



ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT
TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Département Mathématiques et Informatique

Rapport du Contrôle

Filière :

« Ingénierie Informatique : Big Data et Cloud Computing »

II-BDCC

Examen JEE : Gestion des Crédits

Réalisé par :

Yassine HACHGUER

Année Universitaire : 2024-2025

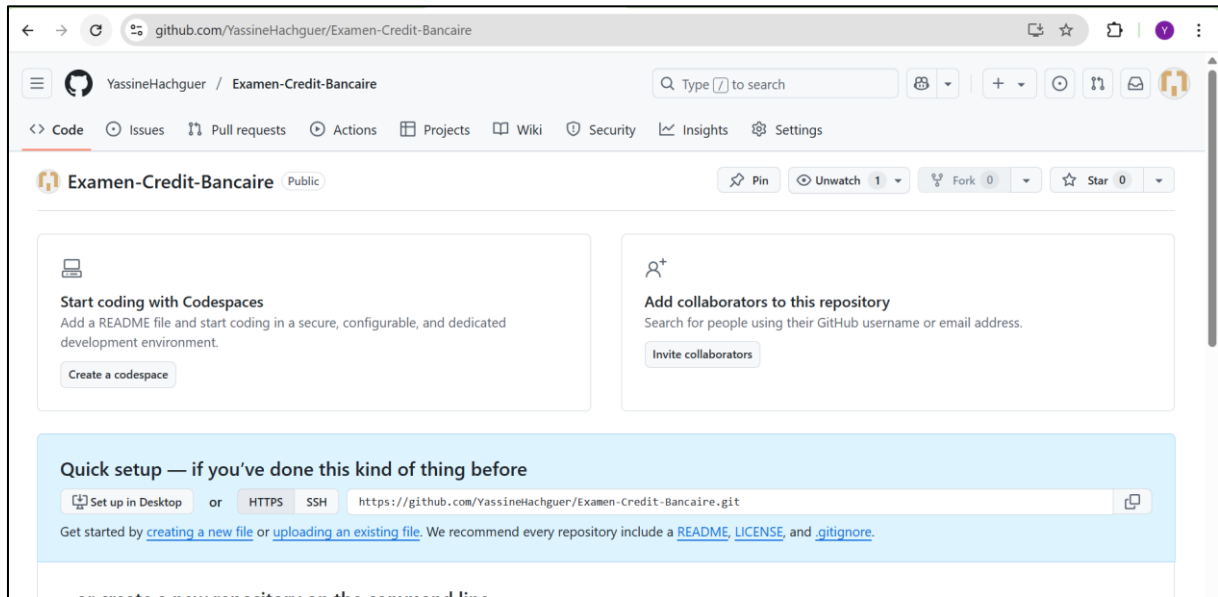
ENSET, Avenue Hassan II - B.P. 159 - Mohammedia - Maroc
05 23 32 22 20 / 05 23 32 35 30 – Fax : 05 23 32 25 46 Site Web: www.enset-media.ac.ma
E-Mail : enset-media@enset-media.ac.ma

Table de matière

Livrables.....	3
1. Créer un repository Github qui porte le nom suivant : VotreNom-VotrePrenom-Exam-JEE 3	
Conception	4
1. Établir une architecture technique du projet.....	4
2. Établir un diagramme de classes qui montre les entités. On ne représentera que les attributs.....	5
Implémentation.....	5
1. Créer un projet Spring boot avec les dépendances requises. Les identifiants du projet GroupId, ArtifactId et le package de base doivent contenir votre nom et prénom	5
2. Couche DAO	6
a. Créer les entités JPA	6
b. Créer les interfaces JPA Repository basées sur Spring Data	6
c. Tester la couche DAO avec une application qui alimente la base de données avec quelques enregistrements de test.	7
3. Créer une couche service en créant les DTOs et les Mappers, en proposant les fonctionnalités que vous voyez importantes	8
4. Créer les Web services (Rest Controllers) en proposant les fonctionnalités que vous estimez importantes. Tester les REST API en générant la documentations SWAGGER (Open API Doc)	10
5. Proposer une application frontend en utilisant Angular Framework	13

