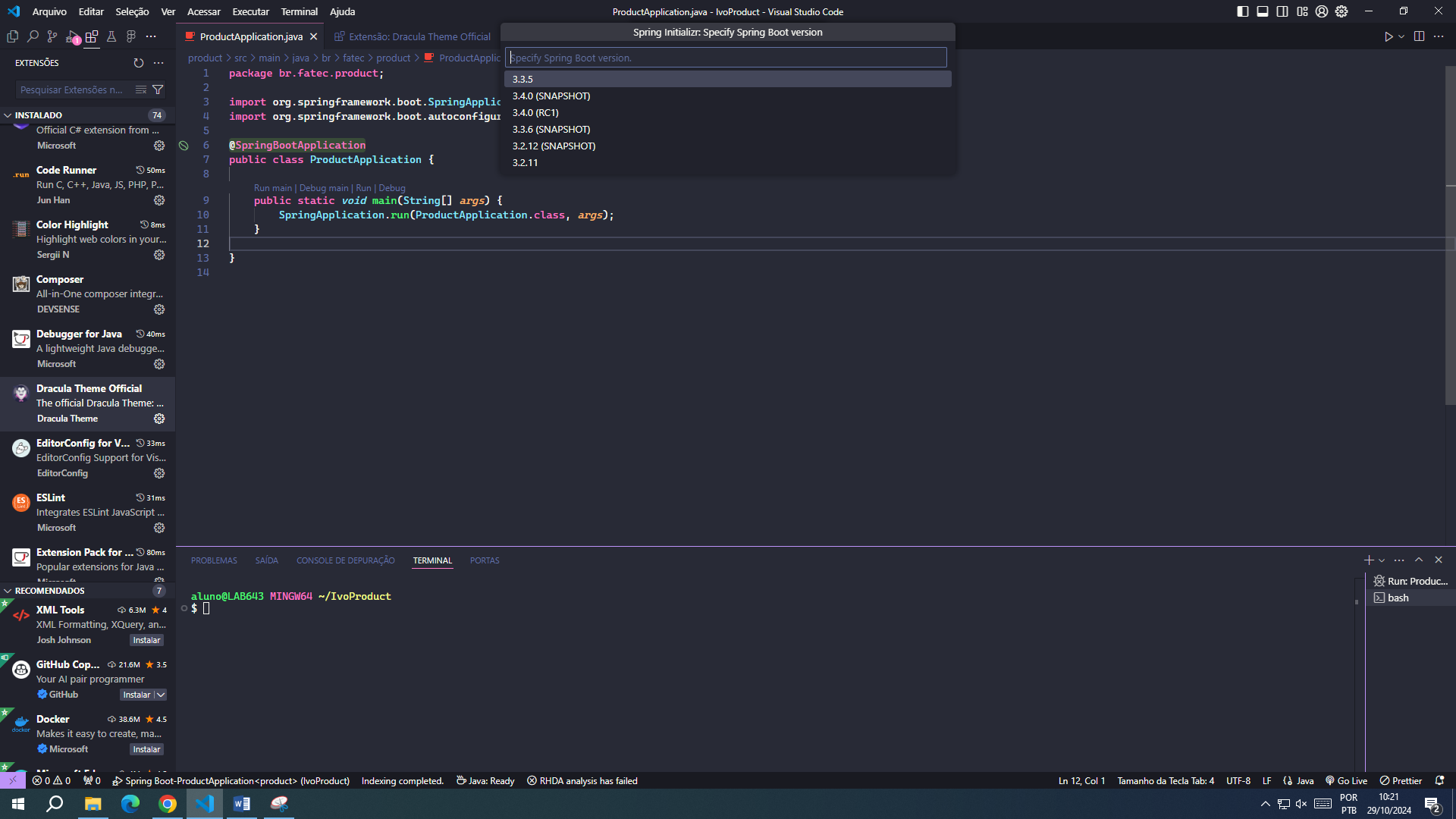
Primeiro o Group e depois o nome do Projeto;

