

Brevet de Technicien Supérieur

Contrôle JAVA :

Gestion des Abonnements



**Réalisé par :**

**ZAKARIA ZIGHIGHI**

**Encadré par :**

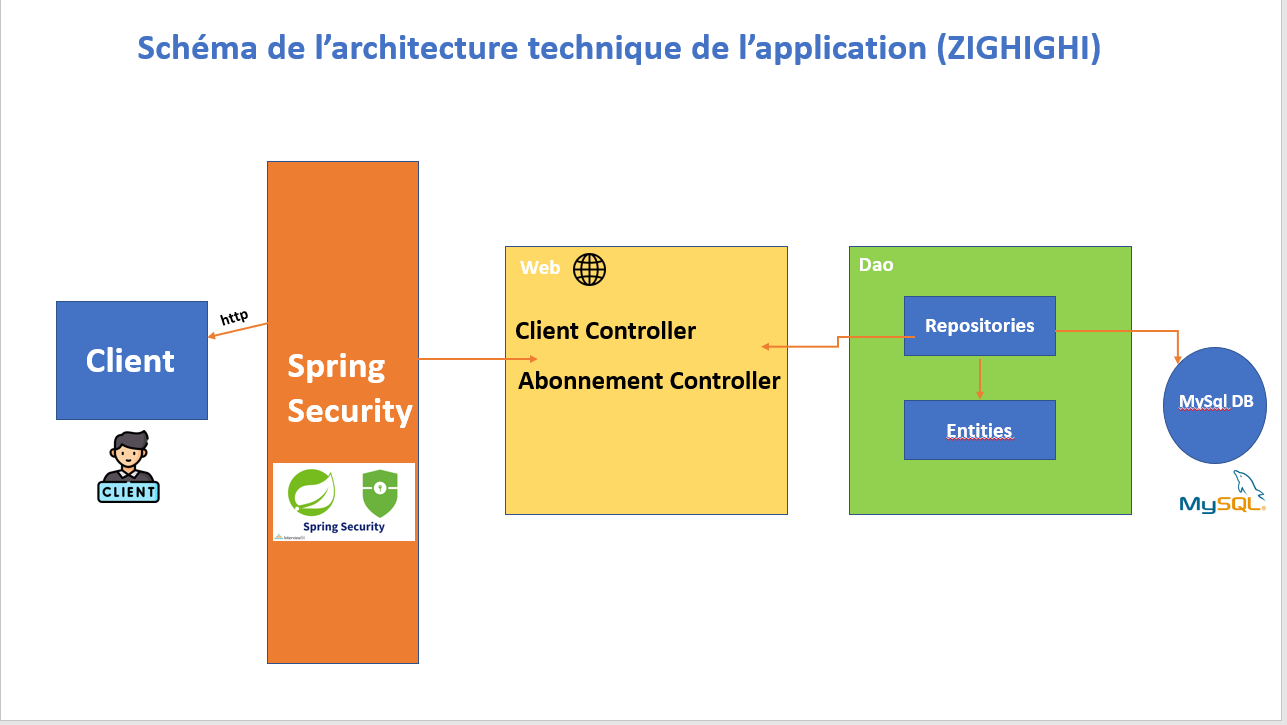
**Pr. Mohammed Youssfi**

GLSID 2

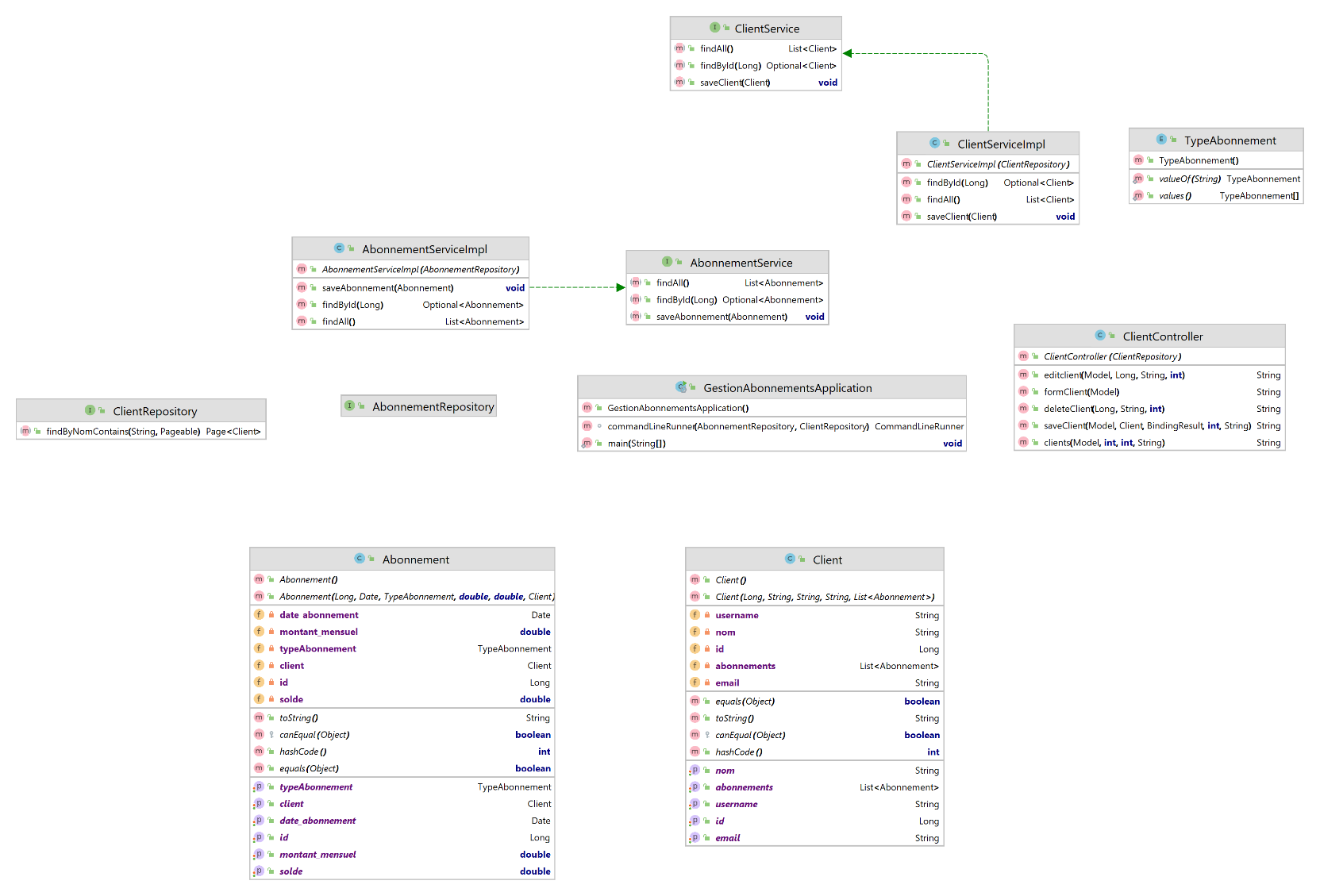
Année universitaire : 2022-23

# **Rapport Examen :**

## **// 1- Schéma de l’architecture technique de l’application :**



## **// 2- Diagramme de classe représentant les données manipulées par l’application**



## **// 3-Couche DAO :**

a. Créer les entités JPA

Abonnement :

**package** ma.enset.gestion\_abonnements.entities;  
  
**import** jakarta.persistence.\*;  
**import** lombok.AllArgsConstructor;  
**import** lombok.Data;  
**import** lombok.NoArgsConstructor;  
**import** ma.enset.gestion\_abonnements.enums.TypeAbonnement;  
  
**import** java.util.Date;  
@Entity  
@Data @NoArgsConstructor @AllArgsConstructor  
**public class** Abonnement {  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 **private** Long **id**;  
 @Temporal(TemporalType.***DATE***)  
 **private** Date **date\_abonnement**;  
  
 @Enumerated(EnumType.***STRING***)  
 **private** TypeAbonnement **typeAbonnement**;  
  
 **private double solde**;  
 **private double montant\_mensuel**;  
  
  
 @ManyToOne  
 **private** Client **client**;  
   
}

Client :

