

*Architecture JEE et Middlewares*

***Compte-rendu***

*Contrôle Architecture JEE et Middlmeware*

***Nom et Prénom*** : **BOUSSAIRI Ikram ENSET GLSID – S4**

***E-mail*** : [**ikramboussairi12@gmail.com**](mailto:ikramboussairi12@gmail.com) ***Professeur*** : Mohamed YOUSSFI



**Travail à faire**

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

b. Créer les interfaces JpaRepository basées sur Spring Data

c. Tester la couche DAO

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

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

b. Gérer les abonnements :

o Afficher les abonnements d’un clients

o Charger le solde de l’abonnement avec un montant

o Autres opérations de gestion des abonnements

5. Créer un web service RESTful qui permet de gérer les clients et les abonnements

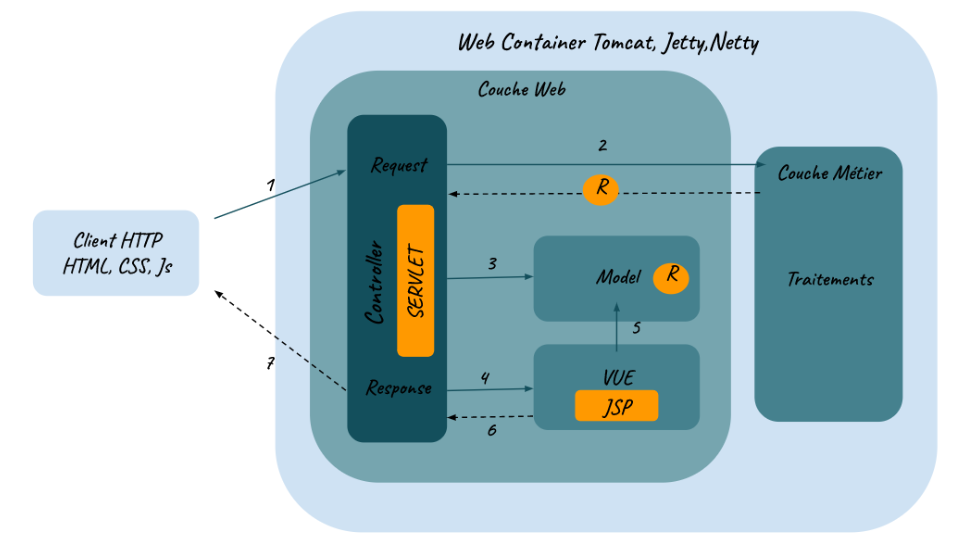
6. Sécurité : Sécuriser l’accès à l’application en développant un système d’authentification statefull basé sur Spring Security avec deux rôles CLIENT et ADMIN. l’application doit répondre aux critères suivants :

a. Authentification avec le rôle CLIENT : le client ne peut voir que : son profile, ses abonnements et peut charger ses abonnements

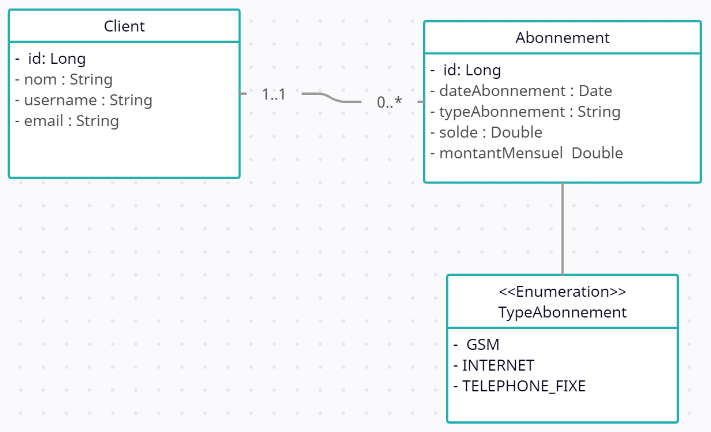
b. Authentification avec le rôle ADMIN : l’administrateur peut gérer les clients et les abonnements avec tous les droits possibles. En plus il peut créer de nouveau utilisateurs et affecter des rôles aux utilisateurs.

**Réalisation**

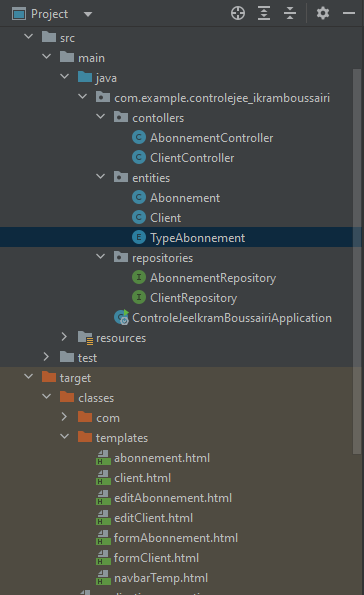
1. Schéma de l’architecture technique de l’application



2. Diagramme de classe représentant les données manipulées par l’application

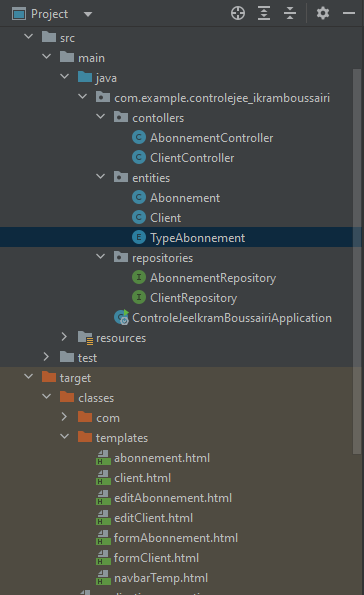


**Structure de projet**

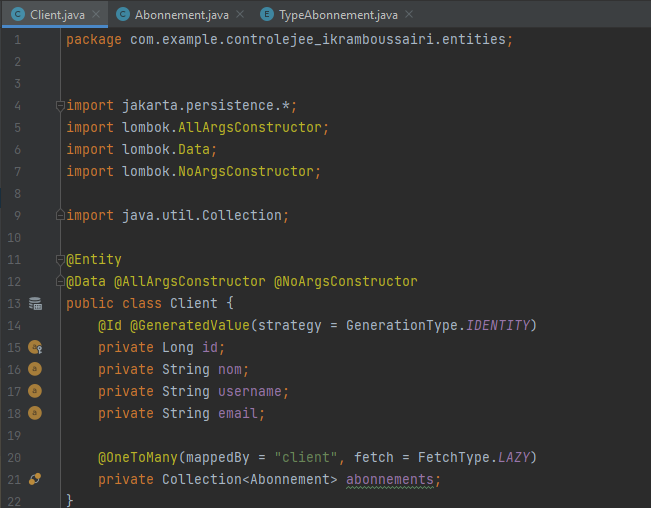


3. Couche DAO :

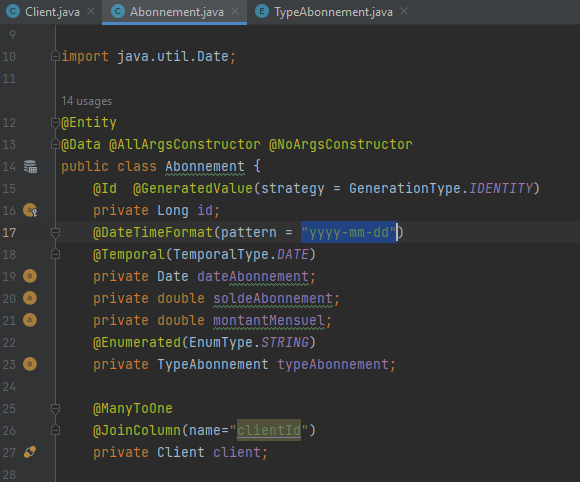
a. Créer les entités JPA



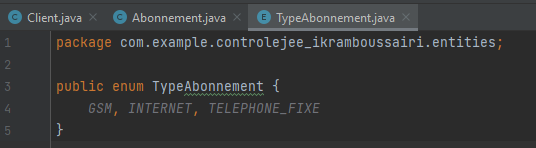
**Client :**



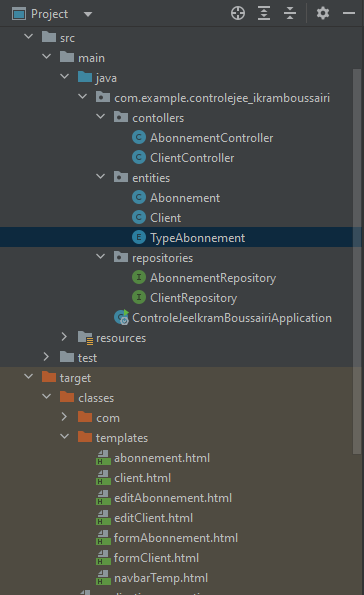
**Abonnement :**

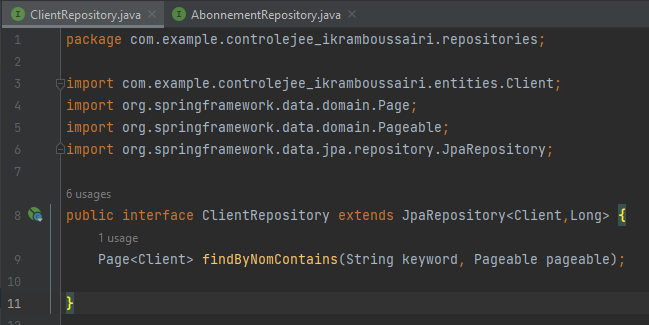


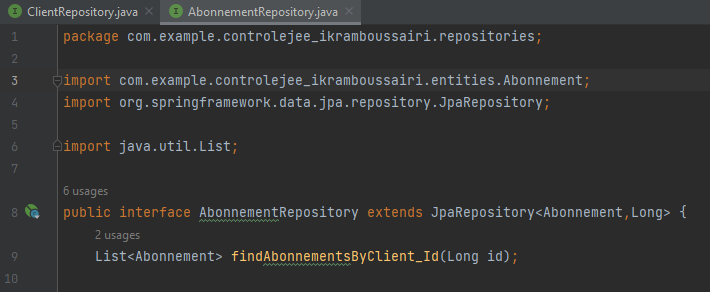
**TypeAbonnement :**



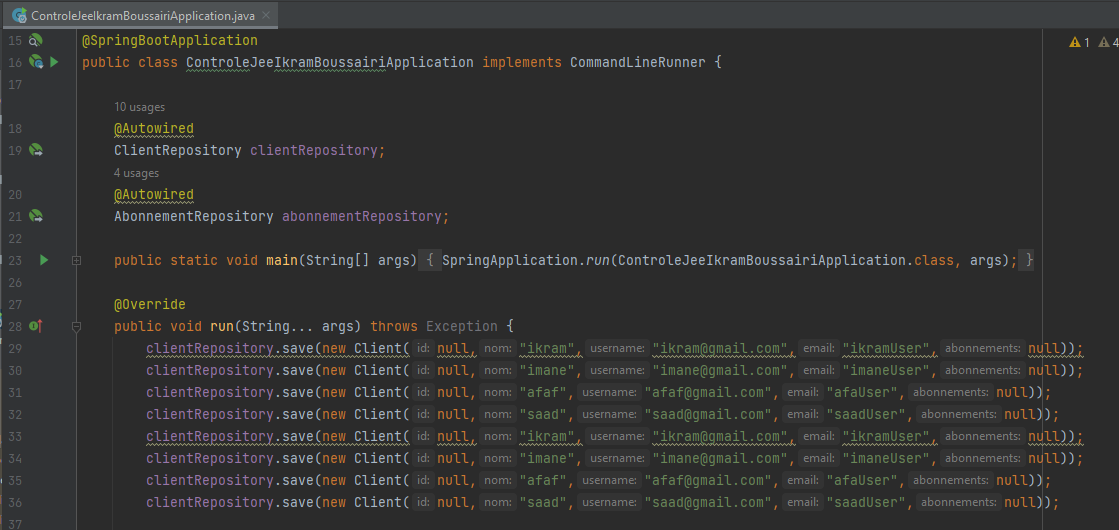
b. Créer les interfaces JpaRepository basées sur Spring Data