Livrables

1. Créer un repository Github qui porte le nom suivant : **VotreNom-VotrePrenom-Exam-JEE**



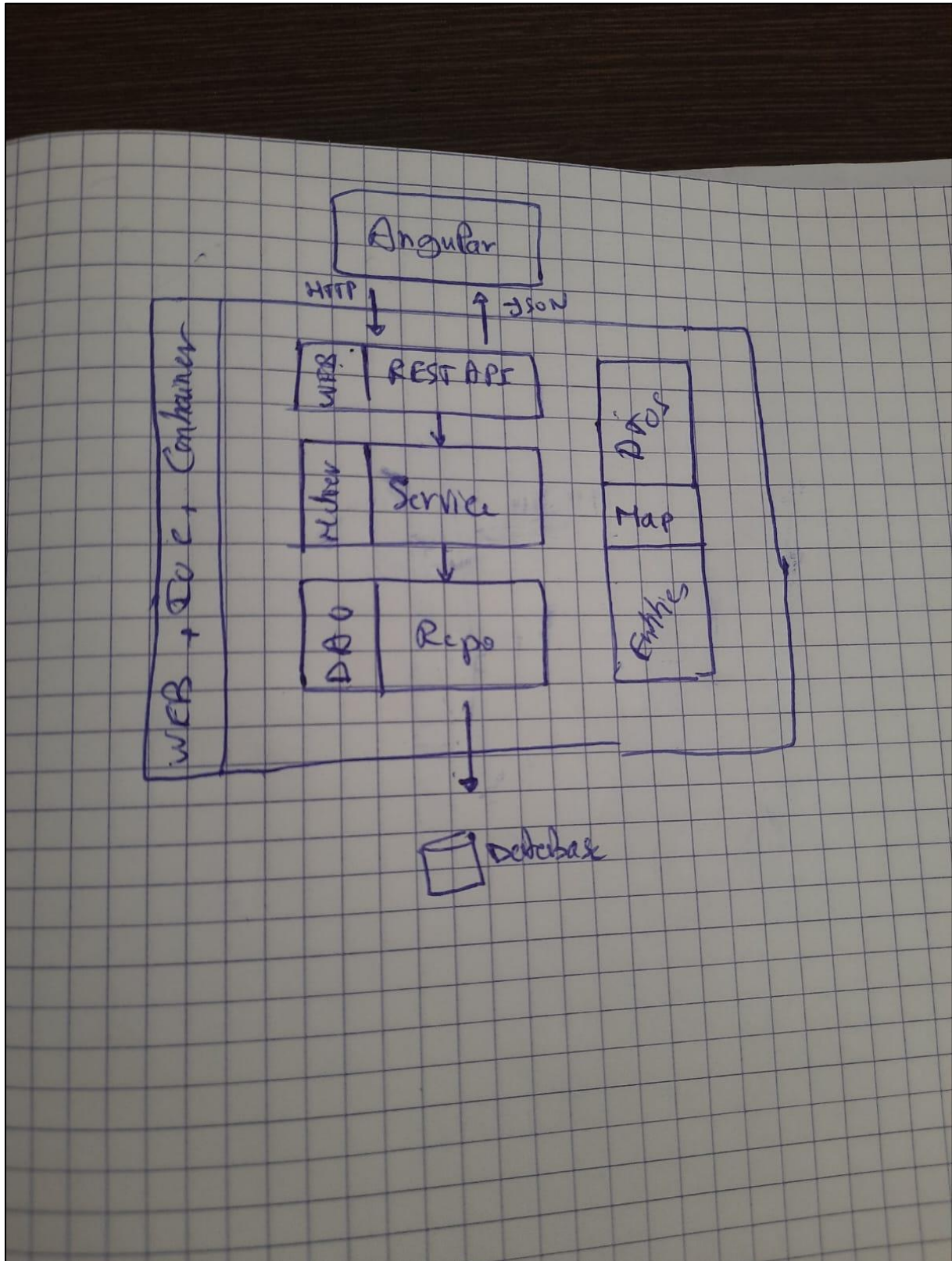
J'ai 2 branch :