**package** ma.enset.gestion\_abonnements.entities;  
  
**import** jakarta.persistence.\*;  
**import** lombok.AllArgsConstructor;  
**import** lombok.Data;  
**import** lombok.NoArgsConstructor;  
  
**import** java.util.List;  
  
@Entity  
@Data @NoArgsConstructor @AllArgsConstructor  
**public class** Client {  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 **private** Long **id**;  
 **private** String **nom**;  
 **private** String **email**;  
 **private** String **username**;  
  
 @OneToMany(mappedBy = **"client"**,fetch = FetchType.***LAZY***)  
 **private** List<Abonnement> **abonnements**;  
}

b. Créer les interfaces JpaRepository basées sur Spring Data

AbonnementRepository:

**package** ma.enset.gestion\_abonnements.repositories;  
  
**import** ma.enset.gestion\_abonnements.entities.Abonnement;  
**import** org.springframework.data.jpa.repository.JpaRepository;  
  
**public interface** AbonnementRepository **extends** JpaRepository<Abonnement, Long> {  
}

ClientRepository:

**package** ma.enset.gestion\_abonnements.repositories;  
  
**import** ma.enset.gestion\_abonnements.entities.Client;  
**import** org.springframework.data.jpa.repository.JpaRepository;  
  
**public interface** ClientRepository **extends** JpaRepository<Client, Long> {  
}

c. Tester la couche DAO

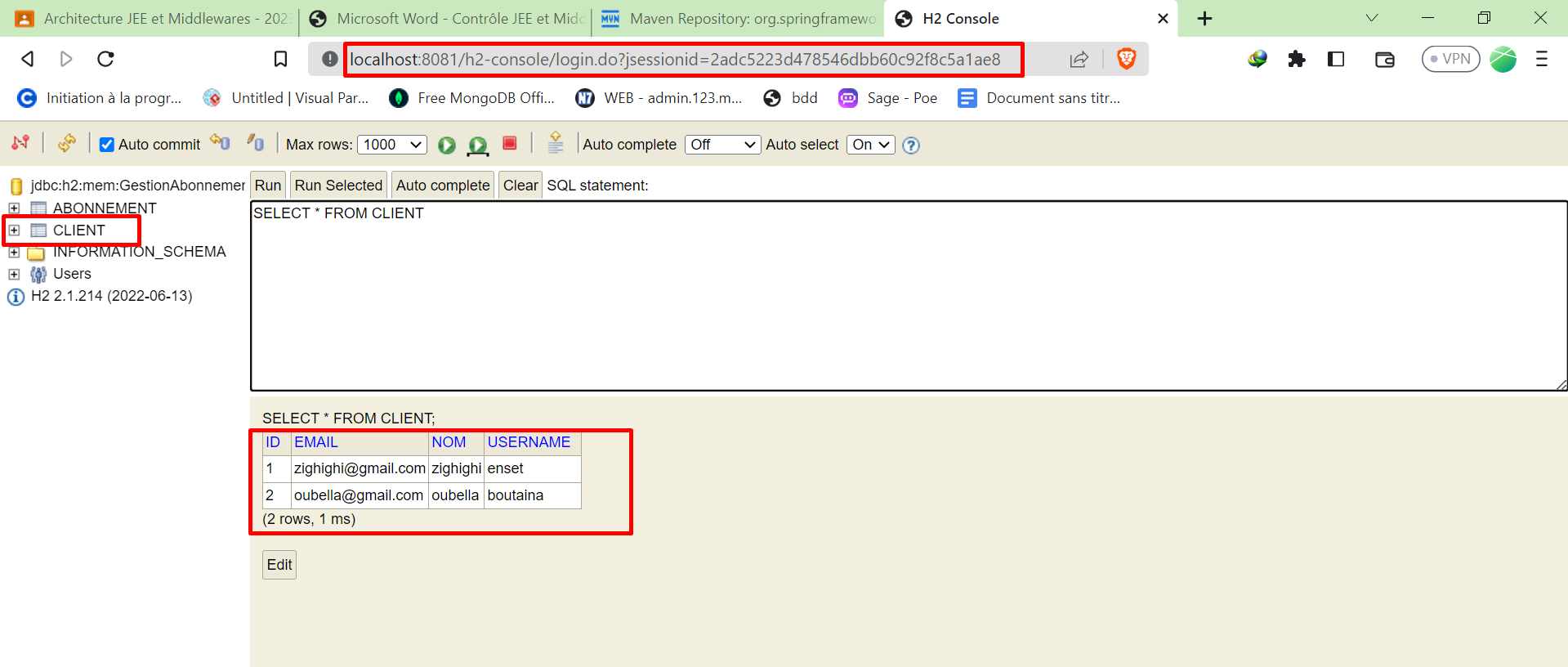
GestionAbonnementsApplication :