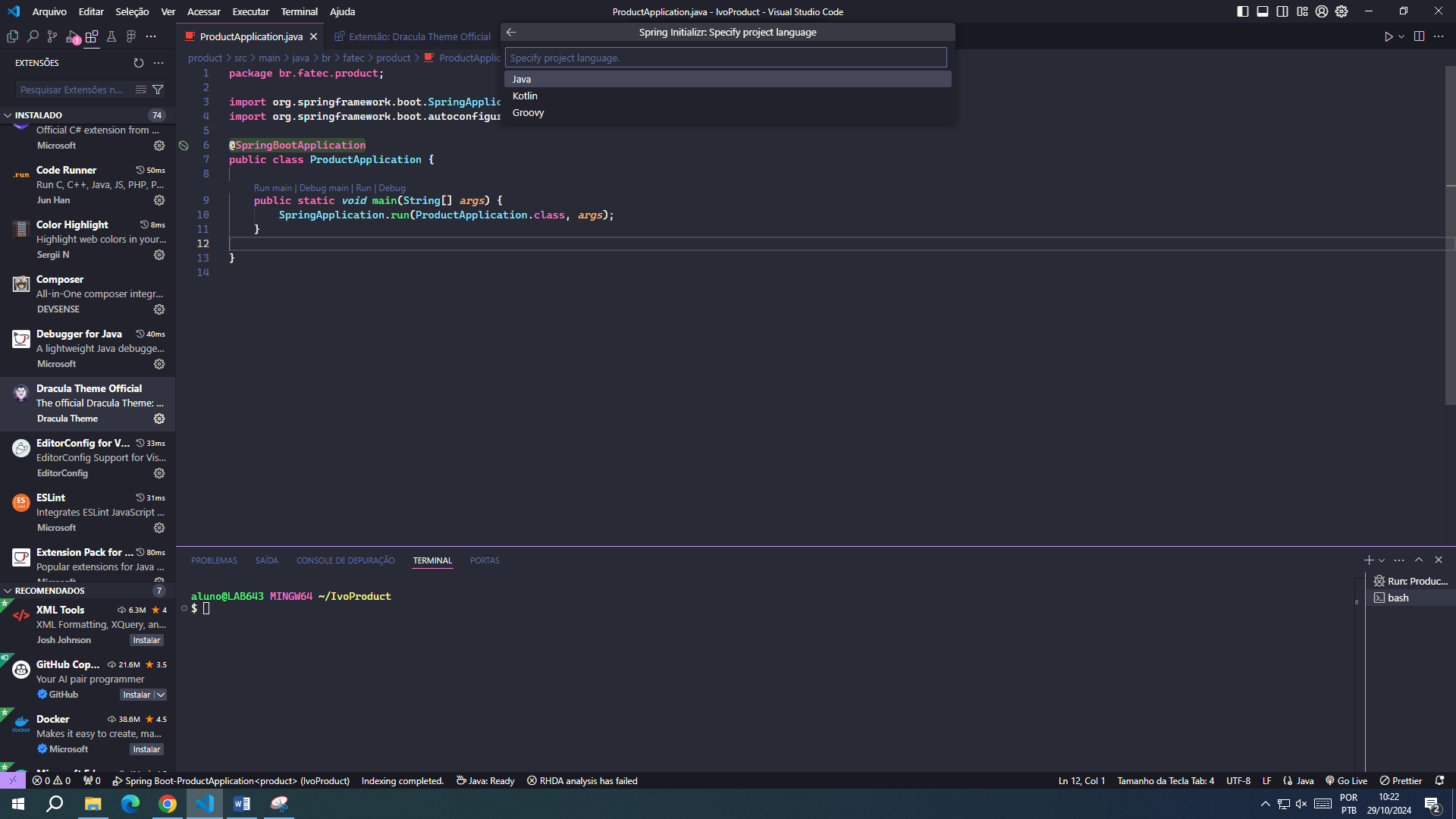
E dá para fazer via Vs code

Onde eu clico em Ctrl + Shift + P

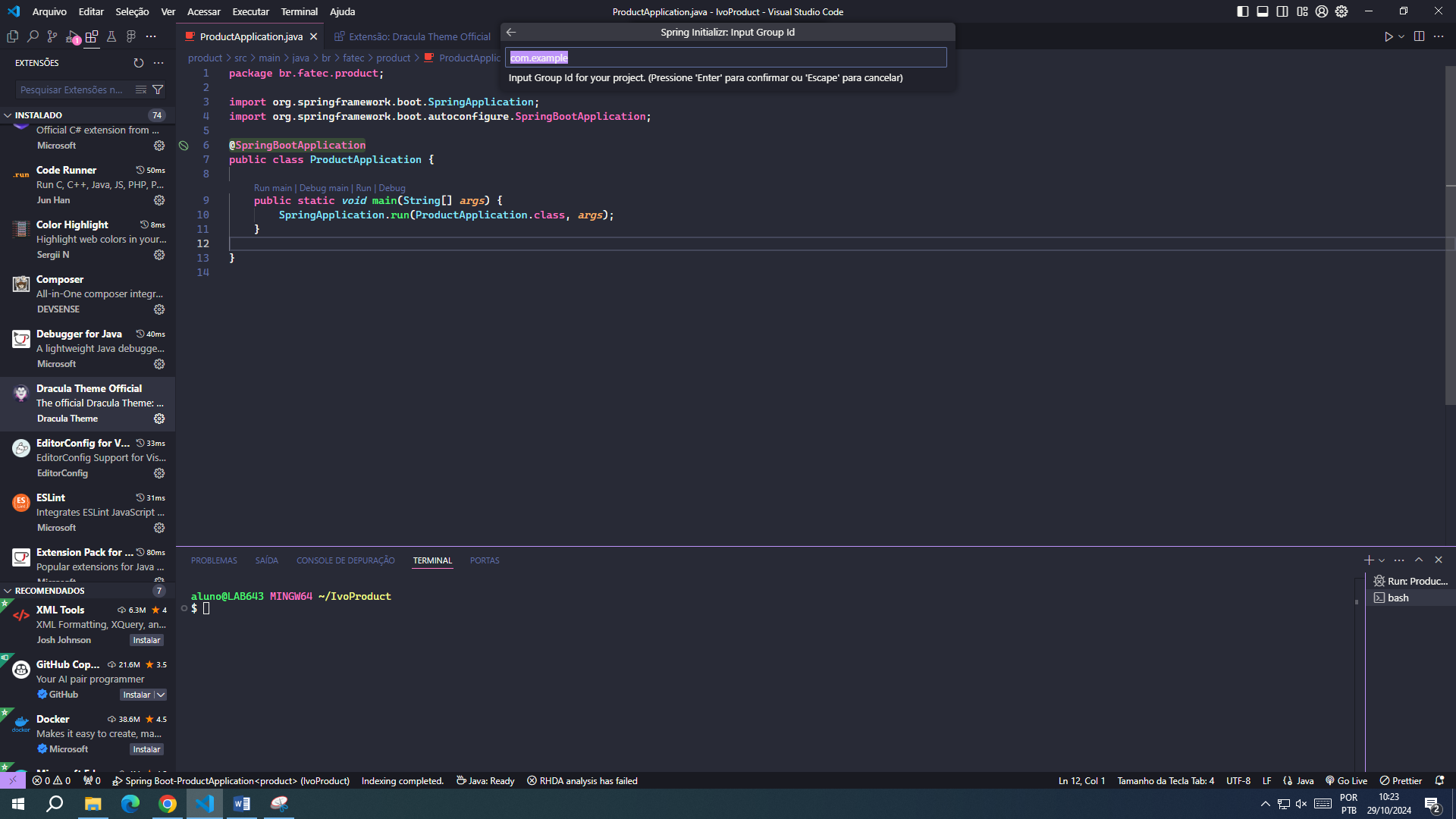


Abrindo está tela clico digito Spring Initialzr: Create a Meaven Project;

 A versão é 🡪 3.3.5

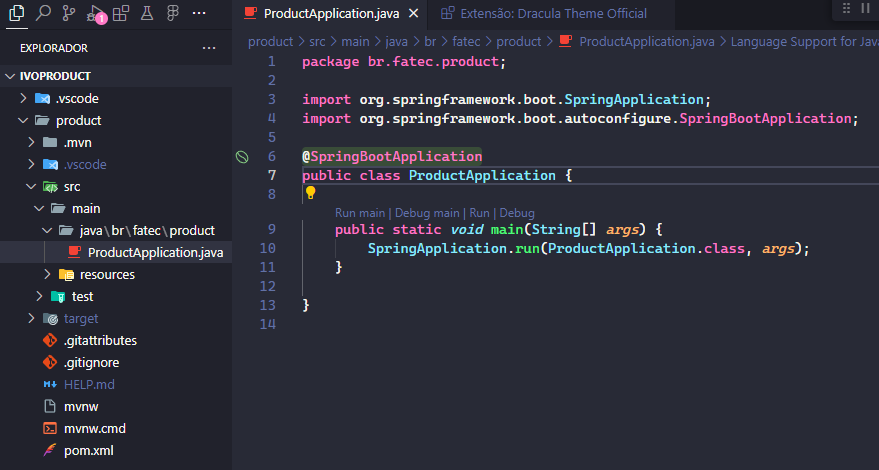


A linguagem é Java;



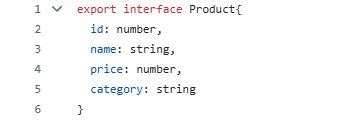
Colocar o nome e seguir o passo a passo seguindo o que foi feito no site spring.io

Depois de todo o projeto iniciado irá ser criado a pasta do projeto e aberto no vscode



Agora em Java criou uma pasta dentro de product com o nome “entities” e dentro criou uma pasta com a entidade JPA para poder relacionar o banco de dados com uma classe;