La branch Main Contient le Backend du projet

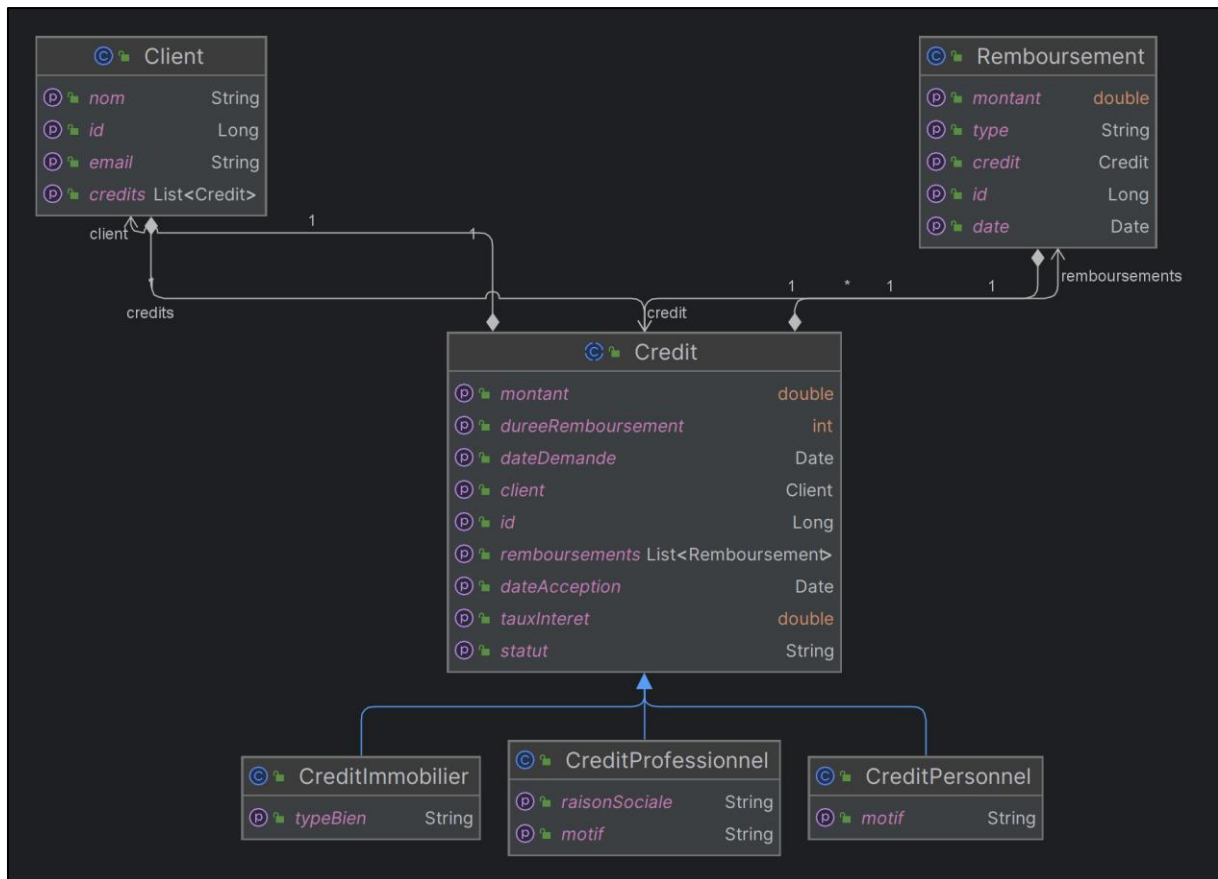
La branch Master Contient le Frontend du projet

Conception

1. Établir une architecture technique du projet

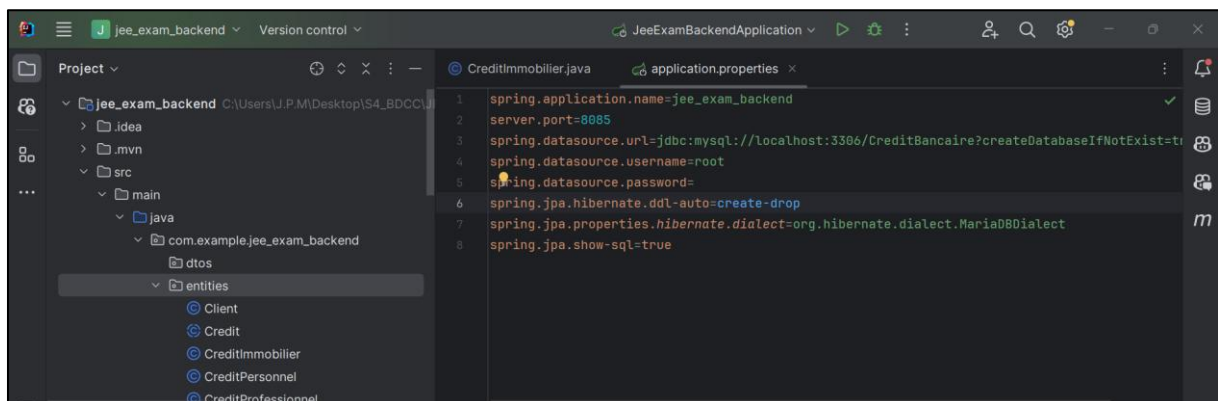


2. Établir un diagramme de classes qui montre les entités. On ne représentera que les attributs