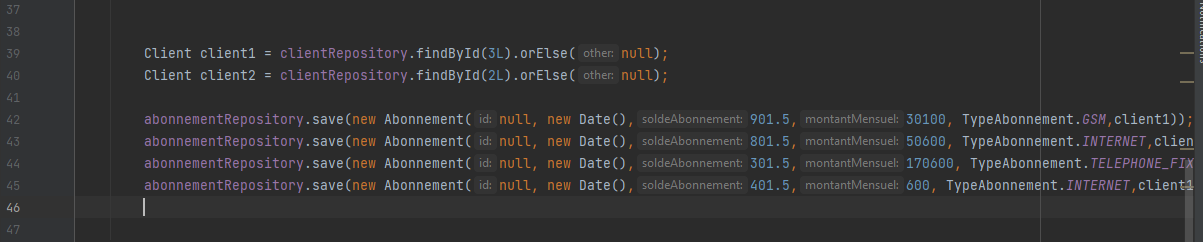


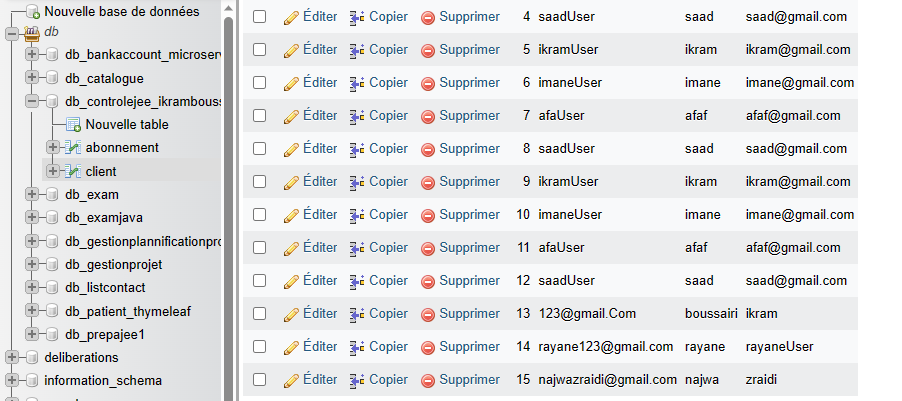




c. Tester la couche DAO









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

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

**Controller**

package com.example.controlejee\_ikramboussairi.contollers;  
  
import com.example.controlejee\_ikramboussairi.entities.Abonnement;  
import com.example.controlejee\_ikramboussairi.entities.Client;  
import com.example.controlejee\_ikramboussairi.repositories.AbonnementRepository;  
import com.example.controlejee\_ikramboussairi.repositories.ClientRepository;  
import jakarta.validation.Valid;  
import org.springframework.beans.factory.annotation.Autowired;  
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;  
  
import java.util.List;  
  
  
@Controller  
public class ClientController {  
 @Autowired  
 private ClientRepository clientRepository;  
  
 @Autowired  
 private AbonnementRepository abonnementRepository;  
  
 @GetMapping("/index")  
 public String ressources(Model model,  
 @RequestParam(name="page", defaultValue = "0") int page,  
 @RequestParam(name="size", defaultValue = "5") int size,  
 @RequestParam(name="keyword", defaultValue = "") String keyword){  
 Page<Client> pageClient = clientRepository.findByNomContains(keyword,PageRequest.*of*(page,size));  
 model.addAttribute("listClient", pageClient.getContent());  
 model.addAttribute("pages", new int[pageClient.getTotalPages()]);  
 model.addAttribute("currentPage",page);  
 model.addAttribute("keyword",keyword);  
  
 return "client";  
 }  
  
 @GetMapping("/")  
 public String home(){  
 return "redirect:/index";  
 }  
  
 @GetMapping("/delete")  
 public String delete(@RequestParam Long id, String keyword, int page){  
  
 List<Abonnement> abonnements = abonnementRepository.findAbonnementsByClient\_Id(id);  
 for (Abonnement a : abonnements ) {  
 abonnementRepository.delete(a);  
 }  
  
 clientRepository.deleteById(id);  
 return "redirect:/index?page="+page+"&keyword="+keyword;  
 }  
  
 @GetMapping("/formClient")  
 public String formRessources(Model model){  
 model.addAttribute("client", new Client());  
 return "formClient";  
 }  
  
 @PostMapping("/save")  
 public String save(Model model, @Valid Client client, BindingResult bindingResult,  
 @RequestParam(defaultValue = "0") int page,  
 @RequestParam(defaultValue = "") String keyword){  
 if(bindingResult.hasErrors()) return "formPatients";  
  
 clientRepository.save(client);  
 return "redirect:/index?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("Patient introuvable");  
  
 model.addAttribute("client",client);  
 model.addAttribute("page",page);  
 model.addAttribute("keyword",keyword);  
  
 return "editClient";  
 }  
  
}

**Templates**

* **Client**

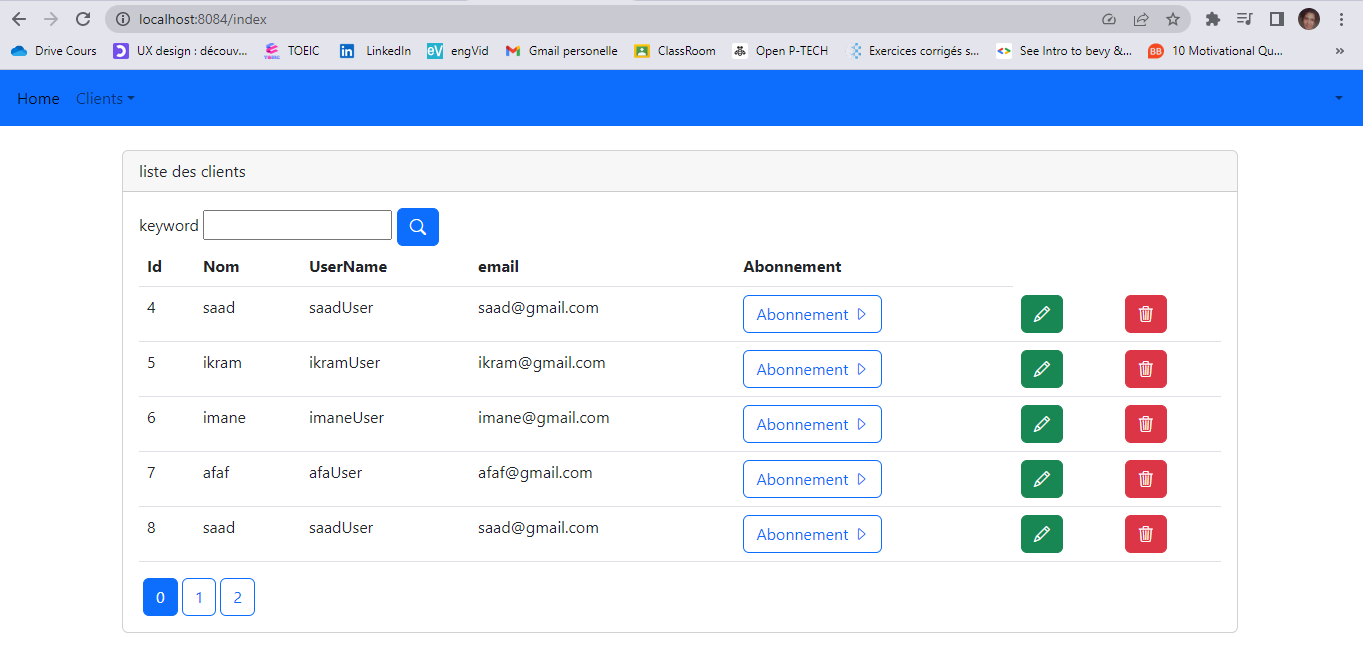
<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultrap.net.nz/thymeleaf/layout"  
 layout:decorate="navbarTemp">  
  
<head>  
 <meta charset="UTF-8">  
 <title>Client</title>  
 <Link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">  
 <Link rel="stylesheet" href="/webjars/bootstrap-icons/1.10.3/font/bootstrap-icons.css">  
  
</head>  
<body>  
<div layout:fragment="content1">  
<div>  
 <div class="container mt-4">  
 <div class="card">  
 <div class="card-header">  
 liste des clients  
 </div>  
 <div class="card-body">  
 <form method="get" th:action="@{/index}">  
 <label>keyword</label>  
 <input type="text" name="keyword" th:value="${keyword}">  
 <button type="submit" class="btn btn-primary"><i class="bi bi-search"></i></button>  
 </form>  
  
 <table class="table">  
 <thead>  
 <tr>  
 <th>Id</th>  
 <th>Nom</th>  
 <th>UserName</th>  
 <th>email</th>  
 <th>Abonnement</th>  
  
 </tr>  
 </thead>  
 <tbo>  
 <tr th:each="c:${listClient}">  
 <td th:text="${c.getId()}"></td>  
 <td th:text="${c.getNom()}"></td>  
 <td th:text="${c.getEmail()}"></td>  
 <td th:text="${c.getUsername()}"></td>  
 <td>  
 <a th:href="@{/abonnement(id=${c.id})}" type="button" class="btn btn-outline-primary">Abonnement <i class="bi bi-caret-right"></i></a>  
 </td>  
  
 <td>  
 <a th:href="@{/editClient(id=${c.getId()}, keyword=${keyword}, page=${currentPage})}" class="btn btn-success"><i class="bi bi-pencil"></i></a>  
 </td>  
  
 <td>  
 <a onclick="return confirm('etes vous sure?')"  
 th:href="@{/delete(id=${c.id}, keyword=${keyword}, page=${currentPage})}" class="btn btn-danger"><i class="bi bi-trash3"></i></a>  
 </td>  
  
 </tr>  
 </tbo>  
 </table>  
 <ul class="nav nav-pills">  
 <li th:each="page,status:${pages}">  
 <a th:class="${status.index==currentPage?'btn btn-primary ms-1' : 'btn btn-outline-primary ms-1'}"  
 th:text="${status.index}"  
 th:href="@{index(page=${status.index}, keyword=${keyword})}"></a>  
 </li>  
 </ul>  
 </div>  
 </div>  
 </div>  
</div>  
</div>  
</body>  
</html>