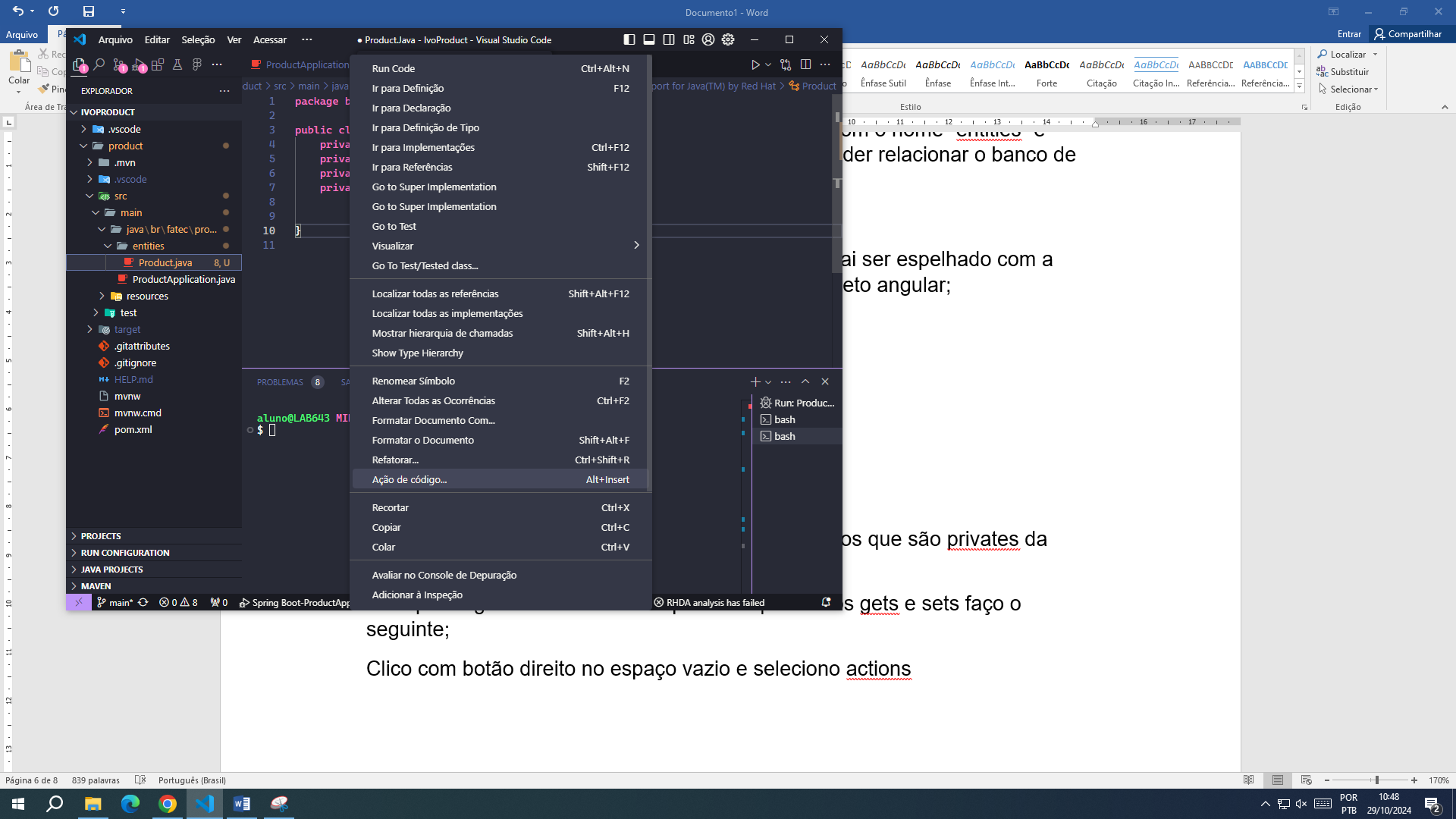
Dentro da pasta crio os atributos desta classe que vai ser espelhado com a interface Product criado em typescript criado no projeto angular;



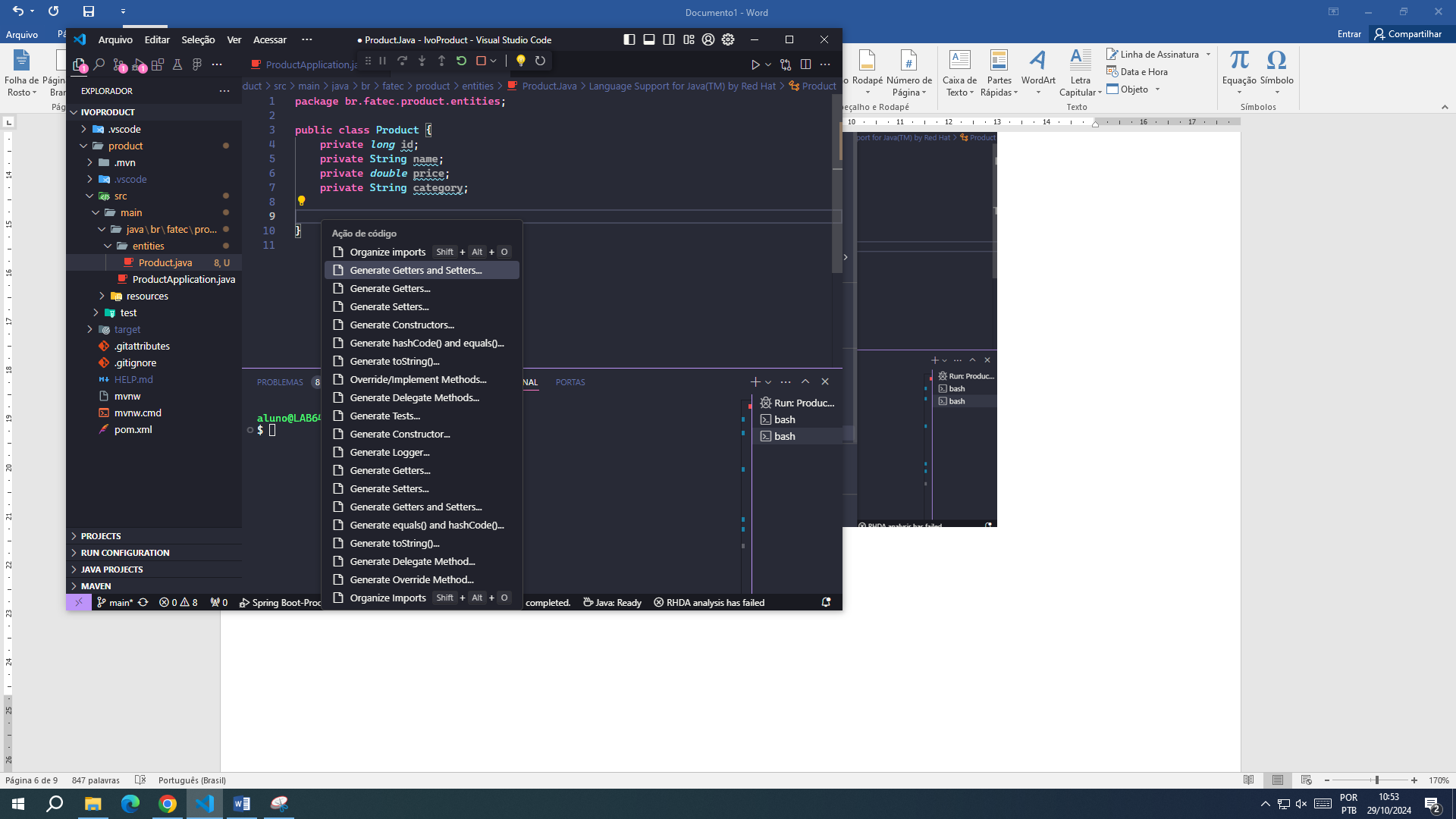
E eu preciso dar métodos públicos aos meus atributos que são privates da classe