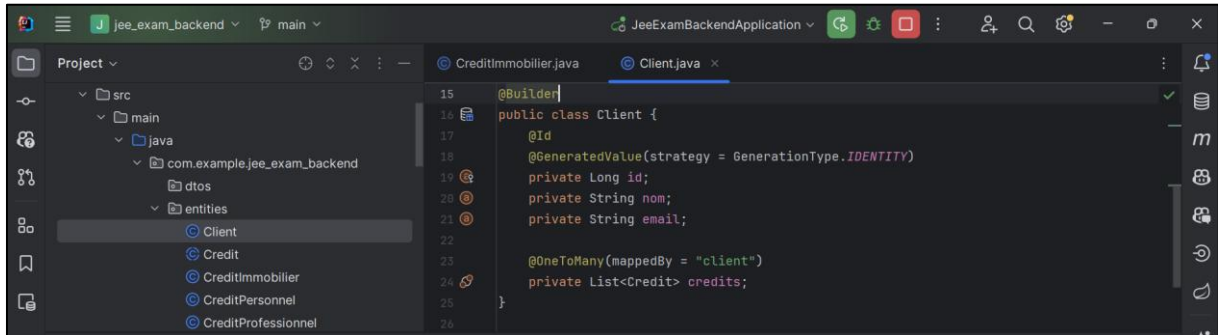
Implémentation

1. Créer un projet Spring boot avec les dépendances requises. Les identifiants du projet `GroupId`, `ArtifactId` et le package de base doivent contenir votre nom et prénom



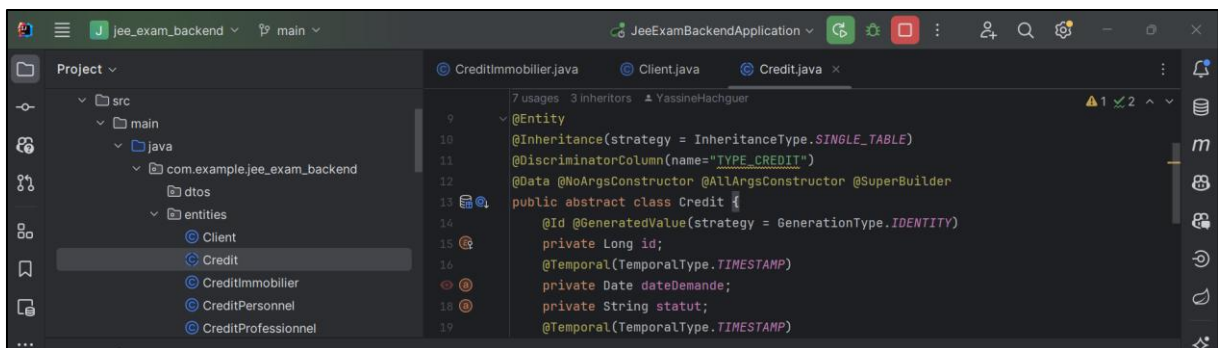
2. Couche DAO

a. Créer les entités JPA



This screenshot shows the IntelliJ IDEA interface with the 'Client.java' file open. The project structure on the left includes 'src/main/java/com/example/jee_exam_backend/entities'. The code for 'Client' is as follows:

