Atelier Framework Spring-03
JPA –Partie02

Objectifs

- **Implémenter le modèle**
  - Créer une entité JPA
  - Créer une base de données Mysql
  - Configurer la base de données avec « **Spring** »

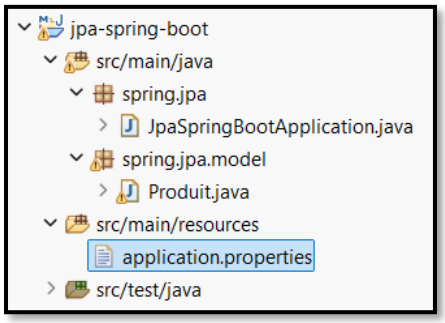
1. Après avoir préparé la structure du projet, passons maintenant à implémenter notre modèle conceptuel :
   - ✓ Créer, dans « **src/main/java** », un package nommé : «**spring.jpa.model**».
   - ✓ Définir la classe «**Produit**» qui présente les caractéristiques suivantes :
     1. Elle implémente l'interface **Serializable** (**pour être utilisée à distance et enoyée entre différentes machines**).
     2. Elle est annotée par « **@Entity** » pour indiquer qu'il s'agit d'une classe persistante (à traduire en une table dans la base de données)
     3. Elle présente pour l'attribut « **id** » (de type Long) deux annotations :
        - **@Id** : pour spécifier une clé primaire
        - **@GeneratedValue** : pour l'auto incrémentation
     4. Voici le code de la classe « **Produit** » :

```
package spring.jpa.model;

import java.io.Serializable;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;

@Entity
public class Produit implements Serializable {
    @Id
    @GeneratedValue
    private Long id;
    private String designation;
    private double prix;
    private int quantite;
    public Long getId() {
```

```
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}
	public String getDesignation() {
		return designation;
	}

	public void setDesignation(String designation) {
		this.designation = designation;
	}

	public double getPrix() {
		return prix;
	}

	public void setPrix(double prix) {
		this.prix = prix;
	}

	public int getQuantite() {
		return quantite;
	}

	public void setQuantite(int quantite) {
		this.quantite = quantite;
	}

	public Produit() {
		super();
	}

	public Produit(String designation, double prix, int quantite) {
		super();
		this.designation = designation;
		this.prix = prix;
		this.quantite = quantite;
	}
	public Produit(Long id, String designation, double prix, int quantite) {
		super();
		this.id = id;
		this.designation = designation;
		this.prix = prix;
		this.quantite = quantite;
	}
}
```

2. Passons maintenant au mapping (ORM du JPA) avec la base de données :

✓ Editer le fichier « **application.properties** » situé sous « **src/main/resources**» pour spécifier les paramètres de connexion à la base de données, ainsi que des paramètres de configuration de Hibernate (framework d'implémentation de JPA):



✓ Saisir les instructions suivantes :

```
#database Configuration:
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/spring-jpa
spring.datasource.username=root
spring.datasource.password=

#Hibernate Configuration:
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto= update
```

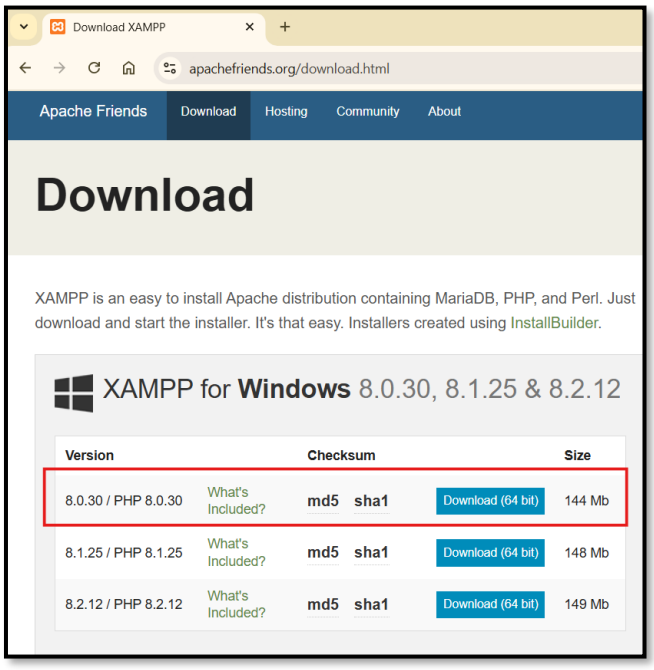
Nom de la base de données

✓ Créer, dans mysql, une base de données nommée : « **spring-jpa** ».