Para poder gerar esses métodos públicos que são os gets e sets faço o seguinte;

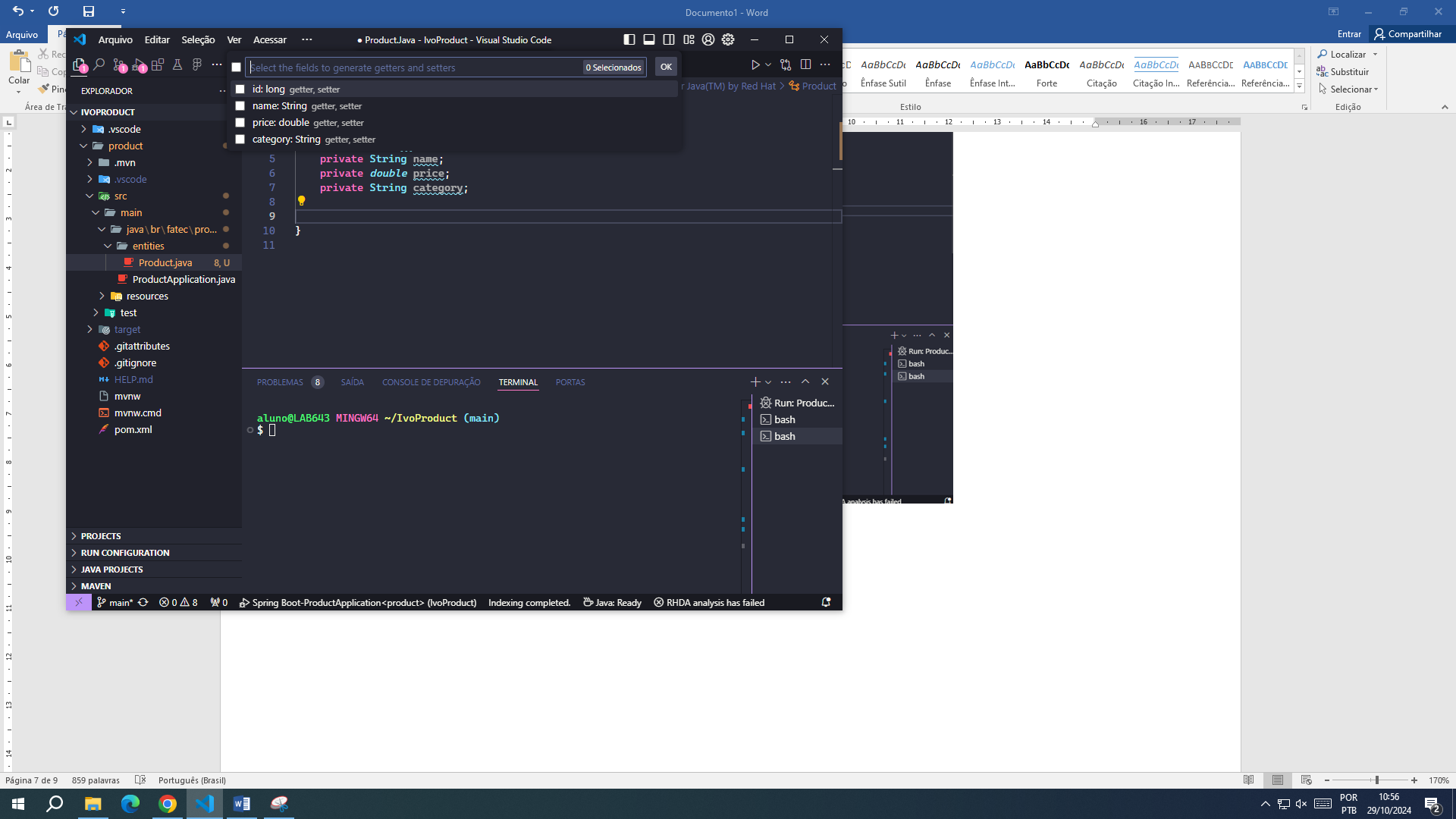
Clico com botão direito no espaço vazio e seleciono actions



Clico neste caso em ação de código e quando aparecer uma lista de opções, clico em Generate Getters and Setters;