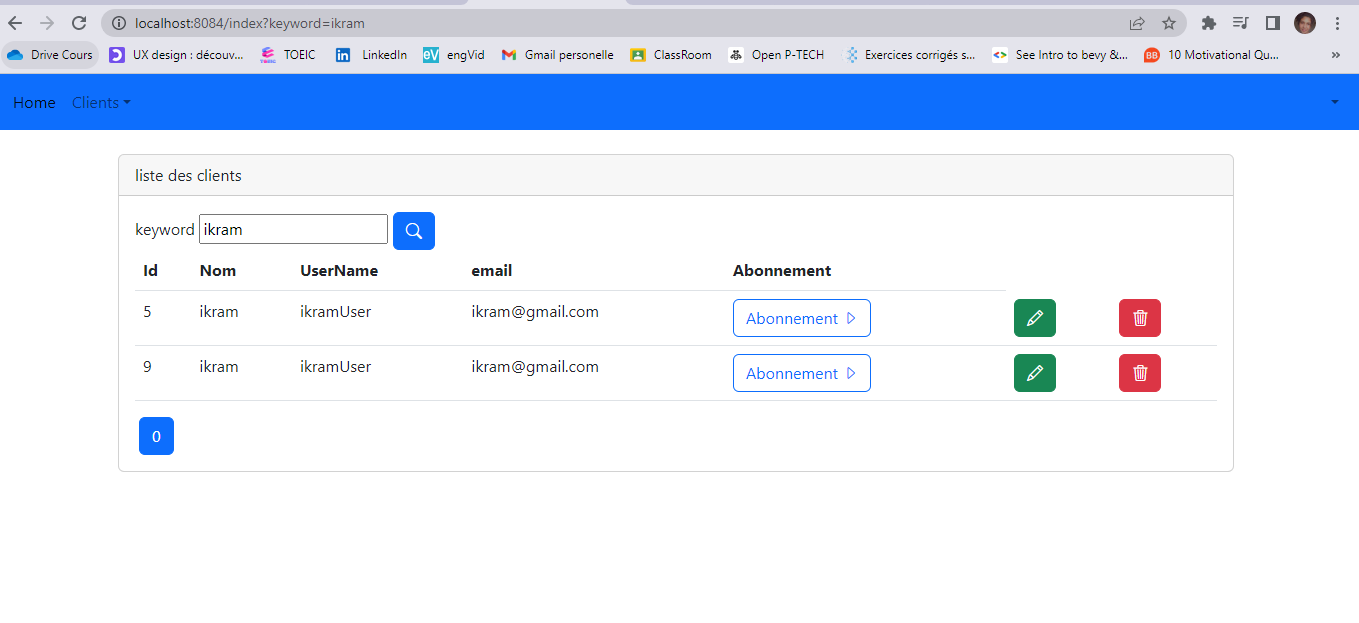
* **Edit Client**

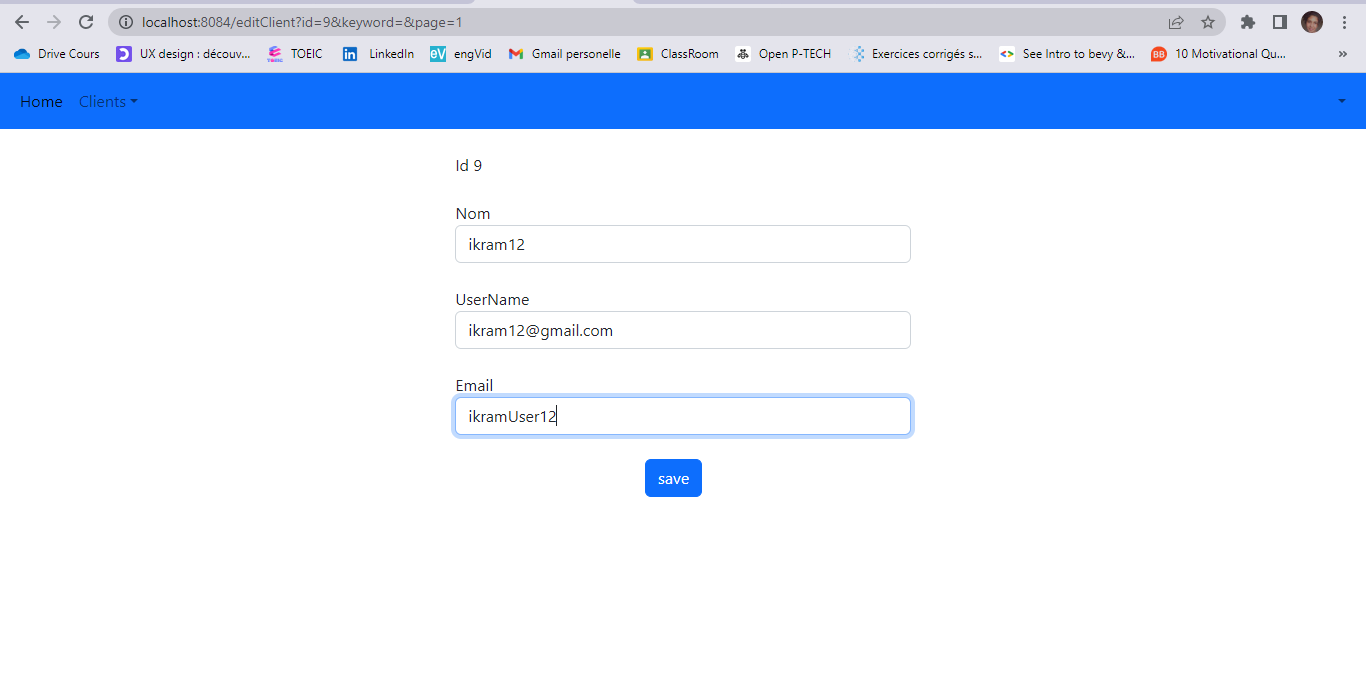
<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultrap.net.nz/thymeleaf/layout"  
 layout:decorate="navbarTemp">  
  
<head>  
 <meta charset="UTF-8">  
 <title>edit Client</title>  
 <Link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">  
</head>  
<body>  
  
<div layout:fragment="content1">  
 <div class="col-md-4 offset-4 mt-4">  
 <form method="post" th:action="@{/save(page=${page}, keyword=${keyword})}">  
  
 <div class="mt-4">  
 <label>Id </label>  
 <label th:text="${client.id}"> </label>  
 <input class="form-control" type="hidden" name="id" th:value="${client.id}">  
 </div>  
  
 <div class="mt-4">  
 <label>Nom</label>  
 <input class="form-control" type="text" name="nom" th:value="${client.nom}">  
 <span class="text-danger" th:errors="${client.nom}"></span>  
 </div>  
 <div class="mt-4">  
 <label>UserName</label>  
 <input class="form-control" type="text" name="username" th:value="${client.username}">  
 <span class="text-danger" th:errors="${client.username}"></span>  
 </div>  
 <div class="mt-4">  
 <label>Email</label>  
 <input class="form-control" type="text" name="email" th:value="${client.email}">  
 <span class="text-danger" th:errors="${client.email}"></span>  
 </div>  
  
  
 <div class="offset-5 mt-4">  
 <button type="submit" class="btn btn-primary "> save</button>  
 </div>  
 </form>  
 </div>  
</div>  
</body>  
</html>

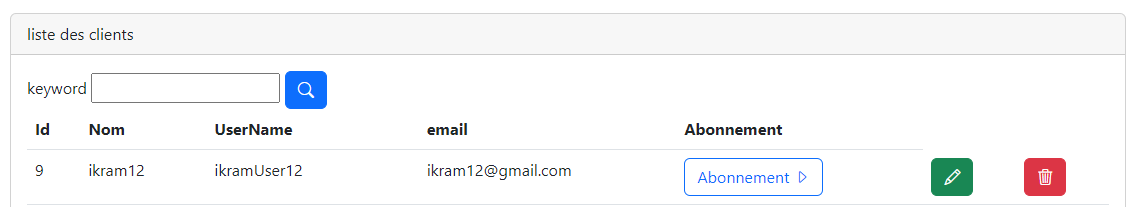
* **Form Client**

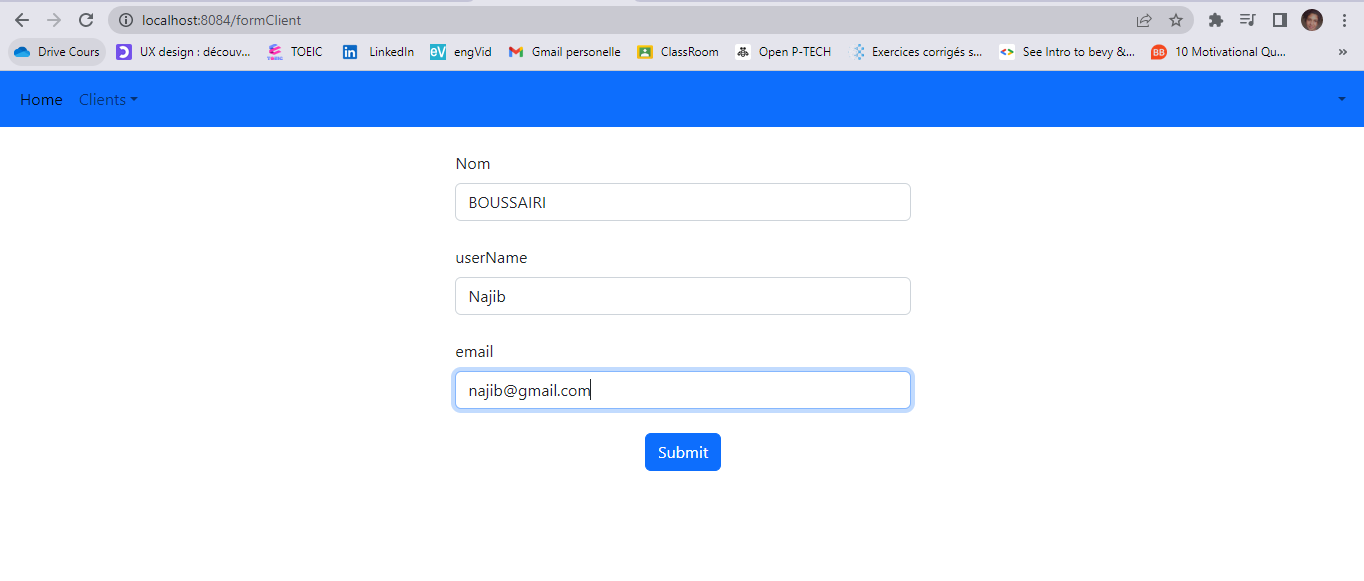
<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultrap.net.nz/thymeleaf/layout"  
 layout:decorate="navbarTemp">  
  
<head>  
 <meta charset="UTF-8">  
 <title>Form Client</title>  
 <Link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">  
</head>  
<body>  
  
<div layout:fragment="content1">  
 <div class="col-md-4 offset-4 mt-4">  
 <form method="post" th:action="@{/save}">  
 <div class="mt-4">  
 <label class="form-label">Nom</label>  
 <input class="form-control" type="text" name="nom" th:value="${client.nom}">  
 <span class="text-danger form-text" th:errors="${client.nom}"></span>  
 </div>  
  
 <div class="mt-4">  
 <label class="form-label">userName</label>  
 <input class="form-control" type="text" name="username" th:value="${client.username}">  
 <span class="text-danger form-text" th:errors="${client.username}"></span>  
 </div>  
  
 <div class="mt-4">  
 <label class="form-label">email</label>  
 <input class="form-control" type="email" name="email" th:value="${client.email}">  
 <span class="text-danger form-text" th:errors="${client.email}"></span>  
 </div>  
  