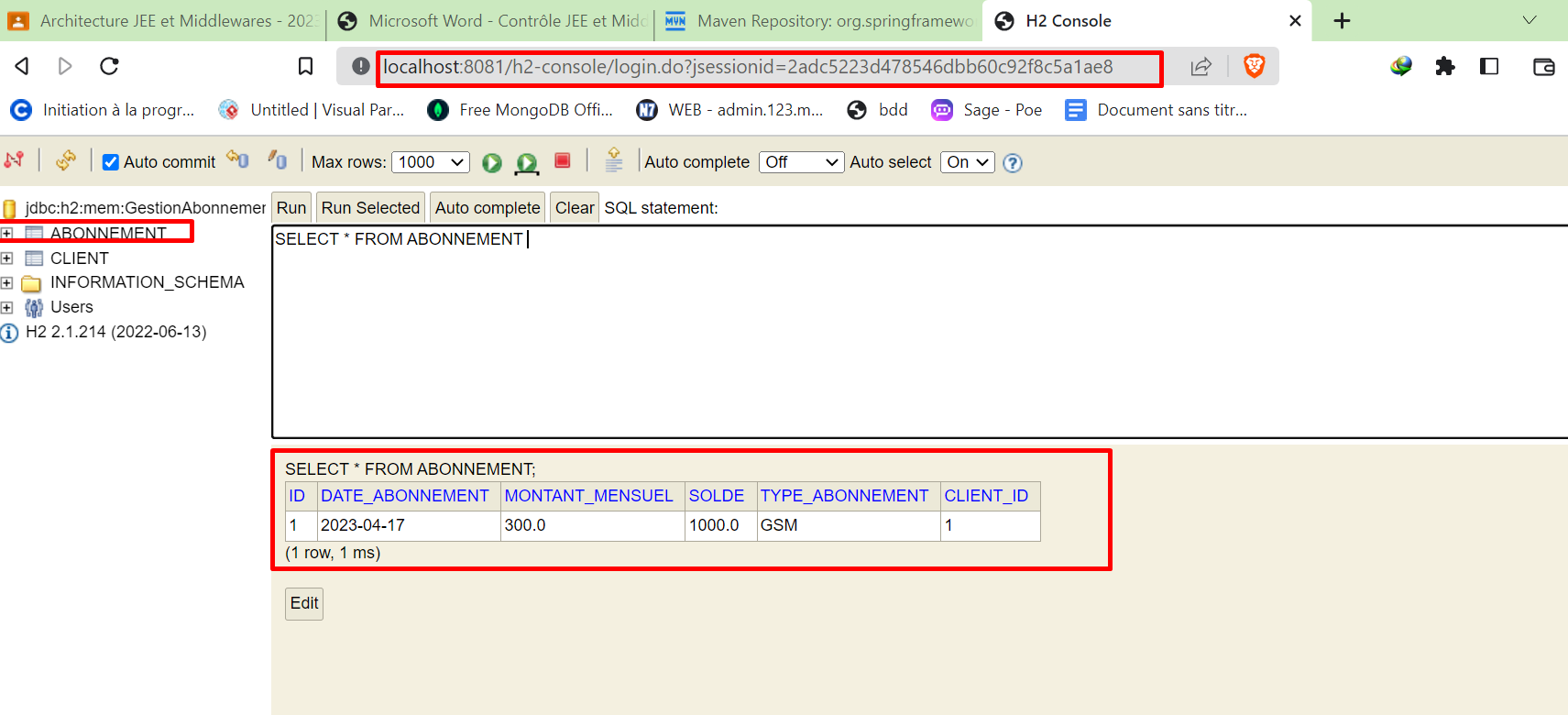
**package** ma.enset.gestion\_abonnements;  
  
**import** ma.enset.gestion\_abonnements.entities.Abonnement;  
**import** ma.enset.gestion\_abonnements.entities.Client;  
**import** ma.enset.gestion\_abonnements.enums.TypeAbonnement;  
**import** ma.enset.gestion\_abonnements.repositories.AbonnementRepository;  
**import** ma.enset.gestion\_abonnements.repositories.ClientRepository;  
**import** org.springframework.boot.CommandLineRunner;  
**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
**import** org.springframework.context.annotation.Bean;  
  
