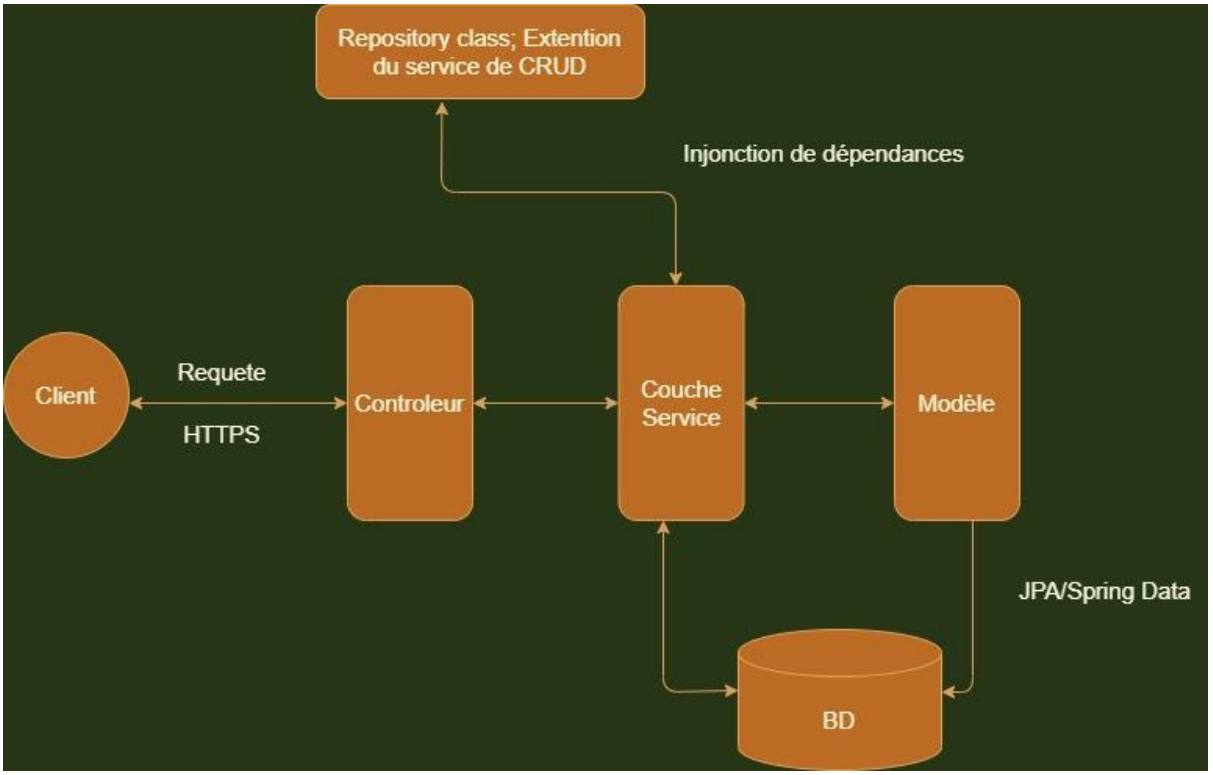
Mise en place d'une API REST

Le (flow) flux d'architecture:



-Nous avons un client REST qui envoie les demandes HTTP (GET, POST, PUT, DELETE)
-La demande HTTP est transmise au contrôleur.
-Le contrôleur

1. mappe la demande
2. la traite
3. appelle la logique du serveur (logique métier ou couche service)

-Au niveau de la couche métier (couche service), nous avons injonction des dépendances de la classe Repository qui étend le service du CRUD à exécuter sur les données de la BD qui sont mappées par les modèles (classes java) via la JPA et Spring Data

-Une fois que la logique métier est exécutée par Spring, les données sont renvoyées au niveau client sous forme JSON.

Mise en place en 5 étapes

1. **Créer un projet spring boot**
2. **Configurer la connexion à la base de données**
3. **Modèles de données**
4. **Couches de persistance et le service / métier**
5. **Contrôleur : qui va exposer notre ressource sur le web.**