 <div class="offset-5 mt-4">  
 <button type="submit" class="btn btn-primary "> Submit</button>  
 </div>  
 </form>  
 </div>  
</div>  
</body>  
</html>

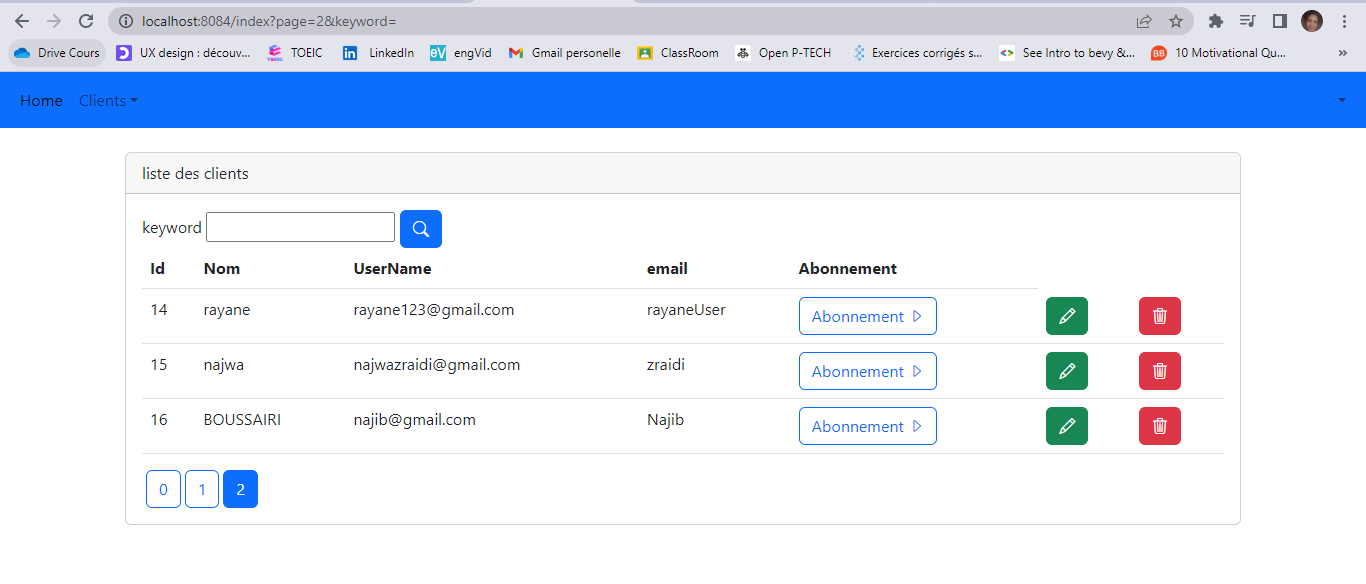












b. Gérer les abonnements :

**Controller**

package com.example.controlejee\_ikramboussairi.contollers;  
  
import com.example.controlejee\_ikramboussairi.entities.Abonnement;  
import com.example.controlejee\_ikramboussairi.entities.Client;  
  
import com.example.controlejee\_ikramboussairi.repositories.AbonnementRepository;  
import com.example.controlejee\_ikramboussairi.repositories.ClientRepository;  
import jakarta.validation.Valid;  
  
import org.springframework.beans.factory.annotation.Autowired;  
  
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;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.util.Date;  
import java.util.List;  
  
@Controller  
public class AbonnementController {  
  
 Long id\_Client;  
  
 @Autowired  
 private AbonnementRepository abonnementRepository;  
 @Autowired  
 private ClientRepository clientRepository;  
  
 @GetMapping("/abonnement/{id}")  
 public String abonnementRessources(Model model,@RequestParam Long id){  
  
 List<Abonnement> abonnements = abonnementRepository.findAbonnementsByClient\_Id(id);  
 model.addAttribute("listAbonnement", abonnements);  
  
 id\_Client = id;  
  
 return "abonnement";  
 }  
  
  
 @PostMapping("/saveAbonnement/{id}")  
 public String saveAbonnement(Model model, @Valid Abonnement abonnement,@RequestParam Long clientId, BindingResult bindingResult){  
 Client client = clientRepository.findById(clientId).orElse(null);  
  
 if(client!=null) abonnement.setClient(client);  
  
 abonnementRepository.save(abonnement);  
 return "redirect:/abonnement?id="+clientId;  
 }  
  
  
 @GetMapping("/editAbonnement/{id}")  
 public String editAbonnement(Model model,@RequestParam Long id){  
 Abonnement abonnement = abonnementRepository.findById(id).orElse(null);  
 if(abonnement==null) throw new RuntimeException("Abonnement introuvable");  
  
 model.addAttribute("abonnement",abonnement);  
  
 return "editAbonnement";  
 }  
  
 @GetMapping("/formAbonnement")  
 public String formRessources(Model model){  
 Abonnement abonnement = new Abonnement();  
  
 abonnement.setDateAbonnement(new Date());  
  
 Client client = clientRepository.findById(id\_Client).orElse(null);  
 abonnement.setClient(client);  
  
 model.addAttribute("abonnement", abonnement);  
 return "formAbonnement";  
 }  
  
 @GetMapping("/deleteAbonnement/{id}")  
 public String deleteAbonnement(@RequestParam Long id){  
  
 abonnementRepository.deleteById(id);  
 return "redirect:/abonnement?id=" + id\_Client;  
 }  
  
}

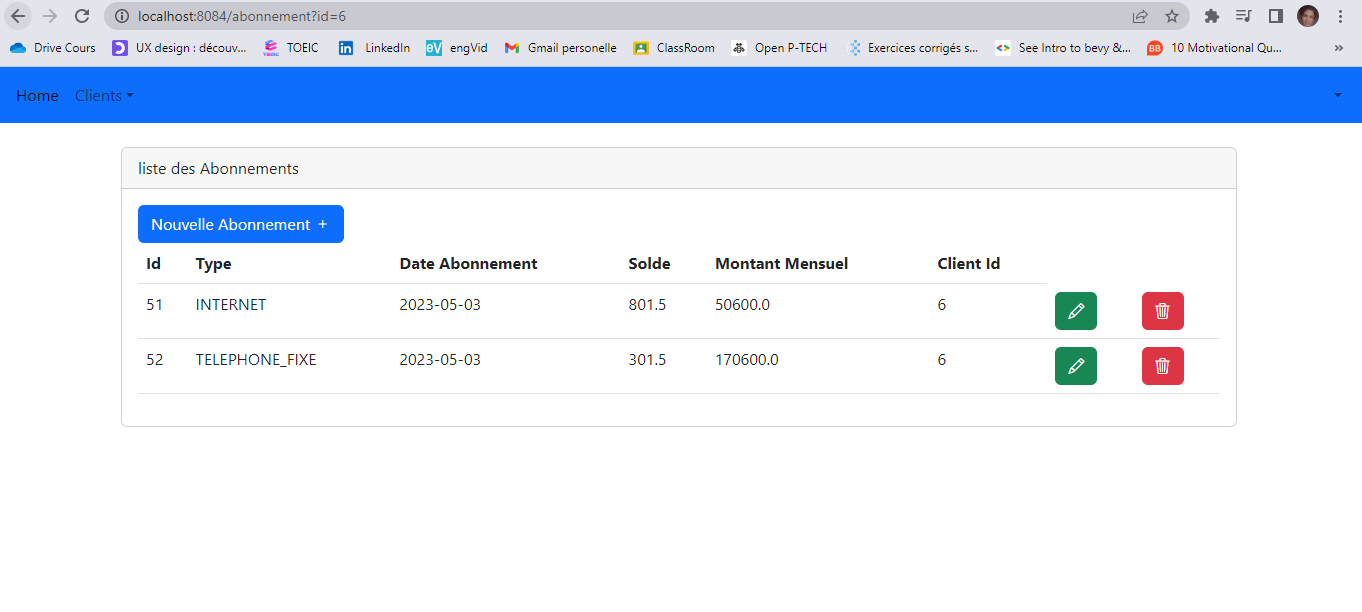
**Templtes**

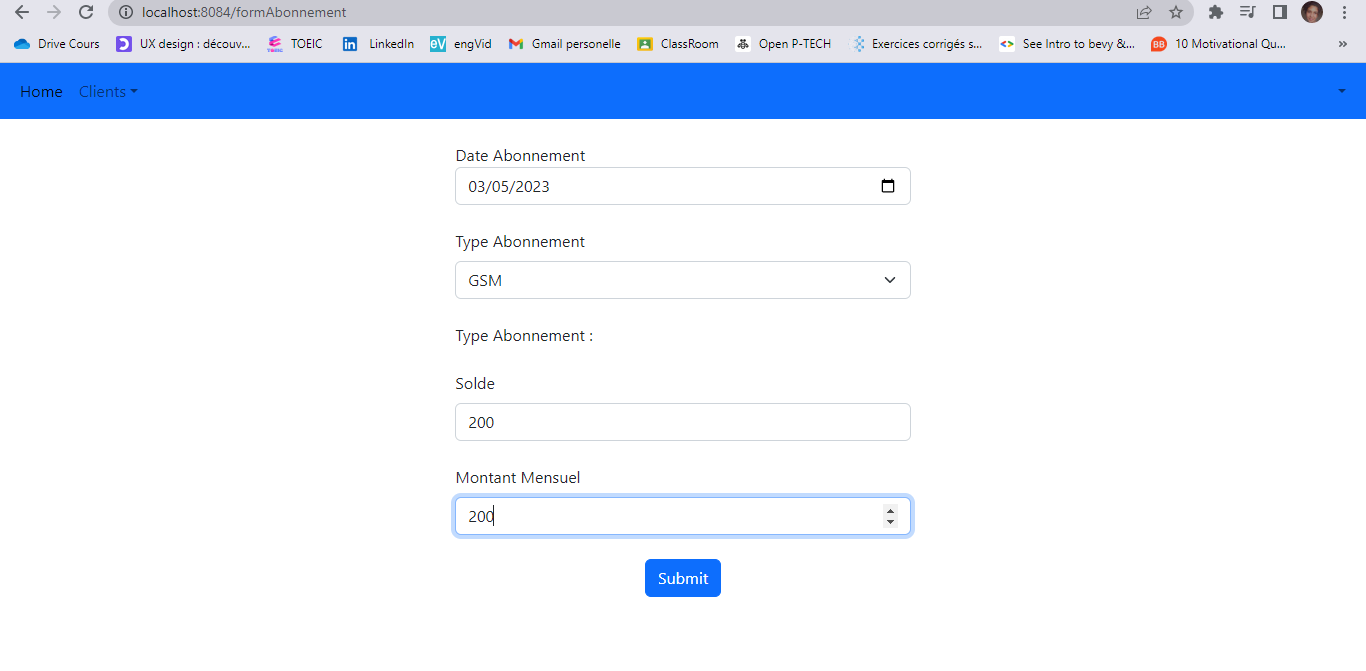
* **Abonnement**
* <!DOCTYPE html>  
  <html lang="en" xmlns:th="http://www.thymeleaf.org"  
   xmlns:layout="http://www.ultrap.net.nz/thymeleaf/layout"  
   layout:decorate="navbarTemp">  
    