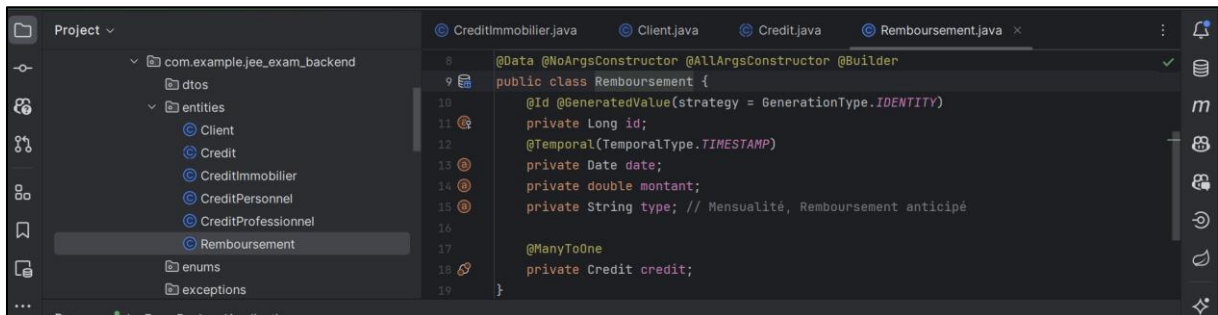
```
15 @Builder
16 public class Client {
17     @Id
18     @GeneratedValue(strategy = GenerationType.IDENTITY)
19     private Long id;
20     private String nom;
21     private String email;
22
23     @OneToMany(mappedBy = "client")
24     private List<Credit> credits;
25 }
26
```



This screenshot shows the IntelliJ IDEA interface with the 'Credit.java' file open. The code for 'Credit' is as follows:

```
9 @Entity
10 @Inheritance(strategy = InheritanceType.SINGLE_TABLE)
11 @DiscriminatorColumn(name="TYPE_CREDIT")
12 @Data @NoArgsConstructor @AllArgsConstructor @SuperBuilder
13 public abstract class Credit {
14     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16     @Temporal(TemporalType.TIMESTAMP)
17     private Date dateDemande;
18     private String statut;
19     @Temporal(TemporalType.TIMESTAMP)

```

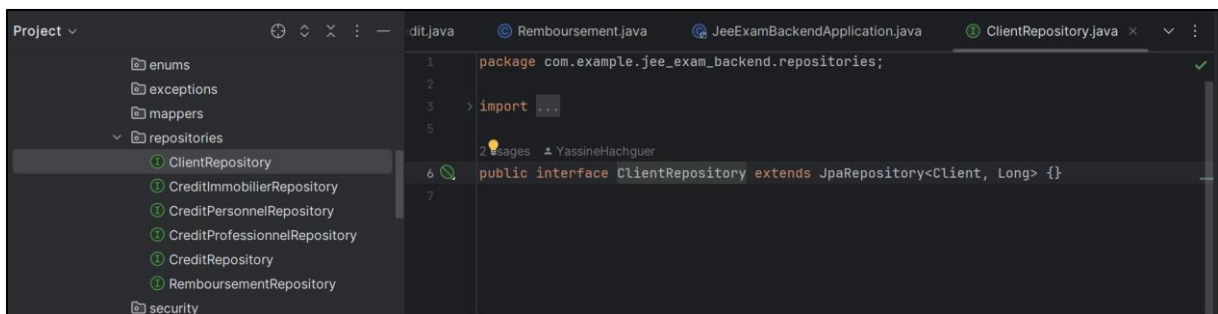


This screenshot shows the IntelliJ IDEA interface with the 'Remboursement.java' file open. The code for 'Remboursement' is as follows:

```
8 @Data @NoArgsConstructor @AllArgsConstructor @Builder
9 public class Remboursement {
10     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
11     private Long id;
12     @Temporal(TemporalType.TIMESTAMP)
13     private Date date;
14     private double montant;
15     private String type; // Mensualité, Remboursement anticipé
16
17     @ManyToOne
18     private Credit credit;
19 }

```

b. Créer les interfaces JPA Repository basées sur Spring Data



This screenshot shows the IntelliJ IDEA interface with the 'ClientRepository.java' file open. The code for 'ClientRepository' is as follows:

```
1 package com.example.jee_exam_backend.repositories;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6 public interface ClientRepository extends JpaRepository<Client, Long> {}
7
```

```

1 package com.example.jee_exam_backend.repositories;
2
3 import com.example.jee_exam_backend.entities.Credit;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface CreditRepository extends JpaRepository<Credit, Long> {}
7

```

```

1 package com.example.jee_exam_backend.repositories;
2
3 import com.example.jee_exam_backend.entities.Remboursement;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface RemboursementRepository extends JpaRepository<Remboursement, Long> {}
7

```

c. Tester la couche DAO avec une application qui alimente la base de données avec quelques enregistrements de test.

```

11 package com.example.jee_exam_backend;
12
13 import org.springframework.boot.SpringApplication;
14
15 public class JeeExamBackendApplication implements CommandLineRunner {
16
17     private final ClientRepository clientRepository;
18     private final CreditRepository creditRepository;
19     private final CreditPersonnelRepository creditPersonnelRepository;
20     private final CreditImmobilierRepository creditImmobilierRepository;
21     private final CreditProfessionnelRepository creditProfessionnelRepository;
22
23 }
24