spring initializ

Project
Maven Project

language
Java

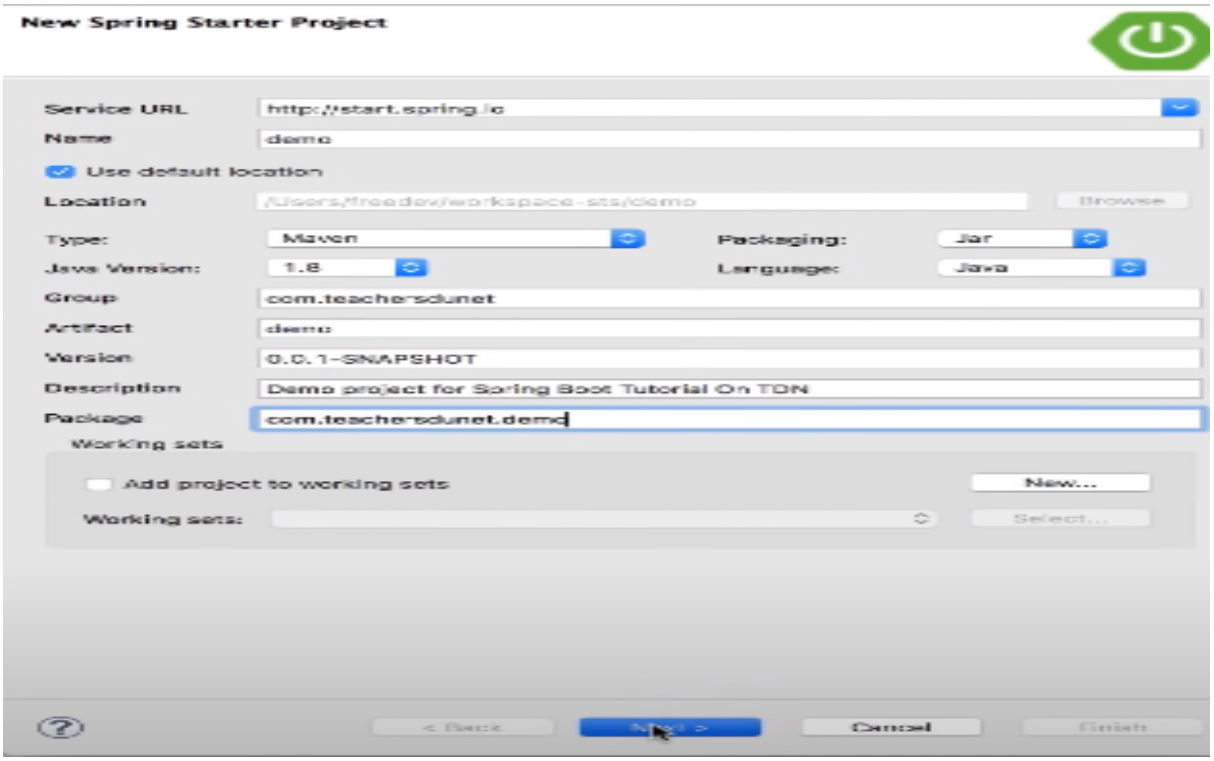
Spring Boot
2.7.0

Project Metadata

Group : com.crud

Artifact : api
Name : crud-app
Description : Crud project for Spring Boot
Package name : com.crud.api
Packaging : Jar
Java : 17

Dependencies :
Spring Web
Spring JPA Data : pour persister les données
MySQL Driver (SQL)
Lombok: (devetools) évite de réécrire les getters, les setters, les constructeurs
Spring Boot DevTools (devetools) : pour beneficier de LiveReload et toutes les configurations pour une expérience assez efficace pour le développement



Dictionnaire de données

Artifact : nom attribué par Maven au niveau du jar
Package : nomGroup.nomArtifact
jar : Caractérise un système entier (application, ordinateur, matériel, réseau), qui n'a pas de lien avec l'extérieur et qui est auto-suffisant - qui n'a pas besoin d'un acte ou d'une information extérieur pour fonctionner.

Pour une application, standalone signifie qu'elle est une application à part entière; ni un plug-in ni un add-on.

Un ordinateur stand-alone signifie qu'il n'est pas relier à un réseau ou à Internet

Un réseau est standalone lorsqu'il n'est pas possible de communiquer vers l'extérieur.

Un fax est standalone, car il permet de tout faire sans autre matériel.

Une imprimante ne l'est pas car elle a besoin d'un dispositif extérieur pour lui fournir l'information à imprimer

@SpringBootApplication: l'annotation qui identifie la classe ApiApplication comme étant une application Spring Boot.
Cette annotation introduit le concept de l'autoconfiguration de notre application

Notre client phpMyAdmin qui nous permet d'interfacer avec notre serveur de base de données MySQL

 # ==============================
 # DB
 # ==============================
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=create-drop

spring.datasource.url=jdbc:mysql://localhost:3306/crud_api?useSSL=false
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.show-sql=true

spring.datasource.url=jdbc:mysql://localhost:3306/crud_api?serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
#spring.jpa.show-sql=true

package repository : continent TelephoneRepository qui représente notre DAO

@Service : considère la classe comme étant une classe qui contient du code métier
@AllArgsConstructor : créer un constructeur avec tous les champs

// @RestController La classe est considérée comme un contrôleur. Elle peut donc exposer une ressource
// control inversion
private final TeleponeService teleponeService;
Car on passe par l'interface pour appeler les méthodes de TelephoneServiceImpl 23

couche persistance : TelephoneRepository
couche métier : TelephoneServiceImpl

revoir 17mn

@RequestBody : Pour que Spring envoie les données au niveau du body de la requête.
L'annotation indiquant un paramètre de méthode doit être liée au corps de la requête Web. Le corps de la requête est transmis via un HttpMessageConverter pour résoudre l'argument de la méthode en fonction du type de contenu de la requête. En option, la validation automatique peut être appliquée en annotant l'argument avec @Valid.

public Telephone create(@RequestBody Telephone telephone)
l'objet json (telephone) sera envoyé au niveau d body de la requête