✓ Pour se faire, utiliser l'outil **Xampp comme suit :**

a. Télécharger le logiciel «**xampp**» à partir de l'adresse suivante :

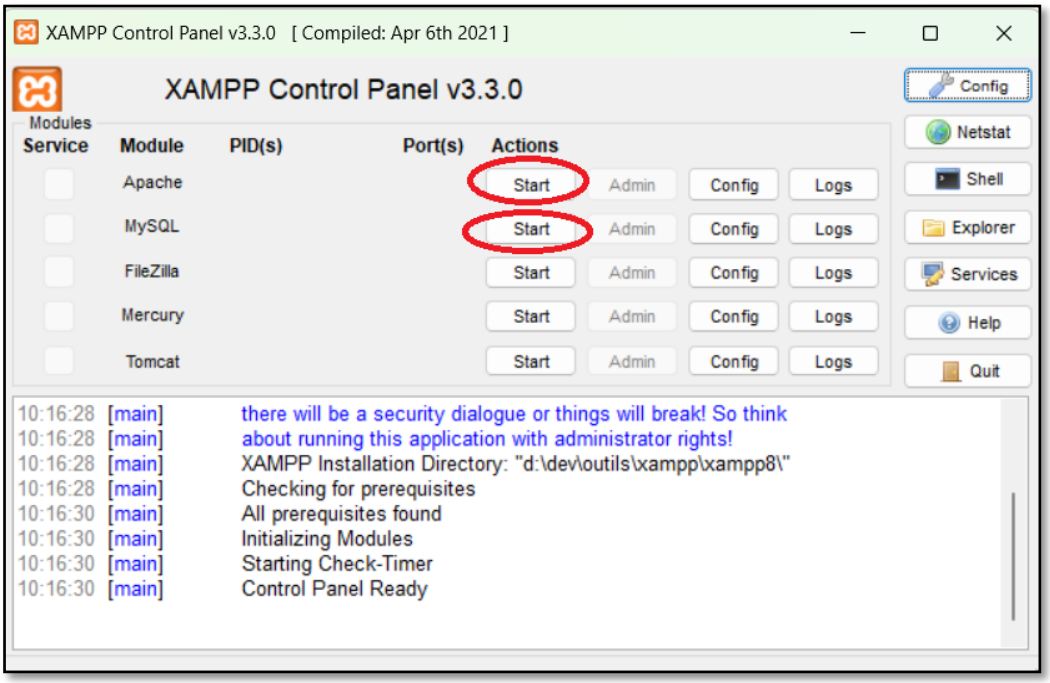
https://www.apachefriends.org/download.html