```

localhost:8081/h2-console/login.do?sessionId=939c5c39e55b0a607a7c24eafdafdb3f

Auto commit: ☒ Max rows: 1000 Auto complete: Off Auto select: On

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM CLIENT;

ID	EMAIL	NOM
1	yassine@mail.com	yassine
2	yahya@mail.com	yahya

(2 rows, 3 ms)

localhost:8081/h2-console/login.do?sessionId=939c5c39e55b0a607a7c24eafdafdb3f

Auto commit: ☒ Max rows: 1000 Auto complete: Off Auto select: On

jdbc:h2:mem:CreditBank

- CLIENT
- CREDIT
- REMBOURSEMENT
- INFORMATION_SCHEMA
- Users
- H2 2.3.232 (2024-08-11)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM CREDIT

SELECT * FROM CREDIT;

DUREE_REMBOURSEMENT	MONTANT	TAUX_INTERET	CLIENT_ID	DATE_ACCESSION	DATE_DEMANDE	ID	TYPE_CREDIT	MOTIF	RAISON_SOCIALE
24	50000.0	5.5	1	null	2025-05-19 09:47:51.022	1	PERSONNEL	Achat Voiture	null
120	300000.0	3.2	2	2025-05-19 09:47:51.027	2025-05-19 09:47:51.027	2	IMMOBILIER	null	null
36	150000.0	6.0	2	null	2025-05-19 09:47:51.029	3	PROFESSIONNEL	Investissement matériel	SARL TechPlus

(3 rows, 1 ms)

Edit

Auto commit: ☒ Max rows: 1000 Auto complete: Off Auto select: On

jdbc:h2:mem:CreditBank

- CLIENT
- CREDIT
- REMBOURSEMENT
- INFORMATION_SCHEMA
- Users
- H2 2.3.232 (2024-08-11)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM REMBOURSEMENT

SELECT * FROM REMBOURSEMENT;

MONTANT	CREDIT_ID	DATE	ID	TYPE
2200.0	1	2025-05-19 09:47:51.031	1	Mensualité
3000.0	2	2025-05-19 09:47:51.032	2	Remboursement anticipé

(2 rows, 0 ms)

Edit

3. Créer une couche service en créant les DTOs et les Mappers, en proposant les fonctionnalités que vous voyez importantes

Project

- com.example.jee_exam_backend
 - dtos
 - ClientDTO
 - CreditDTO
 - CreditImmobilierDTO
 - CreditPersonnelDTO
 - CreditProfessionnelDTO
 - RemboursementDTO
 - entities
 - enums
 - exceptions
 - mappers
 - ClientMapper
 - CreditMapper