**import** java.util.Date;  
  
@SpringBootApplication  
**public class** GestionAbonnementsApplication {  
  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(GestionAbonnementsApplication.**class**, args);  
 }  
  
 @Bean  
 CommandLineRunner commandLineRunner(AbonnementRepository abonnementRepository, ClientRepository clientRepository){  
 **return** args -> {  
 Client client1 = **new** Client();  
 client1.setNom(**"zighighi"**);  
 client1.setEmail(**"zighighi@gmail.com"**);  
 client1.setUsername(**"enset"**);  
 clientRepository.save(client1);  
 Client client2 = **new** Client();  
 client2.setNom(**"oubella"**);  
 client2.setEmail(**"oubella@gmail.com"**);  
 client2.setUsername(**"boutaina"**);  
 clientRepository.save(client2);  
 Abonnement abonnement1=**new** Abonnement();  
 abonnement1.setClient(client1);  
 abonnement1.setTypeAbonnement(TypeAbonnement.***GSM***);  
 abonnement1.setDate\_abonnement(**new** Date());  
 abonnement1.setSolde(1000);  
 abonnement1.setMontant\_mensuel(300);  
 abonnementRepository.save(abonnement1);  
  
  
 };  
 }  
  
}

Test sur h2 database

application.properties :

**spring.datasource.url**=**jdbc:h2:mem:GestionAbonnement  
spring.h2.console.enabled**=**true****server.port**=**8081  
spring.jpa.hibernate.ddl-auto**=**create  
spring.jpa.show-sql**=**true**

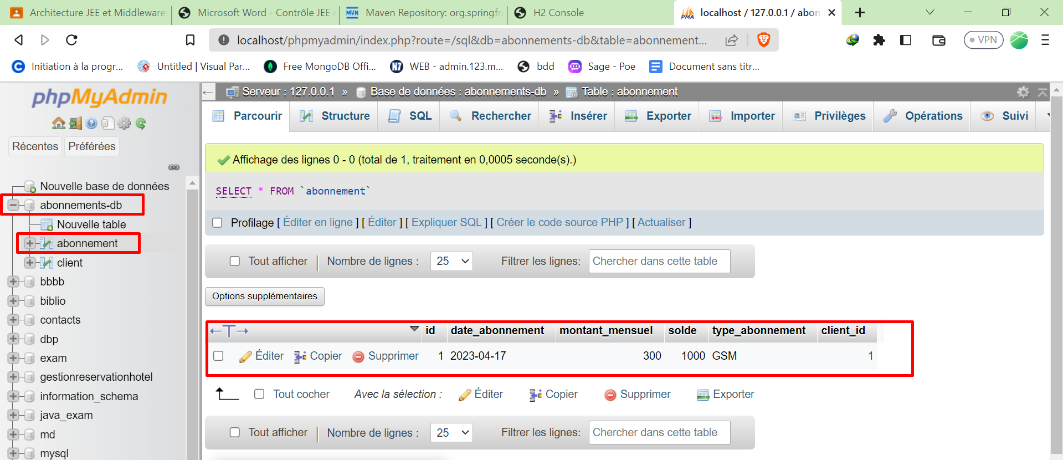


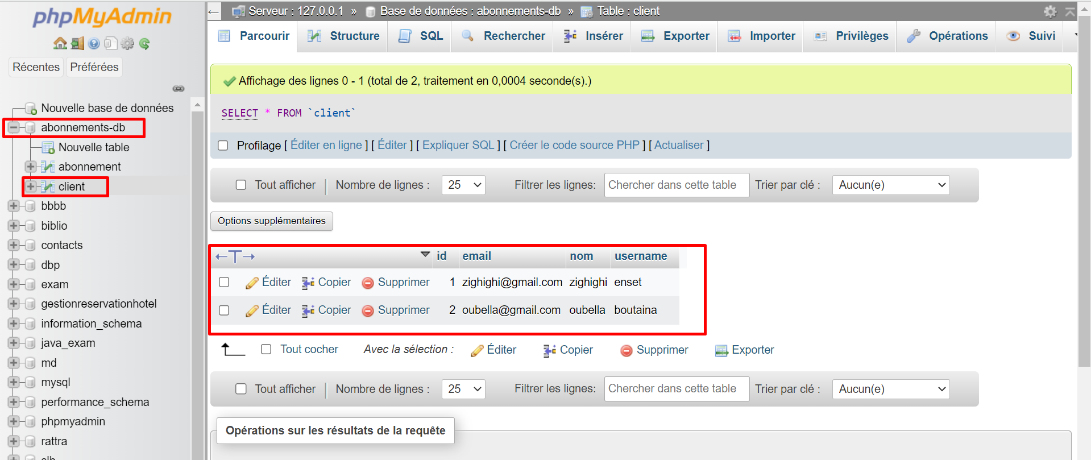


application.properties :

**spring.datasource.url**=**jdbc:mysql://localhost:3306/abonnements-db?createDatabaseIfNotExist=true  
spring.datasource.username**=**root  
spring.datasource.password**=  
**server.port**=**8082  
spring.jpa.hibernate.ddl-auto**=**update  
spring.jpa.properties.hibernate.dialect** = **org.hibernate.dialect.MariaDBDialect  
spring.jpa.show-sql**=**true**