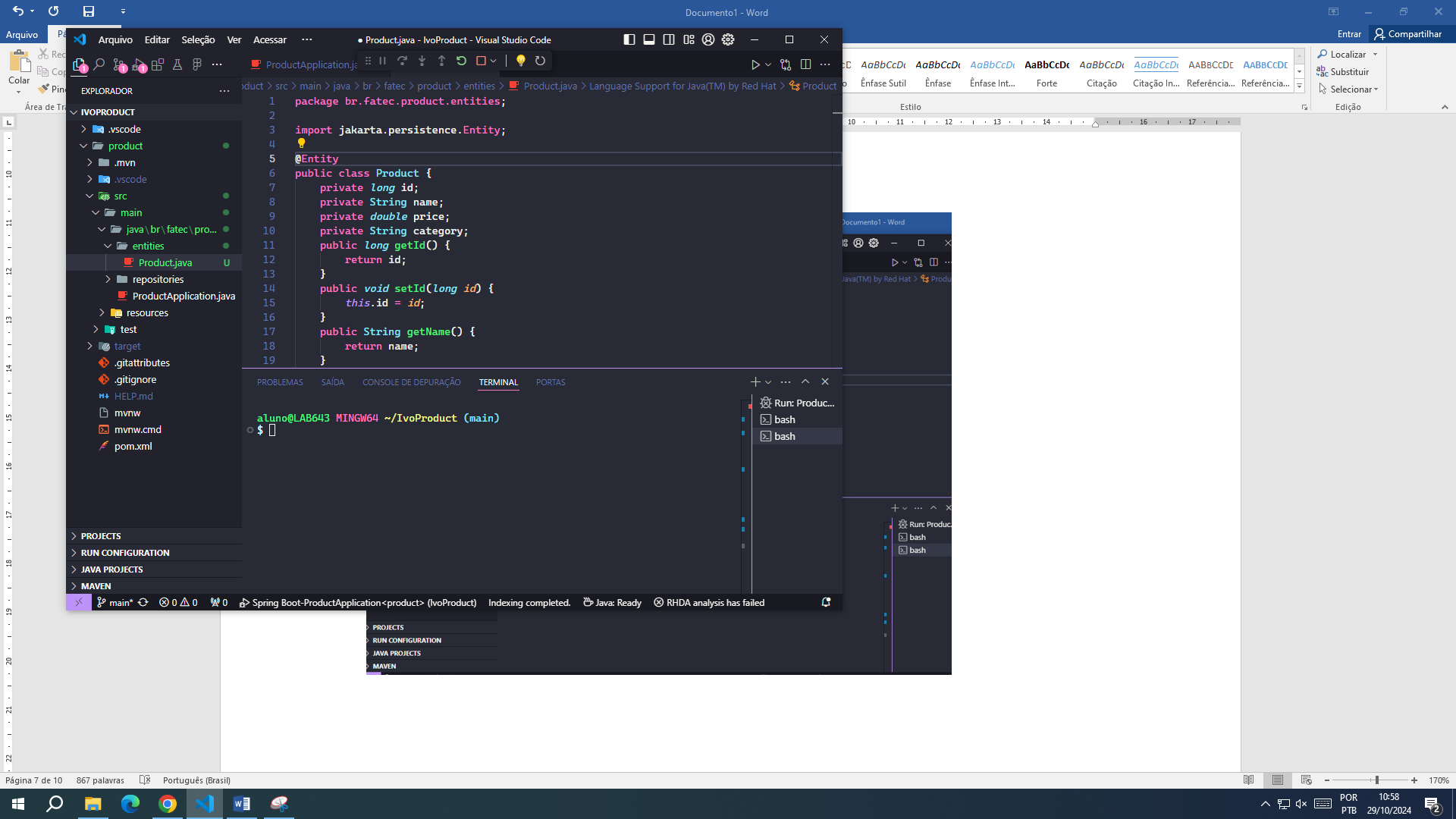


E nesta tela abaixo, seleciono todo os atributos



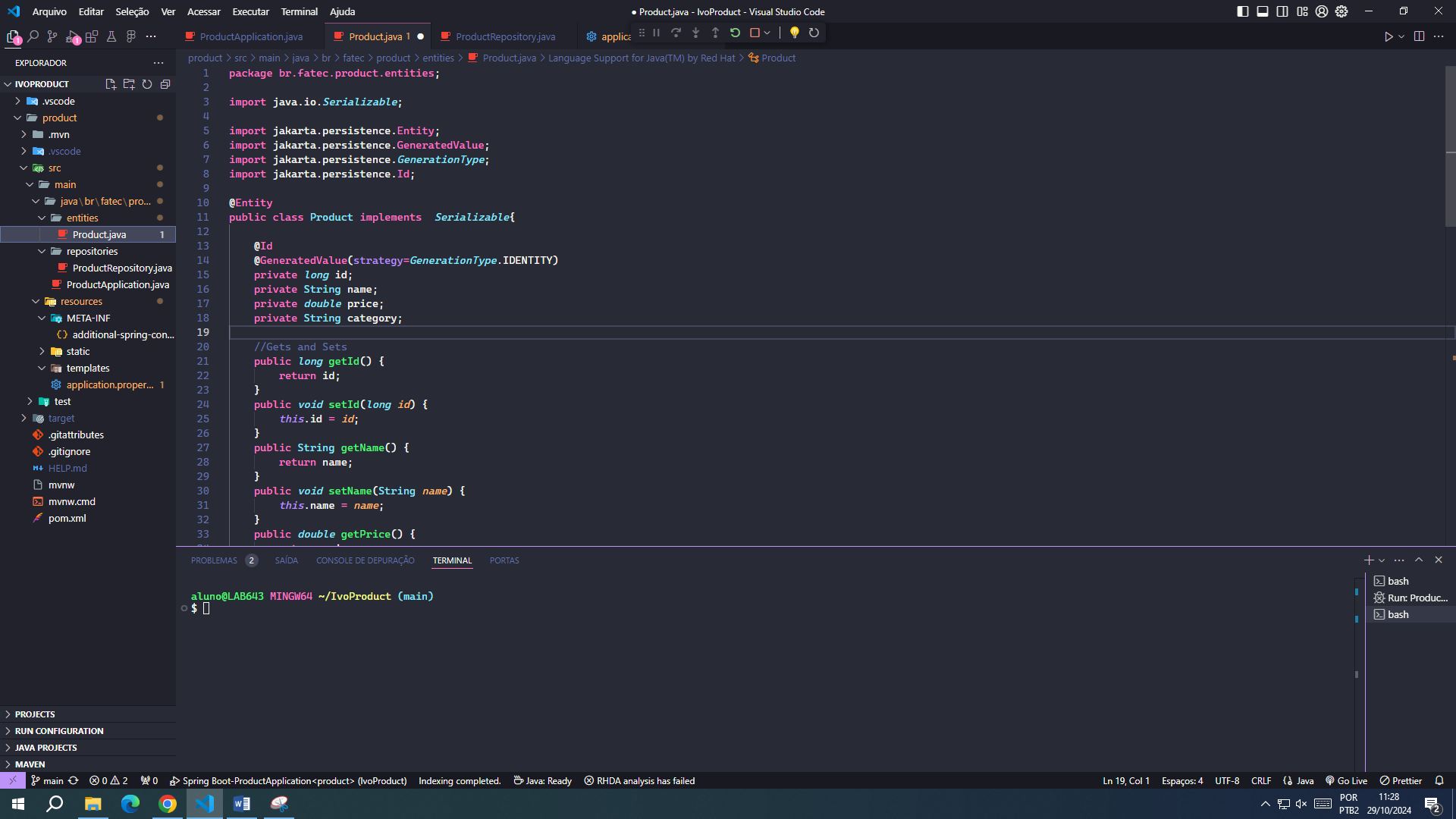
O objetivo é fazer criar uma entidade

Para fazer isso preciso da minha dependência JPA



Colocando @Entity e trazendo o jakarta para dentro do projeto;