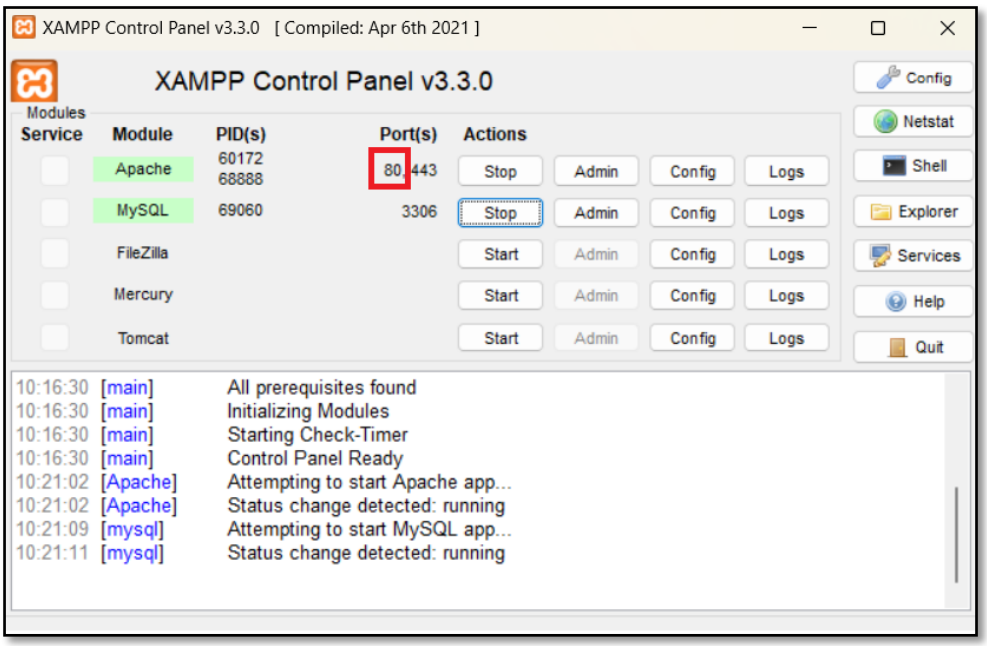
b. Double-cliquer sur le fichier téléchargé pour installer «**xampp**» dans le dossier :

D:\Dev\Outils\Xampp\Xampp8

c. Ouvrir Xampp, un panneau de contrôle de « **Xampp** » s'affiche pour indiquer les états des différents modules. Par défaut, les services des modules sont arrêtés :

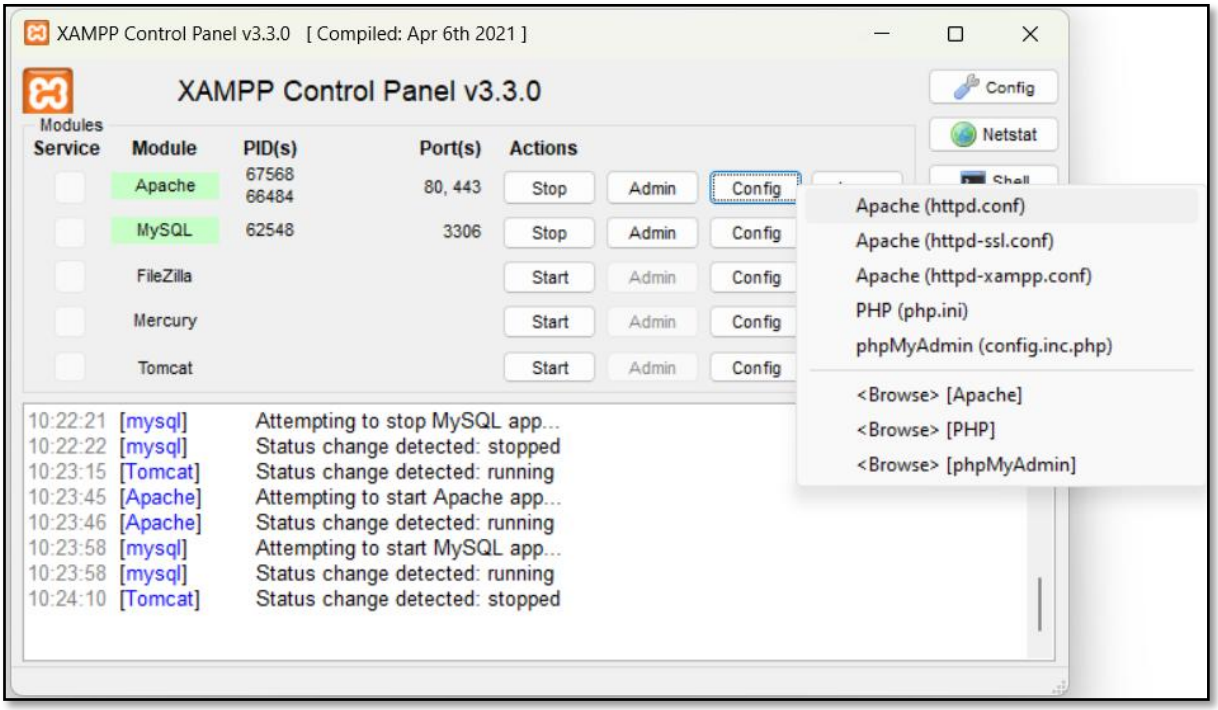


d. Démarrer les deux modules «**Apache** » et « **MySQL** ». Remarquer que le module **Apache** est lancé avec le port « **80** » et que le module MySQL est lancé avec le module « **3306** » :