  <head>  
   <meta charset="UTF-8">  
   <title>Abonnement</title>  
   <Link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">  
   <Link rel="stylesheet" href="/webjars/bootstrap-icons/1.10.3/font/bootstrap-icons.css">  
    
  </head>  
  <body>  
  <div layout:fragment="content1">  
  <div>  
   <div class="container mt-4">  
   <div class="card">  
   <div class="card-header">  
   liste des Abonnements  
   </div>  
    
   <div class="card-body">  
    
   <div>  
   <a th:href="@{/formAbonnement}" type="button" class="btn btn-primary">Nouvelle Abonnement <i class="bi bi-plus"></i></a>  
   </div>  
    
   <table class="table">  
   <thead>  
   <tr>  
   <th>Id</th>  
   <th>Type </th>  
   <th>Date Abonnement</th>  
   <th>Solde</th>  
   <th>Montant Mensuel</th>  
   <th>Client Id </th>  
    
   </tr>  
   </thead>  
   <tbo>  
   <tr th:each="a:${listAbonnement}">  
   <td th:text="${a.getId()}"></td>  
   <td th:text="${a.getTypeAbonnement().toString()}"></td>  
   <td th:text="${a.getDateAbonnement()}"></td>  
   <td th:text="${a.getSoldeAbonnement()}"></td>  
   <td th:text="${a.getMontantMensuel()}"></td>  
   <td th:text="${a.getClient().getId()}"></td>  
    
    
   <td>  
   <a th:href="@{/editAbonnement(id=${a.getId()})}" class="btn btn-success"><i class="bi bi-pencil"></i></a>  
   </td>  
   <td>  
   <a th:href="@{/deleteAbonnement(id=${a.getId()})}" onclick="return confirm('etes vous sure?')"  
   class="btn btn-danger"><i class="bi bi-trash3"></i></a>  
   </td>  
    
   </tr>  
   </tbo>  
   </table>  
    
   <!-- <ul class="nav nav-pills">  
   <li th:each="page,status:${pages}">  
   <a th:class="${status.index==currentPage?'btn btn-primary ms-1' : 'btn btn-outline-primary ms-1'}"  
   th:text="${status.index}"  
   th:href="@{abonnement(page=${status.index}, keyword=${keyword})}"></a>  
   </li>  
   </ul>-->  
    
   </div>  
   </div>  
   </div>  
  </div>  
  </div>  
  </body>  
  </html>
* **Edit Abonnement**

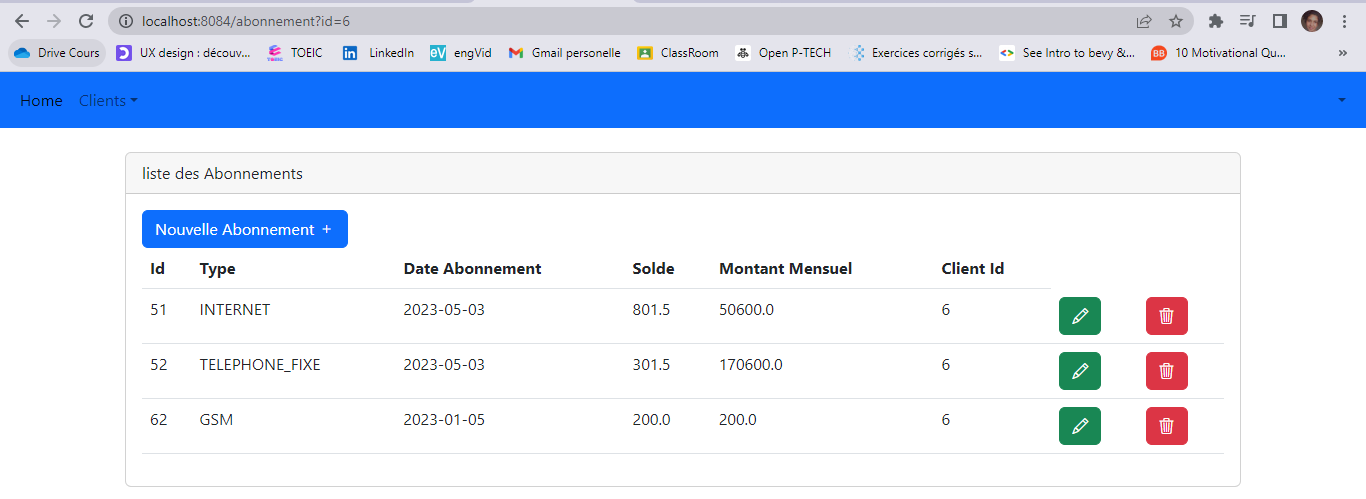
<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultrap.net.nz/thymeleaf/layout"  
 layout:decorate="navbarTemp">  
  
<head>  
 <meta charset="UTF-8">  
 <title>edit Abonnement</title>  
 <Link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">  
</head>  
<body>  
  
<div layout:fragment="content1">  
 <div class="col-md-4 offset-4 mt-4">  
 <form method="post" th:action="@{/saveAbonnement(clientId=${abonnement.getClient().getId()})}">  
  
 <div class="mt-4">  
 <label>Id : </label>  
 <label th:text="${abonnement.id}"> </label>  
 <input class="form-control" type="hidden" name="id" th:value="${abonnement.id}">  
 </div>  
  
 <div class="mt-4">  
 <label>Type Abonnement : </label>  
 <label th:text="${abonnement.typeAbonnement}"> </label>  
 <input class="form-control" type="hidden" name="typeAbonnement" th:value="${abonnement.typeAbonnement}">  
 </div>  
  
 <div class="mt-4">  
 <label>Date Abonnement : </label>  
 <label th:text="${abonnement.dateAbonnement.toString()}"> </label>  
 <input class="form-control" type="hidden" name="dateAbonnement" th:value="${abonnement.dateAbonnement}">  
 </div>  
  
 <div class="mt-4">  
 <label>Solde</label>  
 <input class="form-control" type="number" step="any" name="soldeAbonnement" th:value="${abonnement.soldeAbonnement}">  
 <span class="text-danger" th:errors="${abonnement.soldeAbonnement}"></span>  
 </div>  
  
 <div class="mt-4">  
 <label>Montant Mensuel</label>  
 <input class="form-control" type="number" step="any" name="montantMensuel" th:value="${abonnement.montantMensuel}">  
 <span class="text-danger" th:errors="${abonnement.montantMensuel}"></span>  
 </div>  
  
  
  
 <div class="offset-5 mt-4">  
 <button type="submit" class="btn btn-primary "> saveAbonnement</button>  
 </div>  
 </form>  
 </div>  
</div>  
</body>  
</html>

* **Form Abonnement**

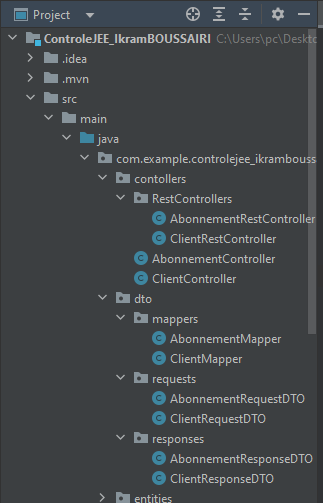
<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultrap.net.nz/thymeleaf/layout"  
 layout:decorate="navbarTemp">  
  
<head>  
 <meta charset="UTF-8">  
 <title>Form Abonnement</title>  
 <Link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">  
</head>  
<body>  
  
<div layout:fragment="content1">  
 <div class="col-md-4 offset-4 mt-4">  
 <form method="post" th:action="@{/saveAbonnement(clientId=${abonnement.getClient().getId()})}">  
  
 <div class="mt-4">  
 <label>Date Abonnement </label>  
 <input class="form-control" type="date" name="dateAbonnement" th:value="${abonnement.dateAbonnement}">  
 </div>  
  
 <div class="mt-4">  
 <label class="form-label">Type Abonnement</label>  
 <select name="typeAbonnement" class="form-select" aria-label="Default select example">  
 <option selected value="GSM">GSM</option>  
 <option th:value="INTERNET">INTERNET</option>  
 <option th:value="TELEPHONE\_FIXE">TELEPHONE\_FIXE</option>  
 </select>  
 <span class="text-danger form-text" th:errors="${abonnement.typeAbonnement}"></span>  
 </div>  
  
  
 <div class="mt-4">  
 <label>Type Abonnement : </label>  
 <label th:text="${abonnement.typeAbonnement}"> </label>  
 <input class="form-control" type="hidden" name="typeAbonnement" th:value="${abonnement.typeAbonnement}">  
 </div>  
  
  
 <div class="mt-4">  
 <label class="form-label">Solde</label>  
 <input class="form-control" type="number" step="any" name="soldeAbonnement" th:value="${abonnement.soldeAbonnement}">  
 <span class="text-danger form-text" th:errors="${abonnement.soldeAbonnement}"></span>  
 </div>  
  
 <div class="mt-4">  
 <label class="form-label">Montant Mensuel</label>  
 <input class="form-control" type="number" step="any" name="montantMensuel" th:value="${abonnement.montantMensuel}">  
 <span class="text-danger form-text" th:errors="${abonnement.montantMensuel}"></span>  
 </div>  
  
  
 <div class="offset-5 mt-4">  
 <button type="submit" class="btn btn-primary "> Submit</button>  
 </div>  
 </form>  
 </div>  
</div>  
</body>  
</html>