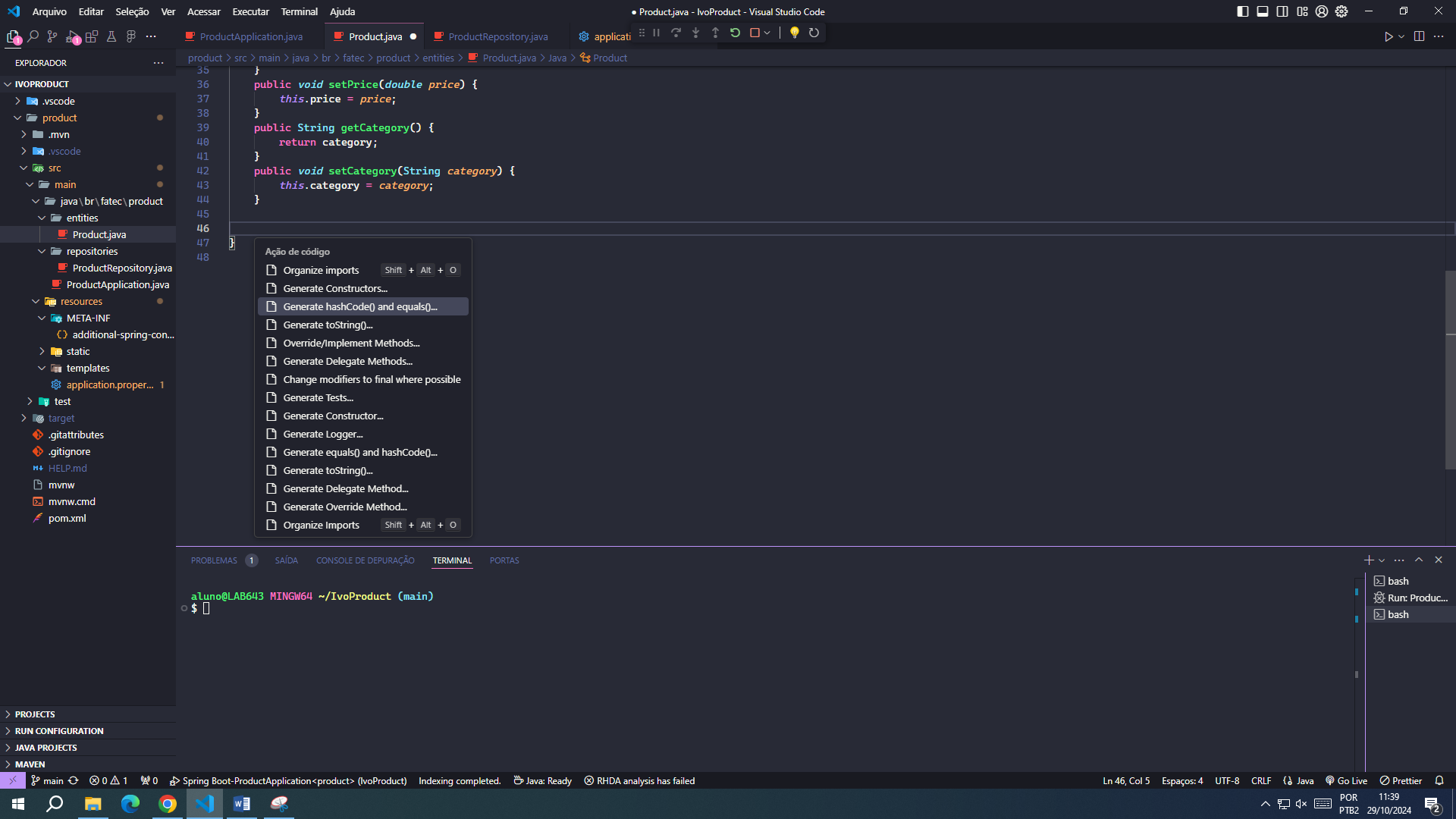
Para poder fazer os ID como chave primária;