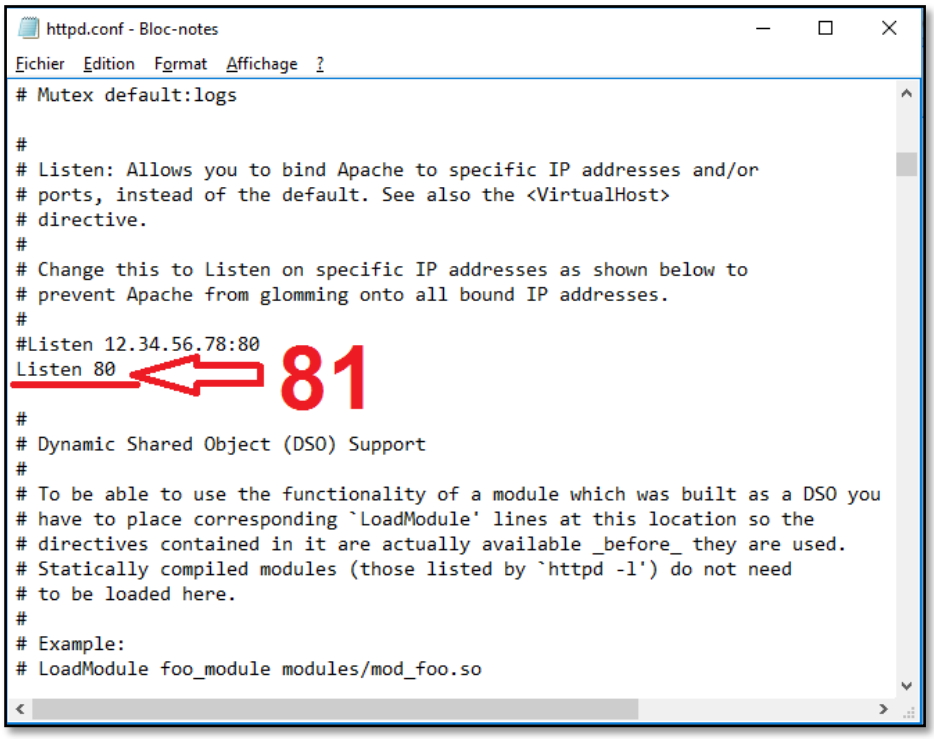


e. Changer le port HTTP à « **81** » (afin de libérer le port « **80** » pour notre serveur web par la suite. Ainsi, suivre les instructions suivantes :