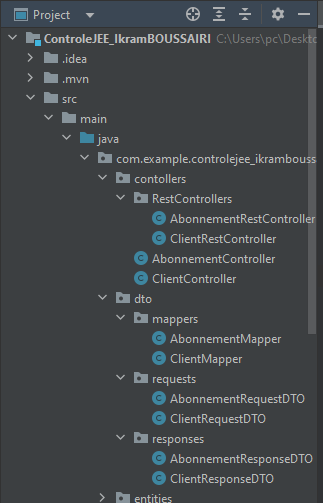


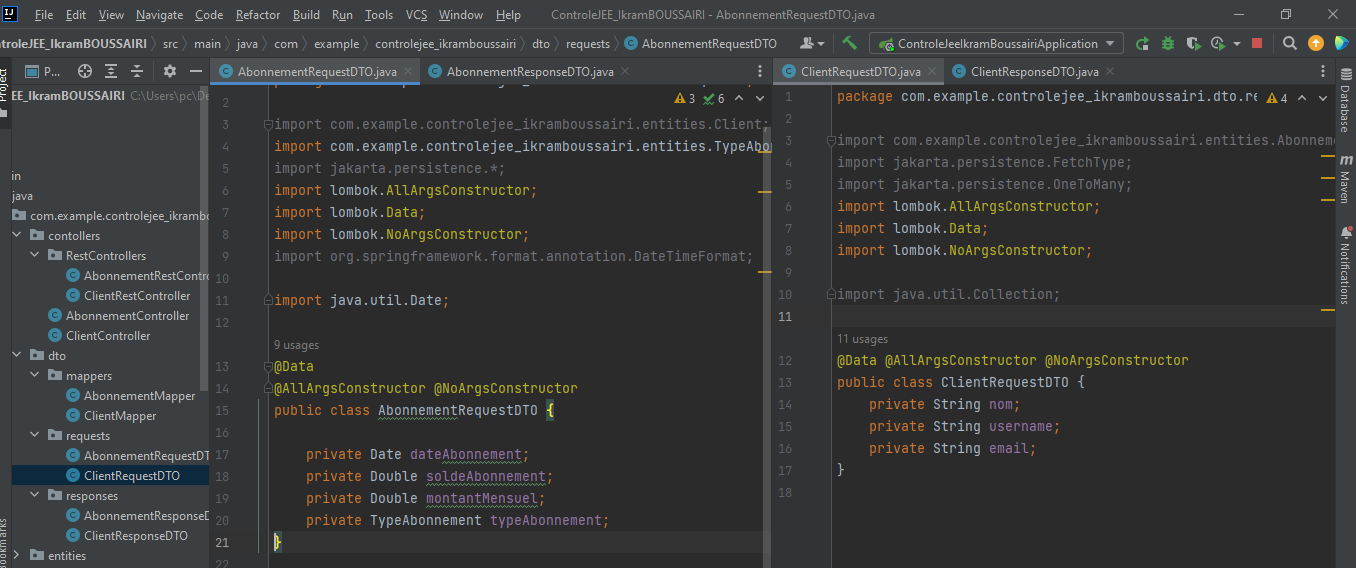


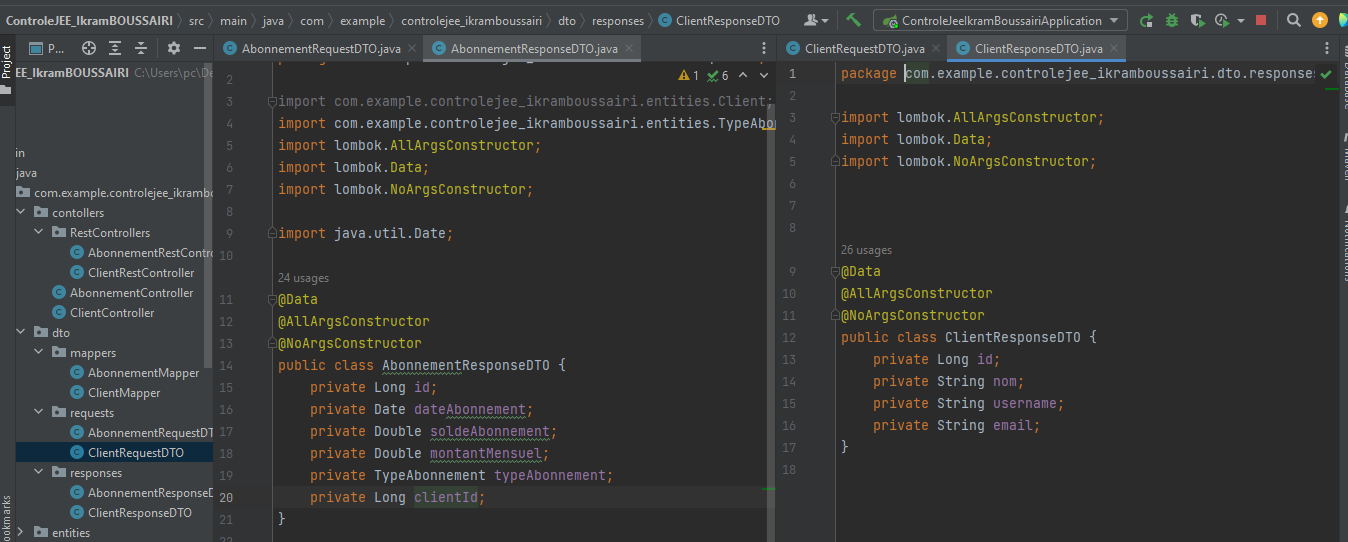
5. Créer un web service RESTful qui permet de gérer les clients et les abonnements



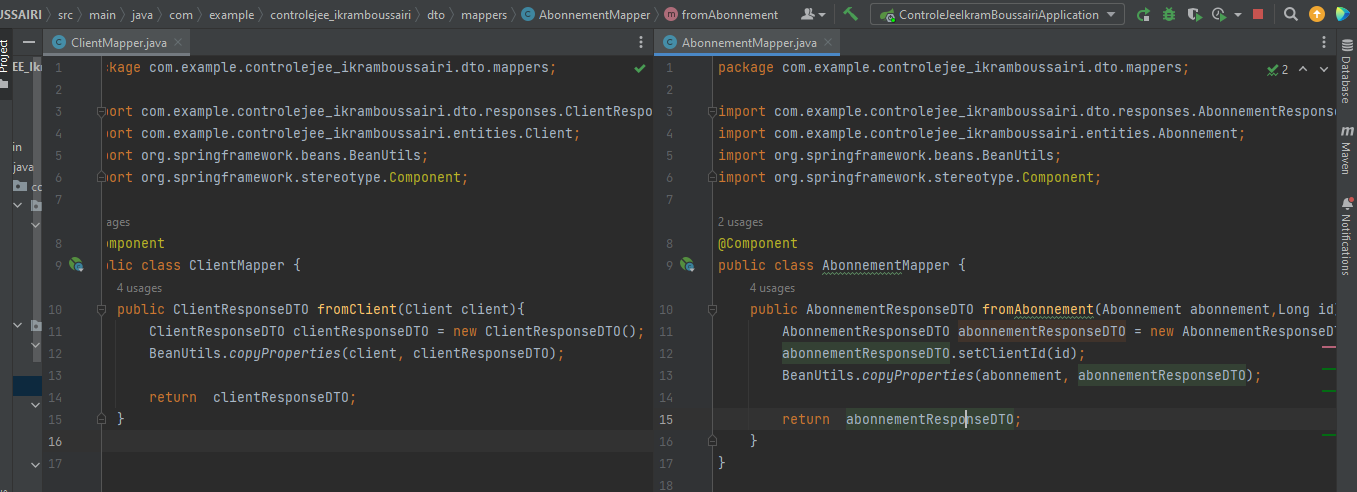
**- DTOs**



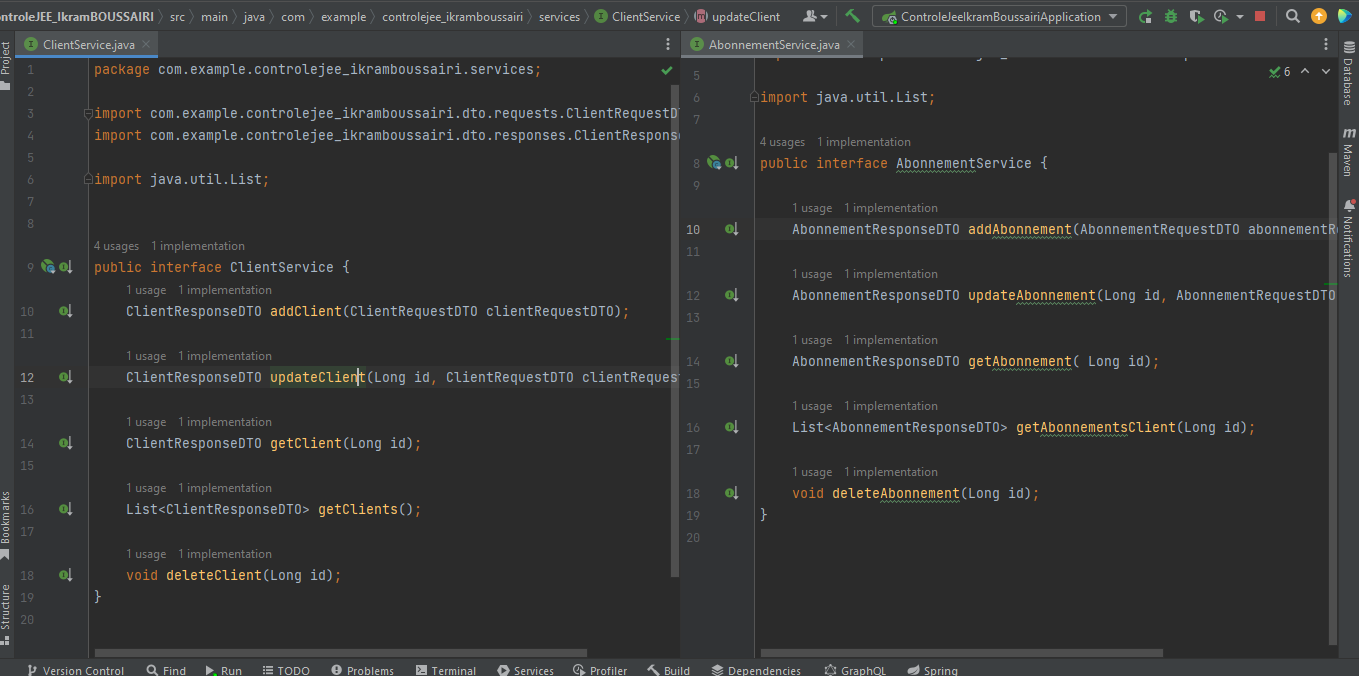




**- Mappers**



**- Interface Service**



**- Services Implémentation**

**Client**

package com.example.controlejee\_ikramboussairi.services.serviceImp;  
  
import com.example.controlejee\_ikramboussairi.dto.mappers.ClientMapper;  
import com.example.controlejee\_ikramboussairi.dto.requests.ClientRequestDTO;  
import com.example.controlejee\_ikramboussairi.dto.responses.ClientResponseDTO;  
import com.example.controlejee\_ikramboussairi.entities.Abonnement;  
import com.example.controlejee\_ikramboussairi.entities.Client;  
import com.example.controlejee\_ikramboussairi.repositories.AbonnementRepository;  
import com.example.controlejee\_ikramboussairi.repositories.ClientRepository;  
import com.example.controlejee\_ikramboussairi.services.ClientService;  
import lombok.AllArgsConstructor;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Service  
@Transactional @AllArgsConstructor  
public class ClientServiceImp implements ClientService {  
  
 private ClientRepository clientRepository;  
 private ClientMapper clientMapper;  
  
 private AbonnementRepository abonnementRepository;  
  
 @Override  
 public ClientResponseDTO addClient(ClientRequestDTO clientRequestDTO) {  
 Client client = new Client();  
 client.setNom(clientRequestDTO.getNom());  
 client.setUsername(clientRequestDTO.getUsername());  
 client.setEmail(clientRequestDTO.getEmail());  
  
 Client saveClinet = clientRepository.save(client);  
  
 return clientMapper.fromClient(saveClinet);  
 }  
  
 @Override  
 public ClientResponseDTO updateClient(Long id, ClientRequestDTO clientRequestDTO) {  
  
 Client client = clientRepository.findById(id).orElse(null);  
 if(client!=null){  
 if(clientRequestDTO.getNom()!=null) client.setNom(clientRequestDTO.getNom());  
 if(clientRequestDTO.getUsername()!=null) client.setUsername(clientRequestDTO.getUsername());  
 if(clientRequestDTO.getEmail()!=null) client.setEmail(clientRequestDTO.getEmail());  
 }  
  
 Client clientUpdated = clientRepository.save(client);  
  
 return clientMapper.fromClient(clientUpdated);  
 }  
  
 @Override  
 public ClientResponseDTO getClient(Long id) {  
 Client client = clientRepository.findById(id).orElse(null);  
  
 return clientMapper.fromClient(client);  
 }  
  
 @Override  
 public List<ClientResponseDTO> getClients() {  
 List<Client> clients = clientRepository.findAll();  
  