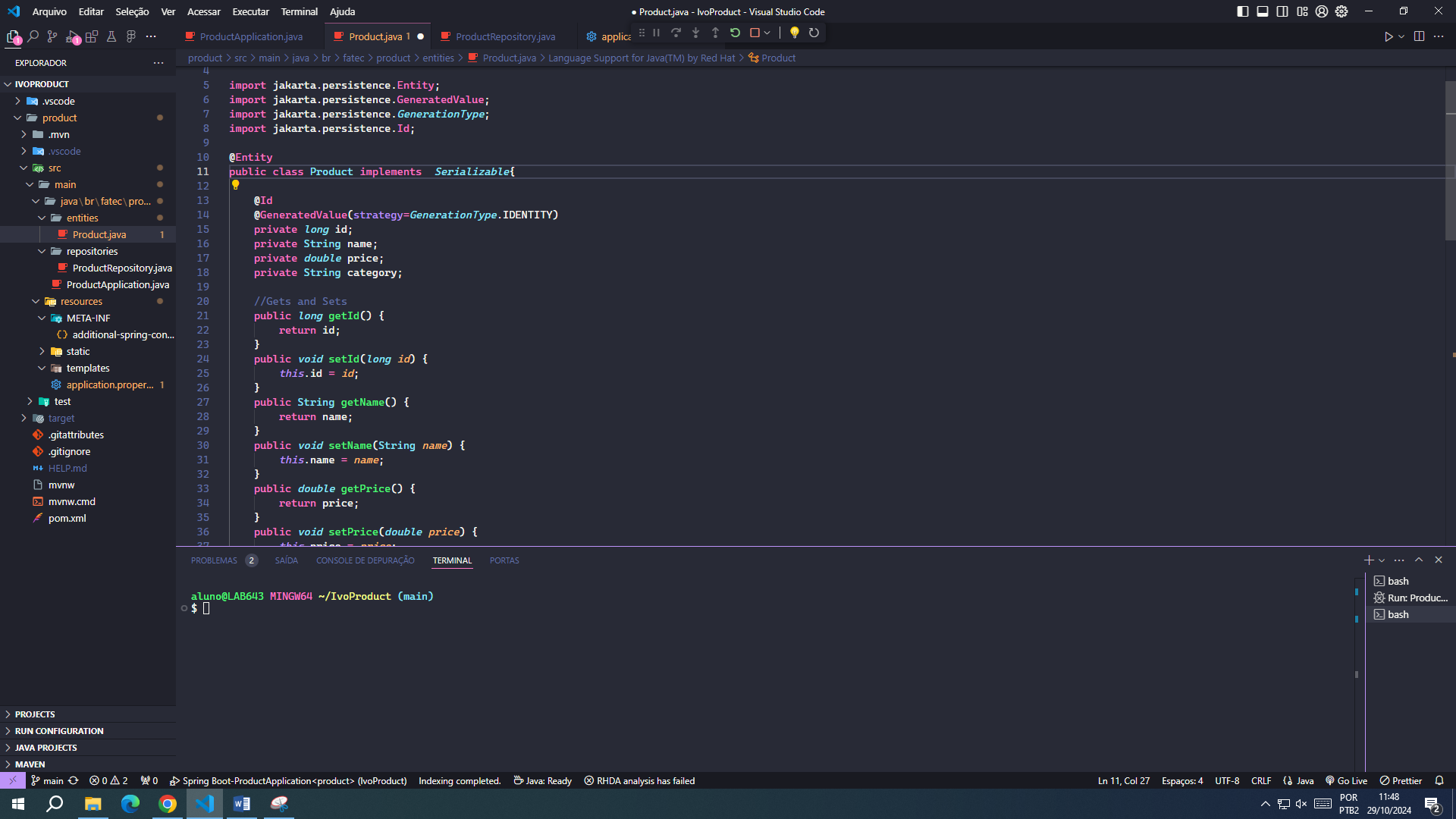
Digito @Id

E @GenerateValue coloco de que forma vai ser o id, será IDENTITY

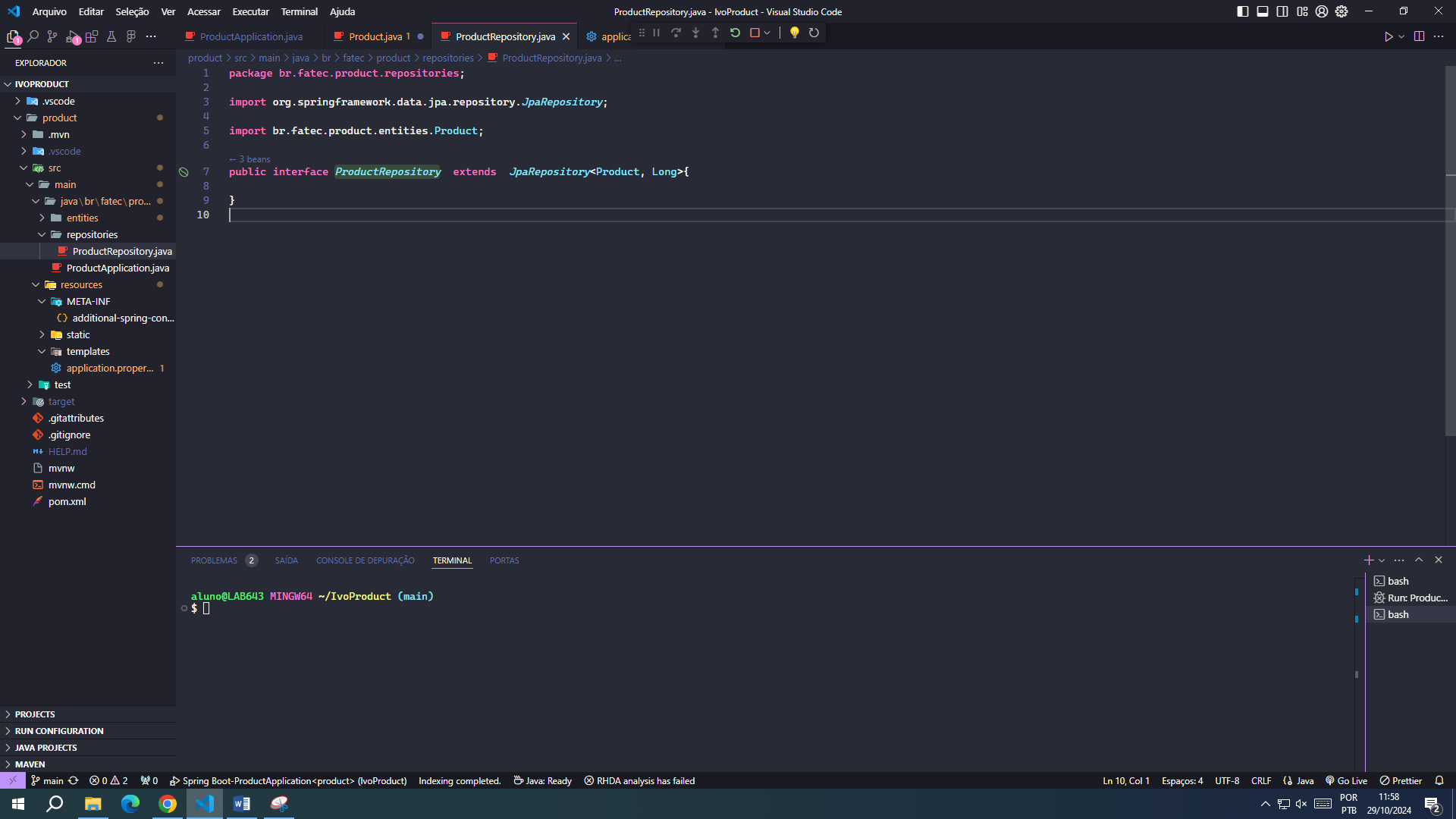
Clico em Generate hashCode() and equals() 🡪 E deixo somento o iD



E implemento o Serializabe

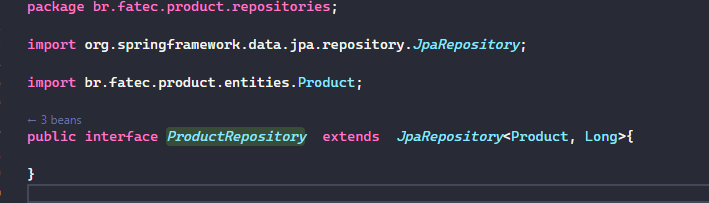


Com isso feito vou e crio uma nova pasta dentro de product com nome de repository



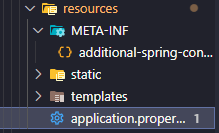
Em repositories

Crio um arquivo e aceso ele

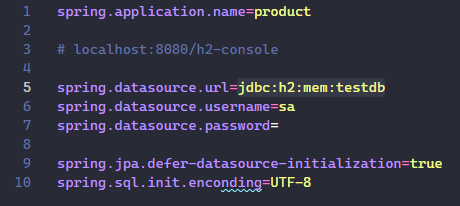


Crio aqui a interface dou um extends usando JbaRepository informando a classe e o Tipo do que quero mapear, no caso o ID está com o tipo long

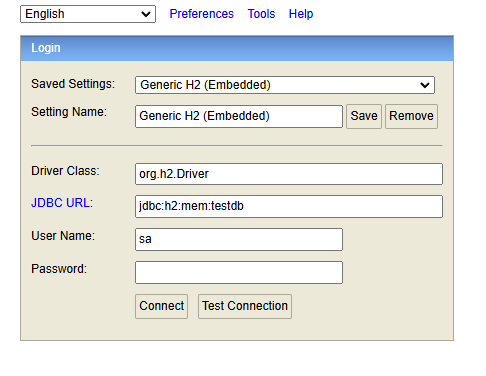
Passando desta parte preciso accear o H2 e faço isso através do aplication property



Abrindo ele e digitando este código dentro dele



Dai com o Java public main rodando vou no meu browser e digito o que está comentado e vai aparecer a imagem abaixo