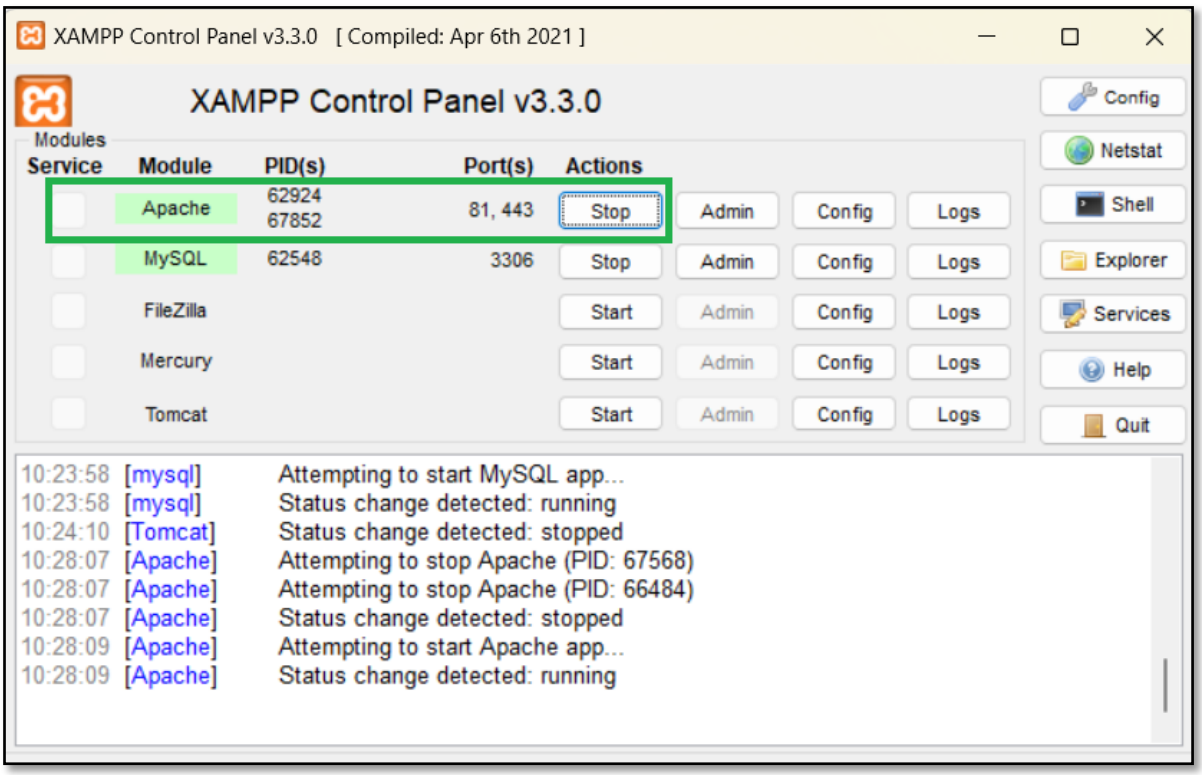
- ❖ Cliquer sur le bouton «**Config**» qui correspond au module «**Apache**»
- ❖ Choisir l'option «**Apache (httpd.conf)**» :



- ❖ Ainsi, ouvrir le fichier « **httpd.conf** » et chercher l'instruction « **Listen 80** » et changer la valeur du port d'écoute du module « **Apache** » à « **81** » et enregistrer.