Test sur myqsl database





## **4. Couche Web : Créer une applications Web qui permet de :**

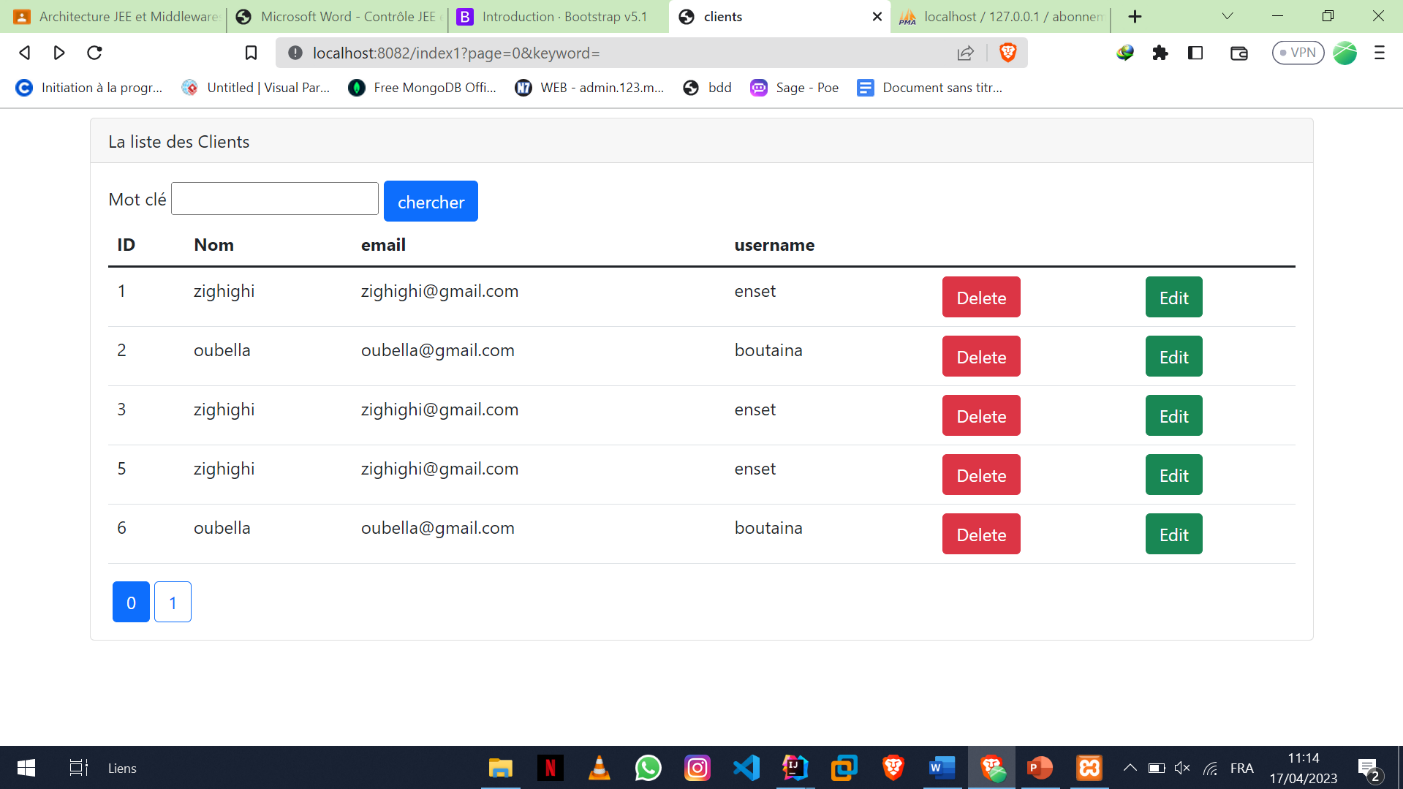
a. Gérer les clients (Chercher, Pagination, Ajout, Edition et Suppression)

ClientController

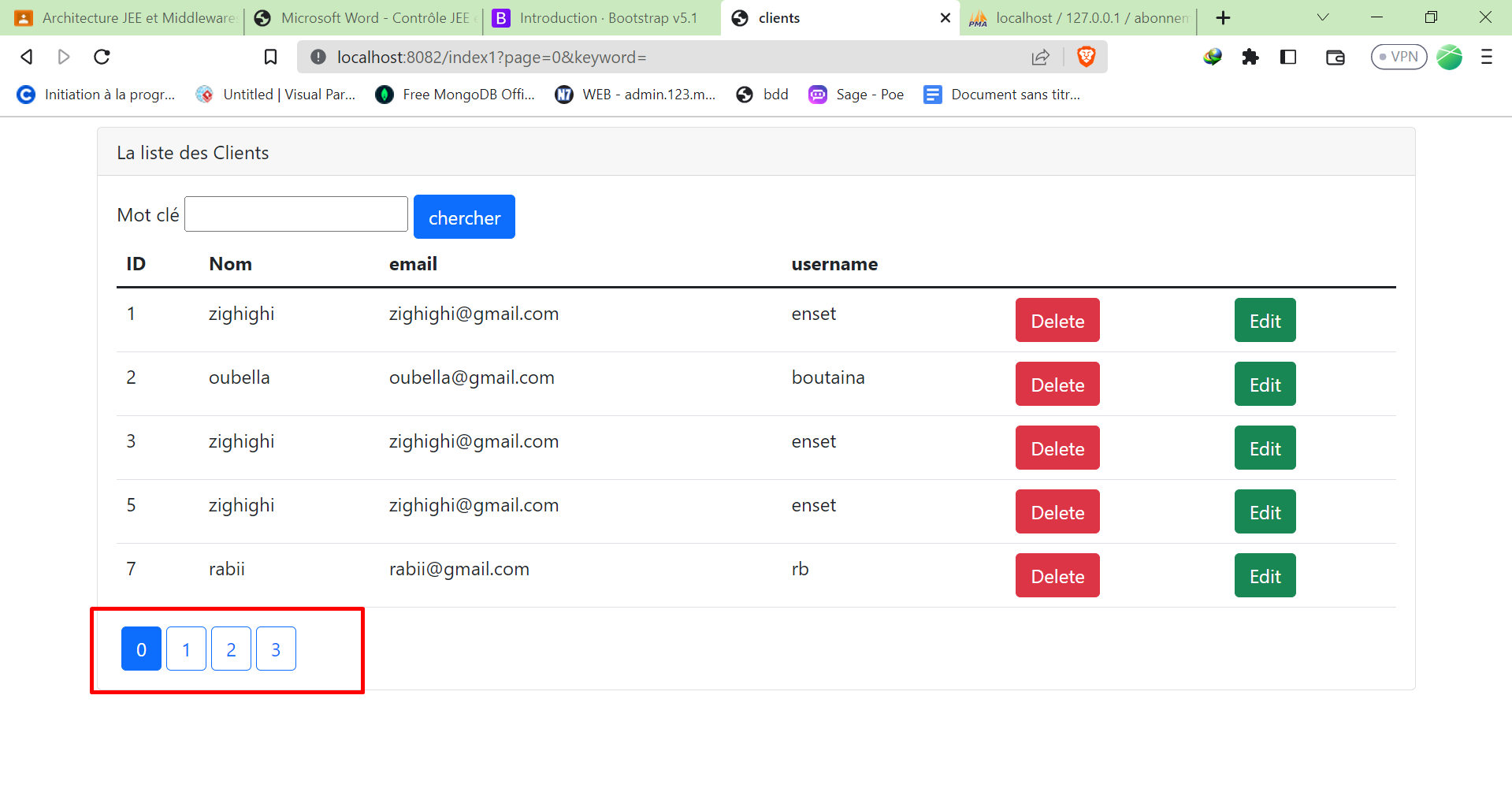
**package** ma.enset.gestion\_abonnements.web;  
  