 List<ClientResponseDTO> responseDTOS = new ArrayList<>();  
  
 for(Client c : clients){  
 ClientResponseDTO clientResponseDTO = new ClientResponseDTO();  
 clientResponseDTO = clientMapper.fromClient(c);  
 responseDTOS.add(clientResponseDTO);  
 }  
 return responseDTOS;  
 }  
  
 @Override  
 public void deleteClient(Long id) {  
 List<Abonnement> abonnements = abonnementRepository.findAbonnementsByClient\_Id(id);  
 for (Abonnement a : abonnements ) {  
 abonnementRepository.delete(a);  
 }  
 clientRepository.deleteById(id);  
 }  
}

**Abonnement**

package com.example.controlejee\_ikramboussairi.services.serviceImp;  
  
import com.example.controlejee\_ikramboussairi.dto.mappers.AbonnementMapper;  
import com.example.controlejee\_ikramboussairi.dto.requests.AbonnementRequestDTO;  
import com.example.controlejee\_ikramboussairi.dto.requests.ClientRequestDTO;  
import com.example.controlejee\_ikramboussairi.dto.responses.AbonnementResponseDTO;  
import com.example.controlejee\_ikramboussairi.dto.responses.ClientResponseDTO;  
import com.example.controlejee\_ikramboussairi.entities.Abonnement;  
import com.example.controlejee\_ikramboussairi.entities.Client;  
import com.example.controlejee\_ikramboussairi.repositories.AbonnementRepository;  
import com.example.controlejee\_ikramboussairi.repositories.ClientRepository;  
import com.example.controlejee\_ikramboussairi.services.AbonnementService;  
import lombok.AllArgsConstructor;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Service  
@AllArgsConstructor @Transactional  
public class AbonnementServiceImp implements AbonnementService {  
  
 private AbonnementRepository abonnementRepository;  
 private AbonnementMapper abonnementMapper;  
 private ClientRepository clientRepository;  
  
 @Override  
 public AbonnementResponseDTO addAbonnement(AbonnementRequestDTO abonnementRequestDTO, Long id) {  
 Abonnement abonnement = new Abonnement();  
 abonnement.setDateAbonnement(abonnementRequestDTO.getDateAbonnement());  
 abonnement.setTypeAbonnement(abonnementRequestDTO.getTypeAbonnement());  
 abonnement.setSoldeAbonnement(abonnementRequestDTO.getSoldeAbonnement());  
 abonnement.setMontantMensuel(abonnementRequestDTO.getMontantMensuel());  
 abonnement.setClient(clientRepository.findById(id).orElse(null));  
  
 Abonnement abonnementsaved = abonnementRepository.save(abonnement);  
  
 return abonnementMapper.fromAbonnement(abonnementsaved,id);  
 }  
  
 @Override  
 public AbonnementResponseDTO updateAbonnement(Long id, AbonnementRequestDTO abonnementRequestDTO) {  
 Abonnement abonnement = abonnementRepository.findById(id).orElse(null);  
 if(abonnement!=null){  
 if(abonnementRequestDTO.getDateAbonnement()!=null) abonnement.setDateAbonnement(abonnementRequestDTO.getDateAbonnement());  
 if(abonnementRequestDTO.getTypeAbonnement()!=null) abonnement.setTypeAbonnement(abonnementRequestDTO.getTypeAbonnement());  
 if(abonnementRequestDTO.getSoldeAbonnement()!=null) abonnement.setSoldeAbonnement(abonnementRequestDTO.getSoldeAbonnement());  
 if(abonnementRequestDTO.getMontantMensuel()!=null) abonnement.setMontantMensuel(abonnementRequestDTO.getMontantMensuel());  
 }  
  
 Abonnement abonnementUpdated = abonnementRepository.save(abonnement);  
 return abonnementMapper.fromAbonnement(abonnementUpdated, abonnement.getClient().getId());  
 }  
  
 @Override  
 public AbonnementResponseDTO getAbonnement(Long id) {  
 Abonnement abonnement = abonnementRepository.findById(id).orElse(null);  
  
 return abonnementMapper.fromAbonnement(abonnement,abonnement.getClient().getId());  
 }  
  
 @Override  
 public List<AbonnementResponseDTO> getAbonnementsClient(Long idClient) {  
 List<Abonnement> abonnements = abonnementRepository.findAbonnementsByClient\_Id(idClient);  
 List<AbonnementResponseDTO> ResponseDTOS = new ArrayList<>();  
  
 for (Abonnement a:abonnements){  
 AbonnementResponseDTO abonnementResponseDTO = new AbonnementResponseDTO();  
 abonnementResponseDTO = abonnementMapper.fromAbonnement(a, a.getClient().getId());  
 ResponseDTOS.add(abonnementResponseDTO);  
 }  
  
  
 return ResponseDTOS;  
 }  
  
 @Override  
 public void deleteAbonnement(Long id) {  
 abonnementRepository.deleteById(id);  
 }  
}

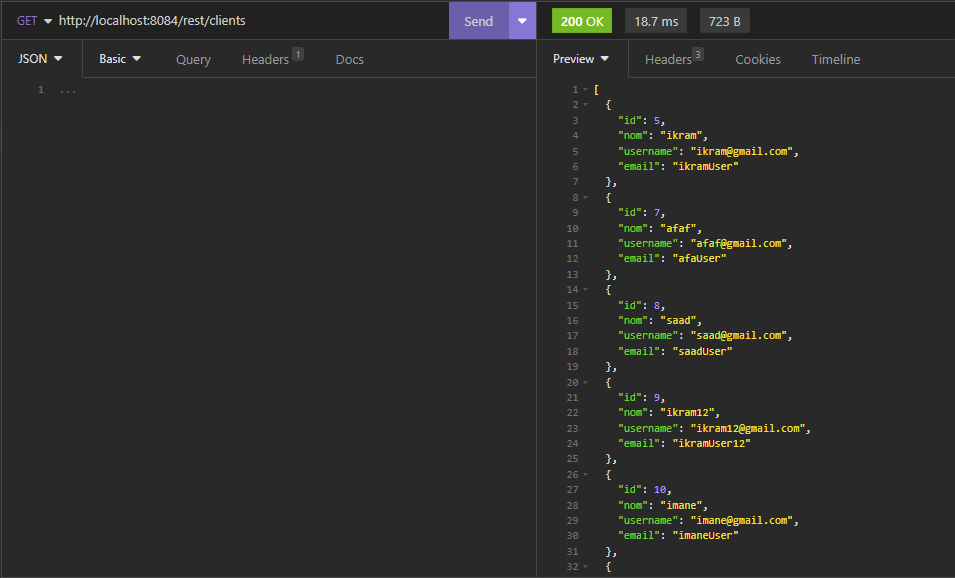
**- Rest Controllers**

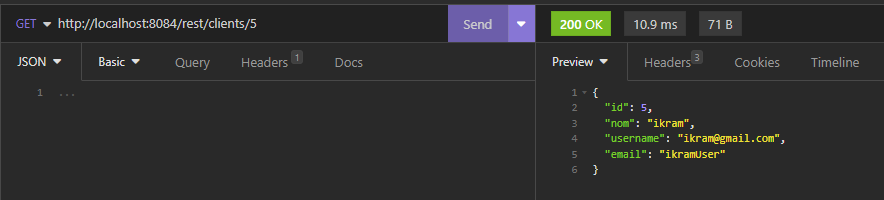
**Client**

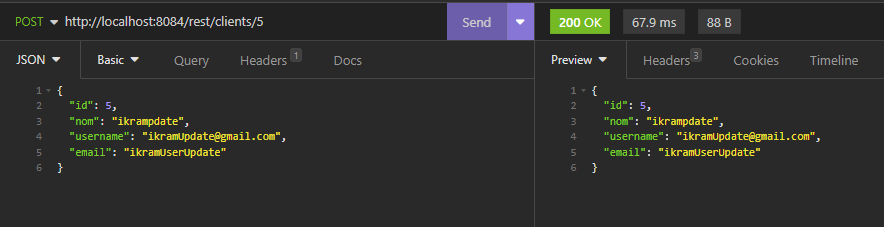
package com.example.controlejee\_ikramboussairi.contollers.RestControllers;  
  
import com.example.controlejee\_ikramboussairi.dto.requests.ClientRequestDTO;  
import com.example.controlejee\_ikramboussairi.dto.responses.ClientResponseDTO;  
  
import com.example.controlejee\_ikramboussairi.services.ClientService;  
import lombok.AllArgsConstructor;  
  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/rest")  
@AllArgsConstructor  
public class ClientRestController {  
 private ClientService clientService;  
  
 @PostMapping("/clients")  
 public ClientResponseDTO saveClient(@RequestBody ClientRequestDTO clientRequestDTO){  
  
 return clientService.addClient(clientRequestDTO);  
  
 }  
  
 @GetMapping("/clients")  
 public List<ClientResponseDTO> getclients(){  
 List<ClientResponseDTO> clients = clientService.getClients();  
 return clients;  
 }  
  
 @GetMapping("/clients/{id}")  
 public ClientResponseDTO getClient(@PathVariable Long id){  
 ClientResponseDTO client = clientService.getClient(id);  
 return client;  
 }  
  
 @PostMapping("/clients/{id}")  
 public ClientResponseDTO updateClient(@PathVariable Long id,@RequestBody ClientRequestDTO clientRequestDTO){  
  
 return clientService.updateClient(id,clientRequestDTO) ;  
 }  
  
 @DeleteMapping("/clients/{id}")  
 public void deleteClient(@PathVariable Long id){  
 clientService.deleteClient(id);  
 }  
  
}

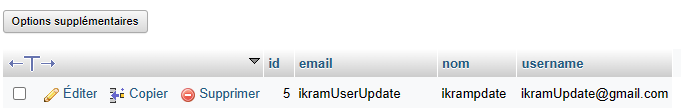
**Abonnement**

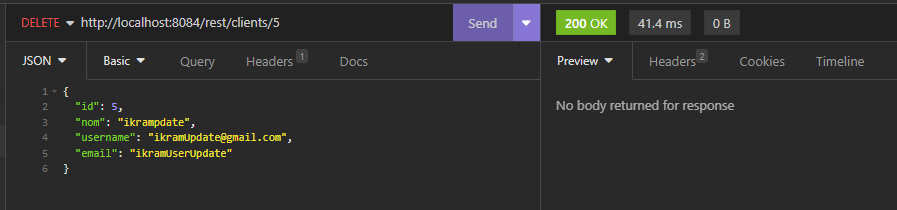
package com.example.controlejee\_ikramboussairi.contollers.RestControllers;  
  
  
import com.example.controlejee\_ikramboussairi.dto.requests.AbonnementRequestDTO;  
import com.example.controlejee\_ikramboussairi.dto.responses.AbonnementResponseDTO;  
import com.example.controlejee\_ikramboussairi.services.AbonnementService;  
import lombok.AllArgsConstructor;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/rest")  
@AllArgsConstructor  
public class AbonnementRestController {  
  
 private AbonnementService abonnementService;  
  
 @PostMapping("/abonnements/{id}")  
 public AbonnementResponseDTO saveAbonnement(@RequestBody AbonnementRequestDTO abonnementRequestDTO, @PathVariable Long id){  
  
 return abonnementService.addAbonnement(abonnementRequestDTO, id);  
  