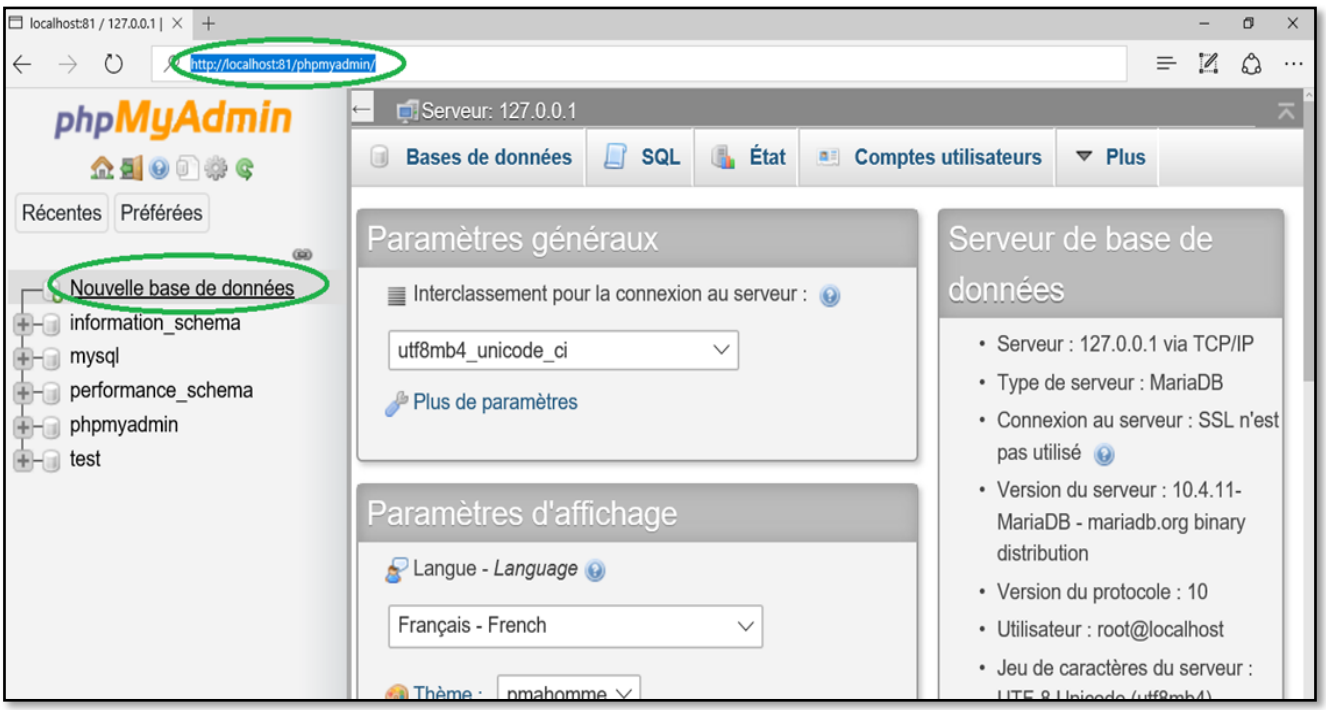


❖ Arrêter le module « **Apache** » puis le démarrer de nouveau. Remarquer le lancement de « **Apache** » avec le port **81** :

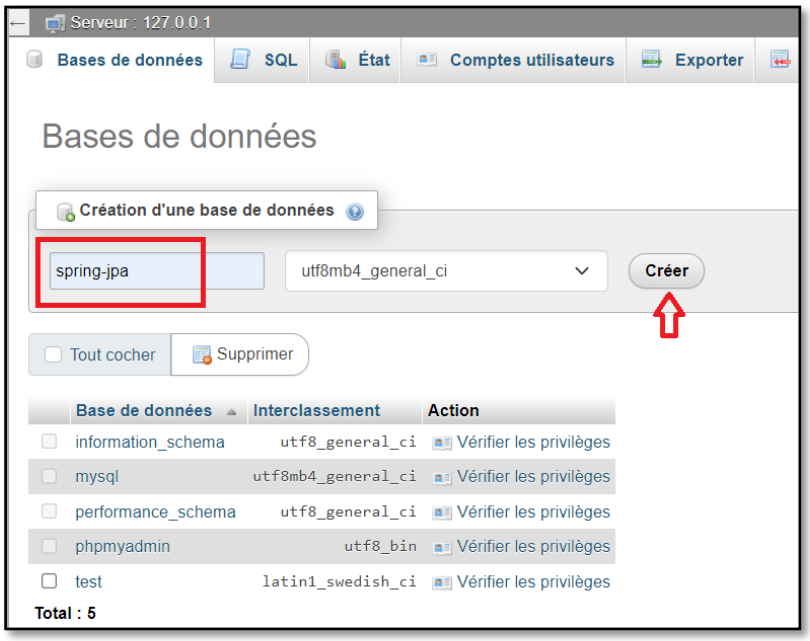


❖ Pour lancer l'application « **phpmyadmin** » ouvrir le navigateur web et spécifier l'url suivante :