**import** lombok.AllArgsConstructor;  
**import** ma.enset.gestion\_abonnements.entities.Client;  
**import** ma.enset.gestion\_abonnements.repositories.ClientRepository;  
**import** org.springframework.data.domain.Page;  
**import** org.springframework.data.domain.PageRequest;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.ui.Model;  
**import** org.springframework.validation.BindingResult;  
**import** org.springframework.web.bind.annotation.GetMapping;  
**import** org.springframework.web.bind.annotation.PostMapping;  
**import** org.springframework.web.bind.annotation.RequestParam;  
  
@Controller  
@AllArgsConstructor  
**public class** ClientController  
{  
 **private** ClientRepository **clientRepository**;  
 @GetMapping(**"/index1"**)  
 **public** String clients(Model model ,  
 *//pour la pagenation* @RequestParam(name = **"page"** , defaultValue =**"0"**) **int** page,  
 @RequestParam(name = **"size"** , defaultValue = **"5"**) **int** size,  
 @RequestParam(name = **"keyword"** , defaultValue = **""**) String keyword  
 ){  
 Page<Client> pageClients = **clientRepository**.findByNomContains( keyword, PageRequest.*of*(page, size));  
 model.addAttribute(**"listClients"**,pageClients.getContent());  
 *//pour afficher les pages et sles stocker* model.addAttribute(**"pages"**, **new int**[pageClients.getTotalPages()]);  
 model.addAttribute(**"currentPage"**, page);  
 model.addAttribute(**"keyword"** , keyword);  
 **return "clients"**;  
 }  
 @GetMapping(**"/deleteClient"**)  
 **public** String deleteClient(Long id , String keyword, **int** page)  
 {  
 **clientRepository**.deleteById(id);  
 **return "redirect:/index1?page="**+page+**"&keyword="**+keyword;  
 }  
 @PostMapping(**"/saveClient"**)  
 **public** String saveClient(Model model , Client client, BindingResult bindingResult ,  
 @RequestParam(defaultValue = **"0"**) **int** page,  
 @RequestParam(defaultValue =**""**) String keyword){  
 **if** (bindingResult.hasErrors()) **return "formClients"**;  
 **clientRepository**.save(client);  
 **return "redirect:/index1?page= "**+page+**"&keyword="**+keyword;  
 }