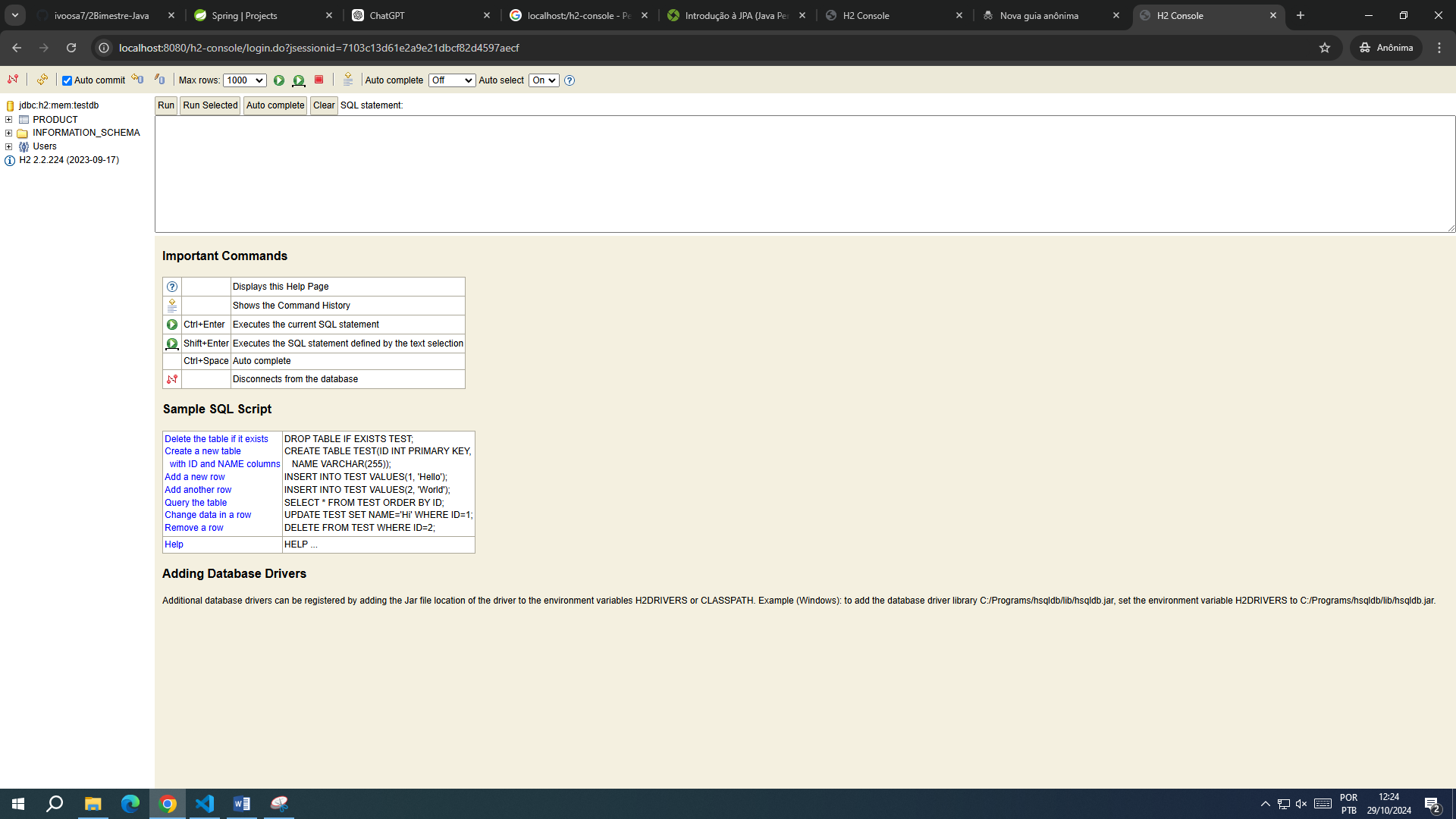


No lugar de JDBC URL copio o que está no meu código



E clico em connect

E então terei acesso ao SpingData para poder fazer meu banco de dados



***Extras da Aula***

O que é JPA

JPA é um framework leve, baseado em [**POJOS (Plain Old Java Objects)**](https://www.devmedia.com.br/diferenca-entre-os-patterns-po-pojo-bo-dto-e-vo/28162) para persistir objetos Java. A **Java Persistence API**, diferente do que muitos imaginam, não é apenas um framework para [**Mapeamento Objeto-Relacional (ORM - Object-Relational Mapping)**](https://www.devmedia.com.br/tecnicas-de-mapeamento-objeto-relacional-revista-sql-magazine-40/6980), ela também oferece diversas funcionalidades essenciais em qualquer aplicação corporativa.

Atualmente temos que praticamente todas as aplicações de grande porte **utilizam JPA para persistir objetos Java**. JPA provê diversas funcionalidades para os programadores

JPA, ou **Java Persistence API**, é uma especificação da plataforma Java que fornece uma maneira padronizada de gerenciar dados em bancos de dados relacionais utilizando objetos Java. Aqui estão os principais pontos sobre JPA:

1. **Mapeamento Objeto-Relacional**: JPA permite mapear classes Java para tabelas de um banco de dados, facilitando a persistência de dados sem a necessidade de escrever consultas SQL complexas.
2. **Entidades**: As classes que representam os dados são chamadas de entidades. Cada instância de uma entidade corresponde a uma linha em uma tabela do banco de dados.
3. **Entity Manager**: O componente central da JPA é o EntityManager, que gerencia o ciclo de vida das entidades, incluindo operações como persistir, atualizar, remover e buscar entidades.
4. **Consultas**: JPA suporta uma linguagem de consulta chamada **JPQL** (Java Persistence Query Language), que permite realizar operações de consulta em entidades em vez de tabelas.
5. **Transações**: JPA integra-se com a gestão de transações, garantindo a consistência dos dados durante operações complexas.
6. **Implementações**: JPA é uma especificação, e várias implementações estão disponíveis, como Hibernate, EclipseLink e OpenJPA.

Em resumo, JPA simplifica o acesso e a manipulação de dados em bancos de dados relacionais, tornando o desenvolvimento de aplicações Java mais eficiente e organizado.

Vou em variáveis de ambiente e configuro para JavaHome

Diferenças entre Gradle e Meaven

Gradle e Maven são ferramentas de automação de build para projetos Java, mas têm algumas diferenças fundamentais:

1. **Modelo de Build**:
   * **Maven**: Usa um modelo baseado em XML e segue uma abordagem de configuração declarativa. Os builds são definidos em um arquivo pom.xml, onde você especifica as dependências e plugins.
   * **Gradle**: Utiliza um modelo baseado em Groovy (ou Kotlin) e adota uma abordagem mais flexível e programática. O arquivo de configuração é build.gradle, permitindo scripts mais dinâmicos.
2. **Desempenho**:
   * **Maven**: A execução é sequencial, o que pode resultar em builds mais lentos, especialmente em projetos grandes.
   * **Gradle**: Oferece build incremental e cache de resultados, o que pode acelerar significativamente o processo de build.
3. **Extensibilidade**:
   * **Maven**: Embora tenha um bom conjunto de plugins, a extensão é mais limitada em comparação ao Gradle.
   * **Gradle**: Permite a criação de plugins personalizados com mais facilidade, além de ter uma API rica para desenvolvimento.
4. **Conceito de Projeto**:
   * **Maven**: Baseia-se em um ciclo de vida de build definido, com fases específicas (compilar, testar, empacotar, etc.).
   * **Gradle**: Utiliza tarefas que podem ser agrupadas e configuradas de maneira mais flexível, permitindo customizações mais profundas.
5. **Comunidade e Suporte**:
   * Ambos têm comunidades ativas, mas Maven tem uma base de usuários mais consolidada, especialmente em projetos legados.

Em resumo, Maven é mais estruturado e fácil de entender para iniciantes, enquanto Gradle oferece mais flexibilidade e desempenho para projetos mais complexos. A escolha entre os dois geralmente depende das necessidades específicas do projeto e das preferências da equipe.