 }  
  
 @GetMapping("/abonnementsClient/{idClient}")  
 public List<AbonnementResponseDTO> getAbonnementsClient(@PathVariable Long idClient){  
 List<AbonnementResponseDTO> abonnements = abonnementService.getAbonnementsClient(idClient);  
 return abonnements;  
 }  
  
 @GetMapping("/abonnement/{id}")  
 public AbonnementResponseDTO getAbonnement(@PathVariable Long id){  
 AbonnementResponseDTO abonnement = abonnementService.getAbonnement(id);  
 return abonnement;  
 }  
  
 @PostMapping("/abonnements/update/{id}")  
 public AbonnementResponseDTO updateAbonnement(@PathVariable Long id,@RequestBody AbonnementRequestDTO abonnementRequestDTO){  
  
 return abonnementService.updateAbonnement(id,abonnementRequestDTO) ;  
 }  
  
 @DeleteMapping("/abonnements/{id}")  
 public void deleteAbonnement(@PathVariable Long id){  
 abonnementService.deleteAbonnement(id);  
 }  
  
}

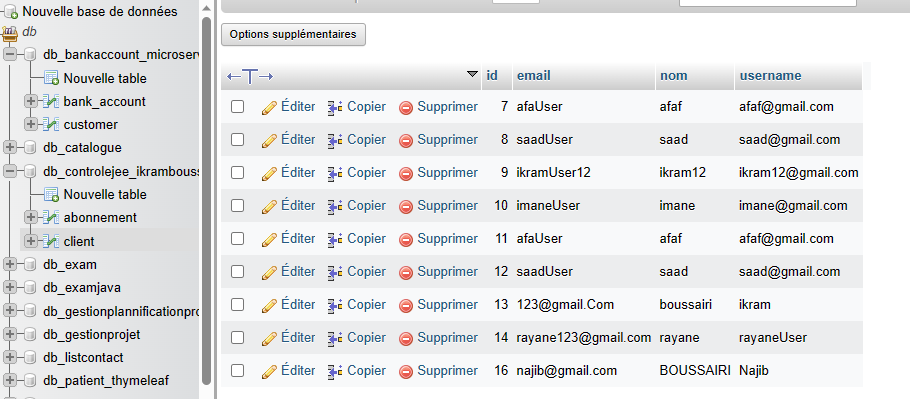


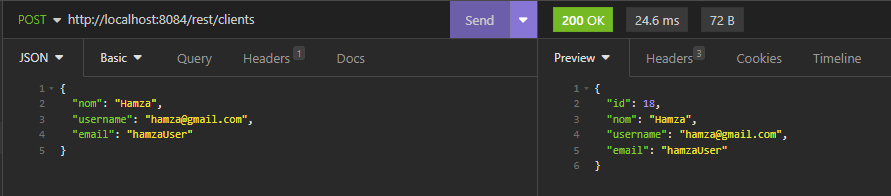


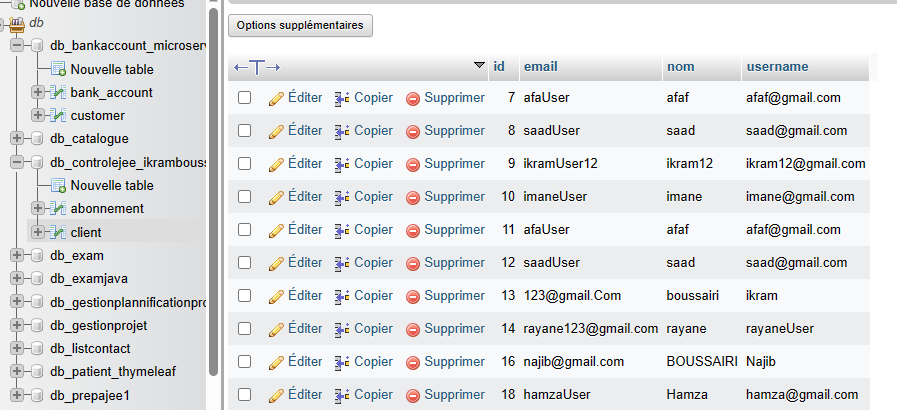


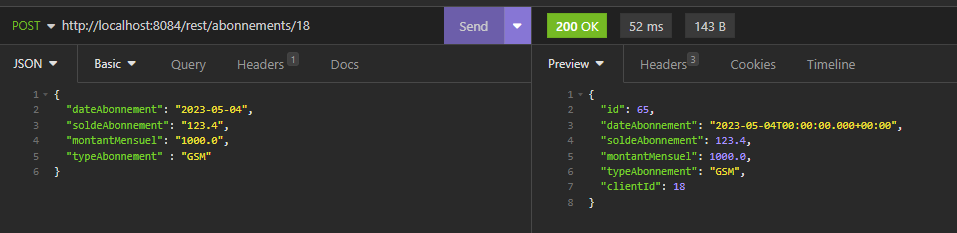


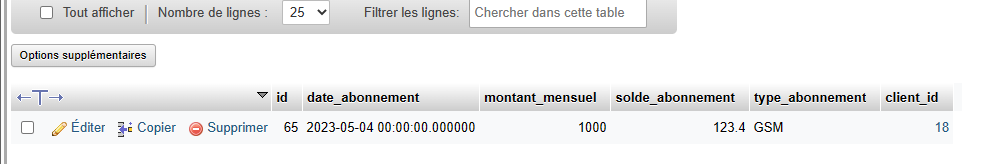


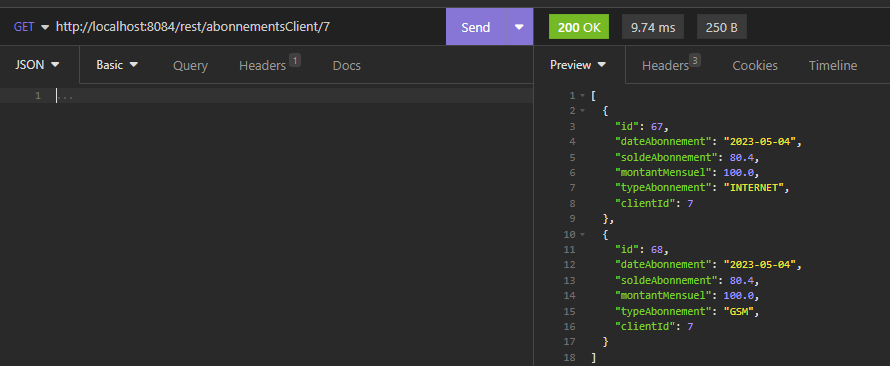


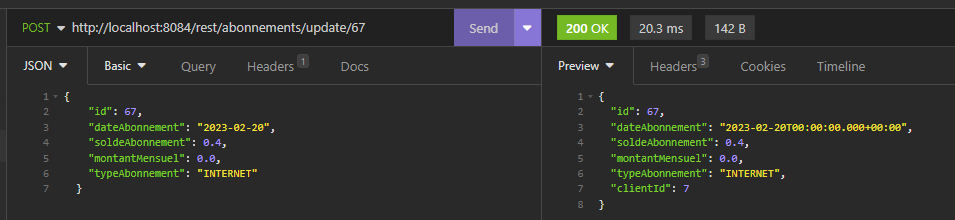


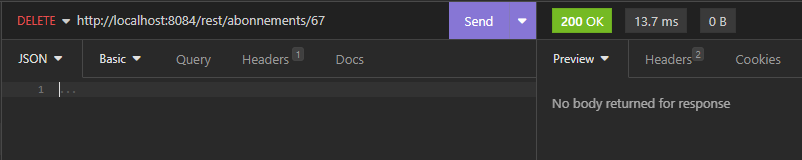


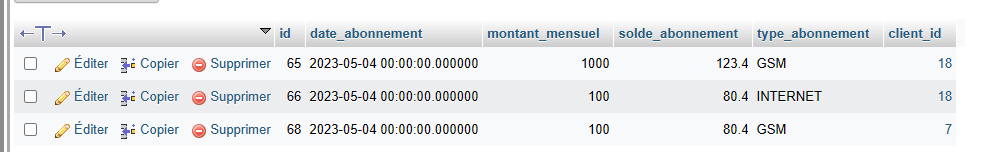












6. Sécurité :

- Security Config

package com.example.controlejee\_ikramboussairi.security;  
  
import lombok.AllArgsConstructor;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.provisioning.InMemoryUserDetailsManager;  
import org.springframework.security.web.SecurityFilterChain;  
  
@Configuration  
@EnableWebSecurity  
@AllArgsConstructor  
public class SecurityConfig {  
  
 private PasswordEncoder passwordEncoder;  
  
  
 @Bean  
 public InMemoryUserDetailsManager inMemoryUserDetailsManager(){  
 return new InMemoryUserDetailsManager(  
 User.*withUsername*("ikram").password(passwordEncoder.encode("ikram")).roles("USER").build(),  
 User.*withUsername*("rayane").password(passwordEncoder.encode("rayane")).roles("USER").build(),  
 User.*withUsername*("afaf").password(passwordEncoder.encode("afaf")).roles("USER","ADMIN").build()  
 );  
 }  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {  
 //httpSecurity.formLogin();  
 httpSecurity.authorizeHttpRequests().requestMatchers("/webjars/\*\*").permitAll();  
 httpSecurity.formLogin().loginPage("/login").permitAll();  
 httpSecurity.authorizeHttpRequests().requestMatchers("/user/\*\*").hasRole("USER");  
 httpSecurity.authorizeHttpRequests().requestMatchers("/admin/\*\*").hasRole("ADMIN");  
 httpSecurity.authorizeHttpRequests().anyRequest().authenticated();  
  
 httpSecurity.exceptionHandling().accessDeniedPage("/notAuthorized");  
  
 return httpSecurity.build();  
 }  
  
}

- Security Controller

package com.example.controlejee\_ikramboussairi.security;  
  
import org.springframework.stereotype.Component;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
  
@Controller  
public class SecurityController {  
 @GetMapping("/notAuthorized")  
 public String notAuthorized(){  
 return "notAuthorized";  
 }  
  
 @GetMapping("/login")  
 public String login(){  
 return "login";  
 }  
  
}

**Templates**

* Login

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
  
<head>  
 <meta charset="UTF-8">  
 <title>Form Ressource</title>  
 <Link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">  
</head>  
<body>  
  
<div class=" mt-4">  
 <div class="col-md-4 offset-4 ">  
 <div class="card">  
 <div class="card-header">Authentication</div>  
 <div class="card-body">  
 <form method="post" th:action="@{/login}">  
 <div class="mt-4">  
 <label for="username" class="form-label">UserName</label>  
 <input id="username" class="form-control" type="text" name="username" placeholder="username">  
 </div>  
  
 <div class="mt-4">  
 <label for="password" class="form-label">Password</label>  
 <input id="password" class="form-control" type="password" name="password" >  
 </div>  
  
 <div class="mt-4">  
 <button type="submit" class="btn btn-primary "> Login</button>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
</div>  
  
</body>  
</html>

* Not Authaurized

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultrap.net.nz/thymeleaf/layout"  
 layout:decorate="navbarTemp">  
  
<head>  
 <meta charset="UTF-8">  
 <title>not Authorized</title>  
 <Link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">  
</head>  
<body>  
  
<div layout:fragment="content1">  
 <div class=" alert alert-danger mt-3">  
 <h2>Not Authorized</h2>  
 </div>  
</div>  
</body>  
</html>