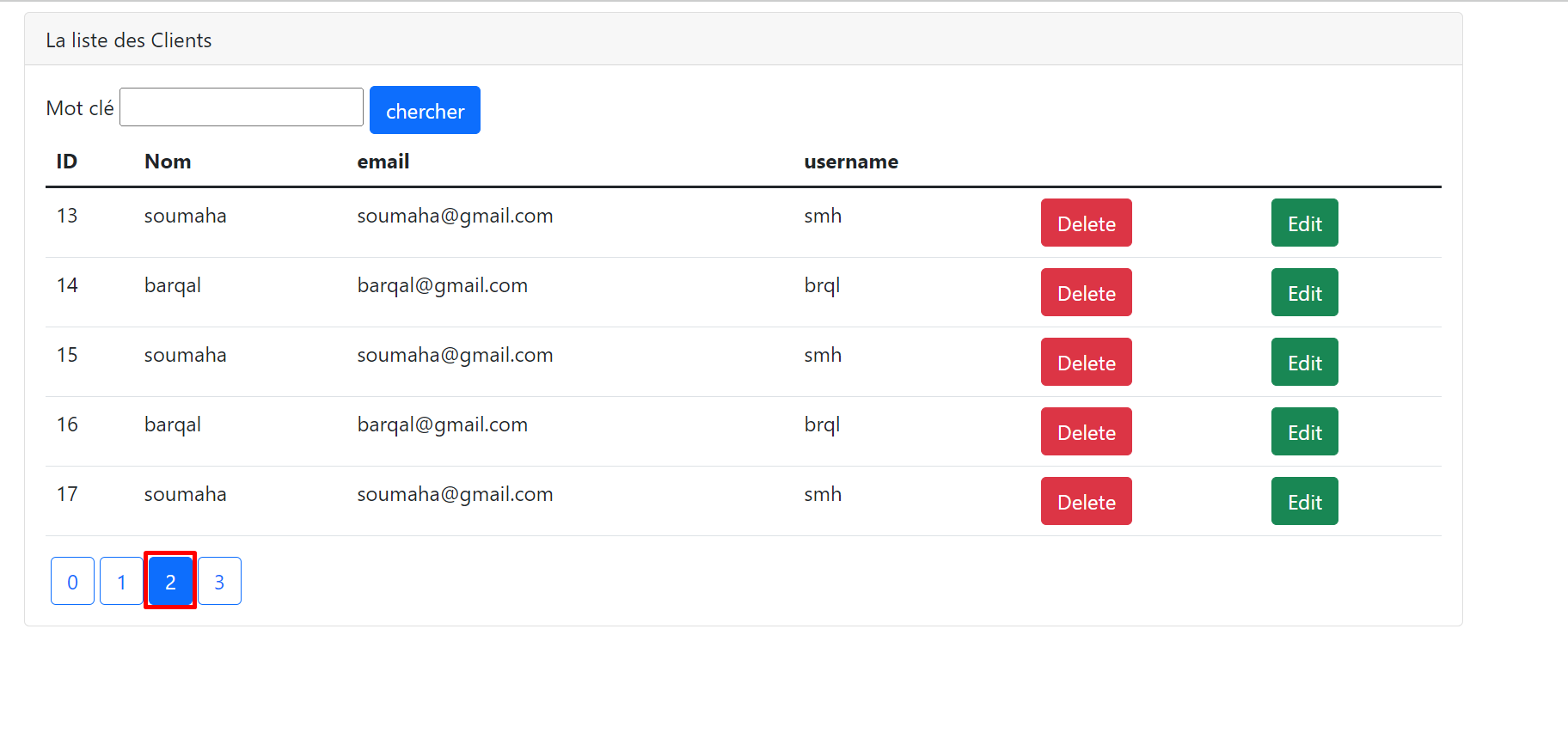
@GetMapping(**"/editClient"**)  
 **public** String editclient(Model model , Long id , String keyword , **int** page){  
 Client client = **clientRepository**.findById(id).orElse(**null**);  
 **if** (client==**null**) **throw new** RuntimeException(**"client introuvable"**);  
 model.addAttribute(**"client"**, client);  
 model.addAttribute(**"page"** , page);  
 model.addAttribute(**"keyword"** , keyword);  
 **return "editClient"**;  
 }  
 @GetMapping(**"/formClient"**)  
 **public** String formClient(Model model){  
 model.addAttribute(**"client"** , **new** Client());  
 **return "formClients"**;  
 }  
}