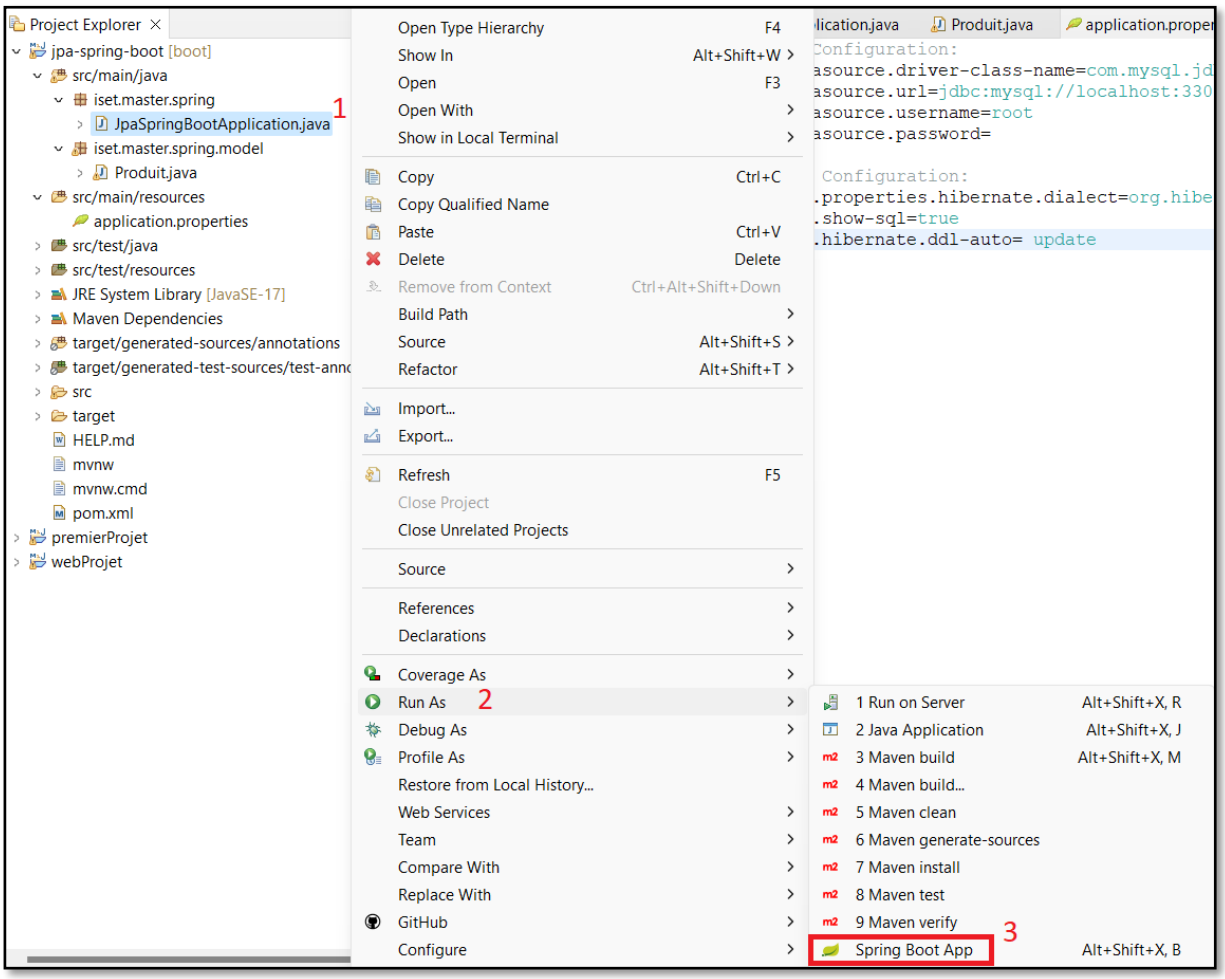
http://localhost:81/phpmyadmin/



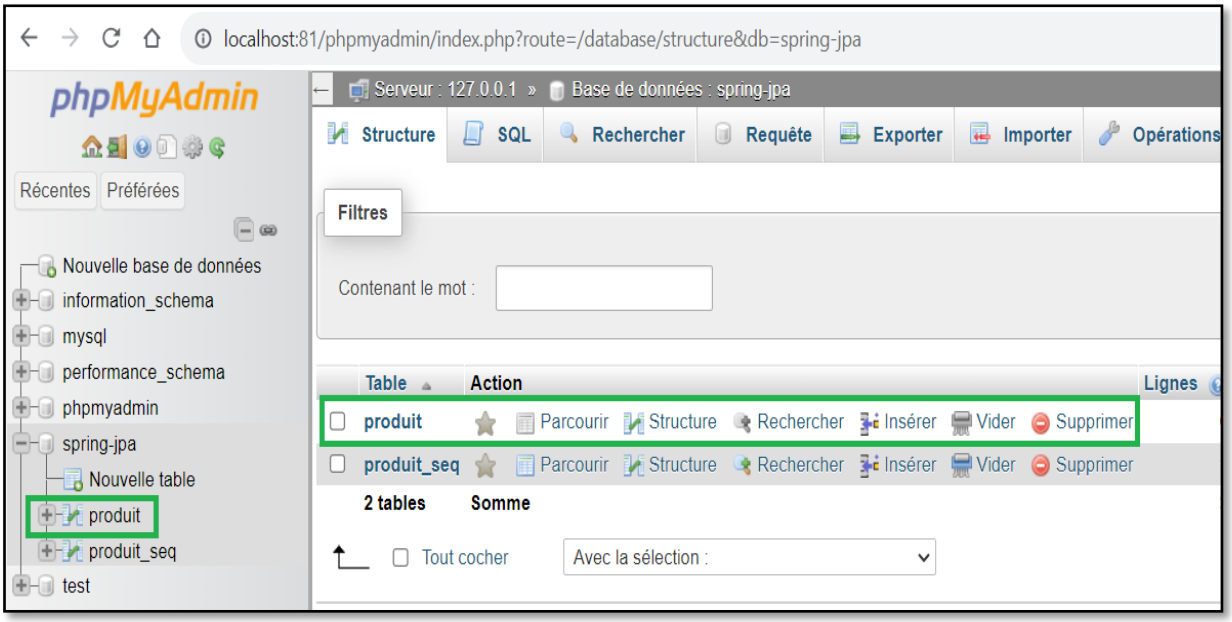
❖ Créer une base de données nommée «**spring-jpa**» :



✓ Maintenant, lancer l'exécution de la classe «**JpaSpringBootApplication**» (classe de démarrage) avec « **Run As/ Spring Boot App** » pour lancer « **Spring Boot**» qui charge le fichier de configuration « **application.properties** » et déclenche le mécanisme ORM :



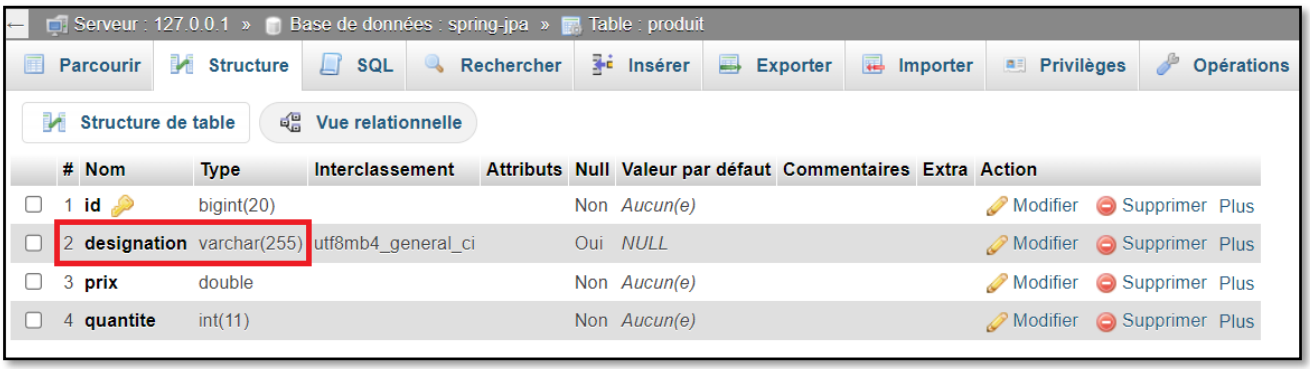
✓ Remarquer la création de la table « **produit** » :



NB :

Il est possible de créer automatiquement la base de données à travers l'application gérée par Spring Boot en ajoutant la configuration suivante, dans le fichier « **application.properties** », à la propriété « **spring.datasource.url** » comme suit :

✓ Le type de la colonne « **designation** » est « **varchar** » (relatif au type « **String** » de l'attribut de la classe) et sa taille est, par défaut, « **255** » :



Pour personnaliser la taille de cette colonne, nous revenons au code de la classe « **Produit** » pour ajouter une annotation « **@Column** » avant la déclaration de l'attribut « **designation** » comme suit (ajouter l'instruction **import** nécessaire) :

```
@Column (length=50)
private String designation;
```

✓ Redémarrer Spring Boot, et remarquer le changement de la structure de la table « **produit** » par la modification de la taille de la colonne « **designation** ».