Test

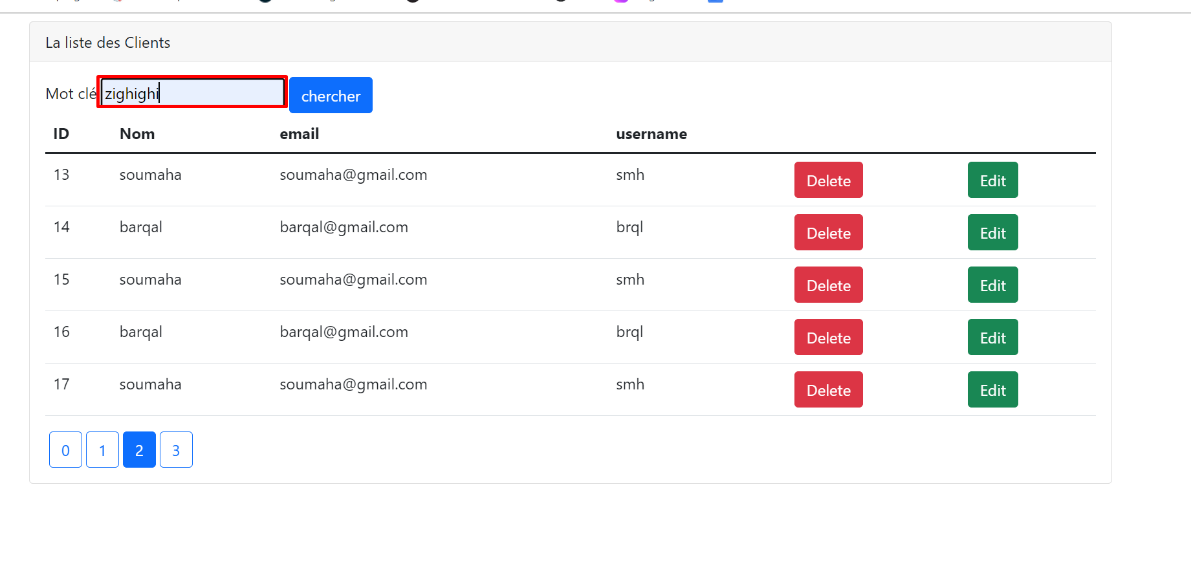
Voilà notre interface

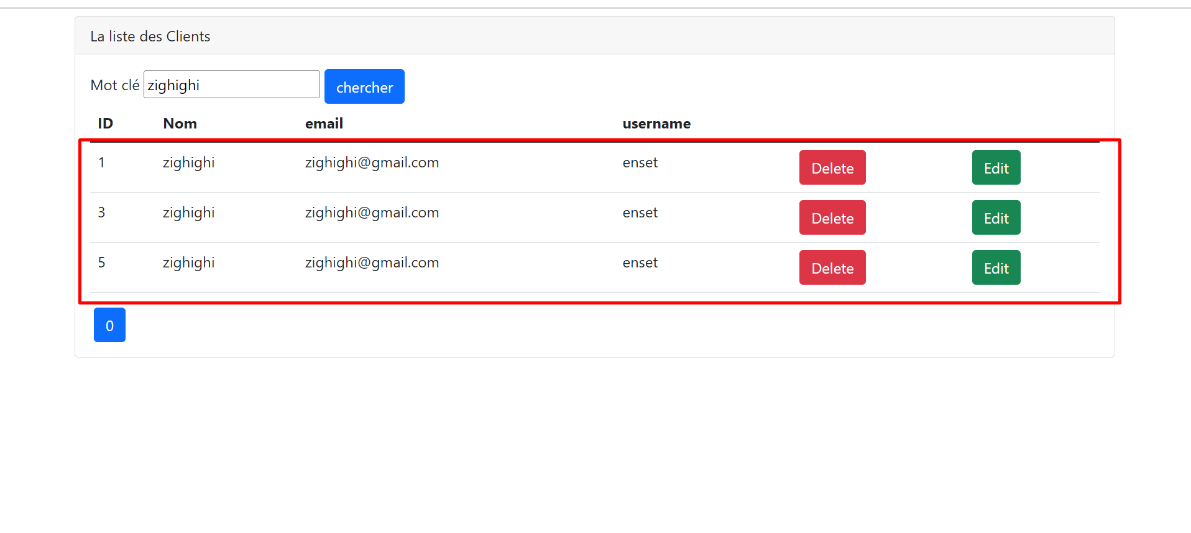
La pagination ça marche



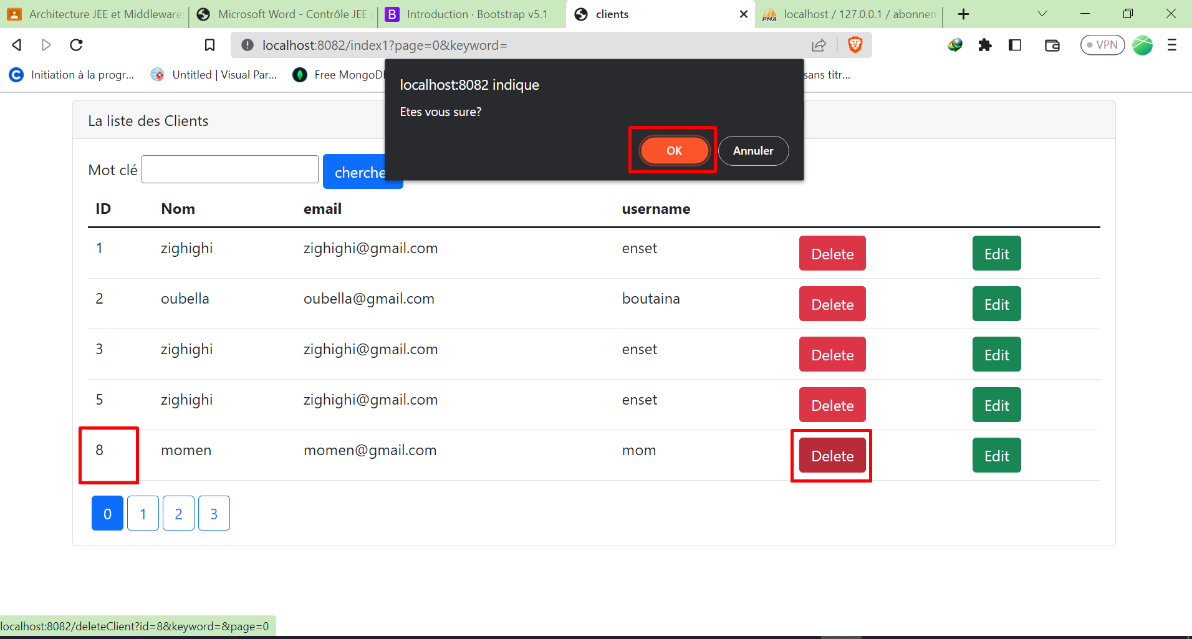
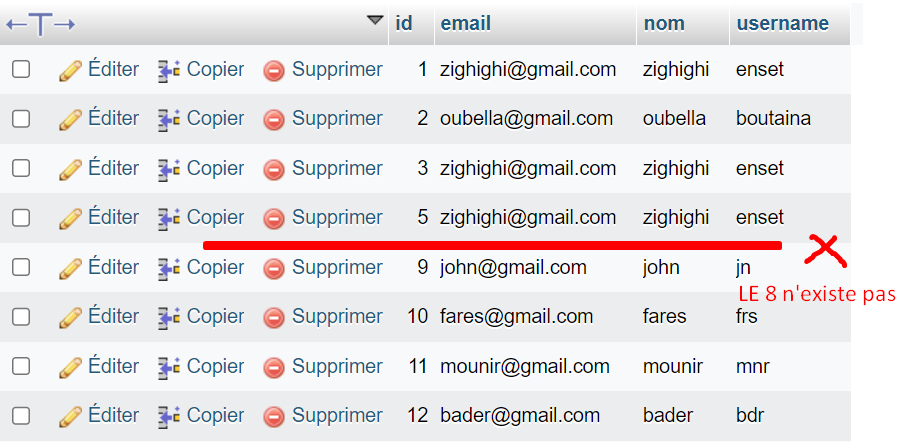


On va tester la recherche



La recherche ça marche

On va tester la suppresion

On supprime le client avec id 8

On vérifie dans la base de données et ça marche