RemboursementDTO.java

```
import lombok.Data;
```

ClientDTO.java

```
import lombok.Data;
```

CreditDTO.java

```
import lombok.Data;
```

```
@Data
```

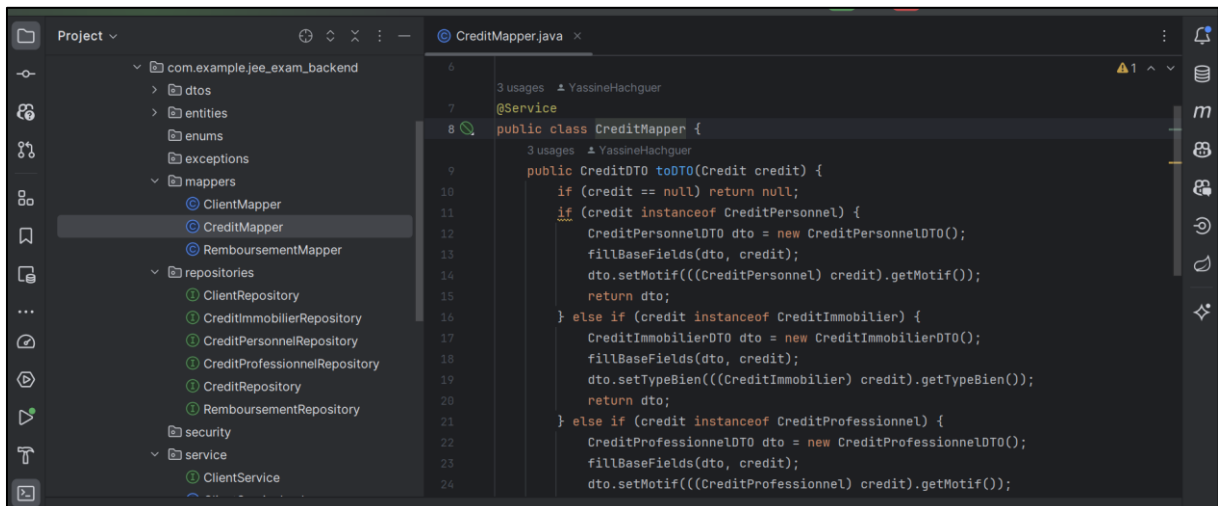
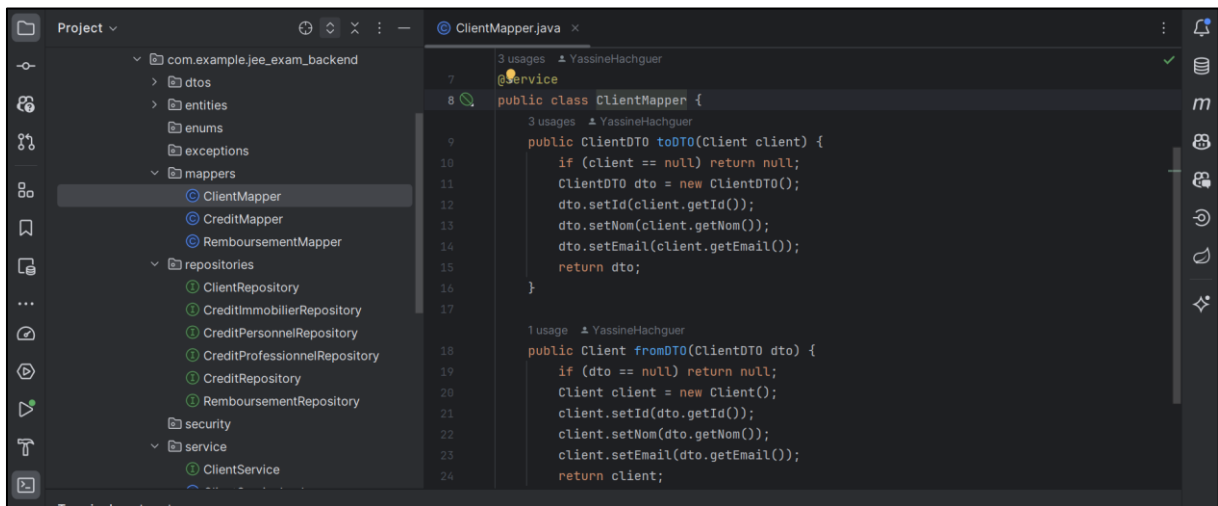
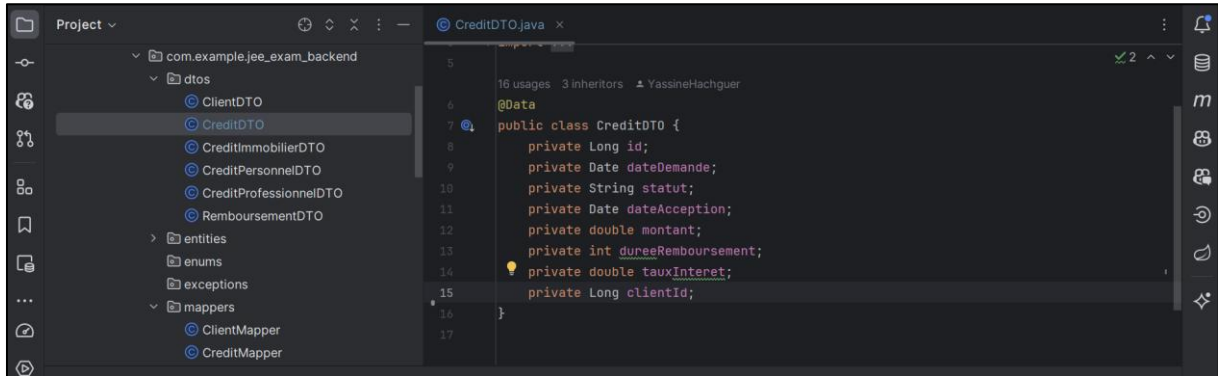
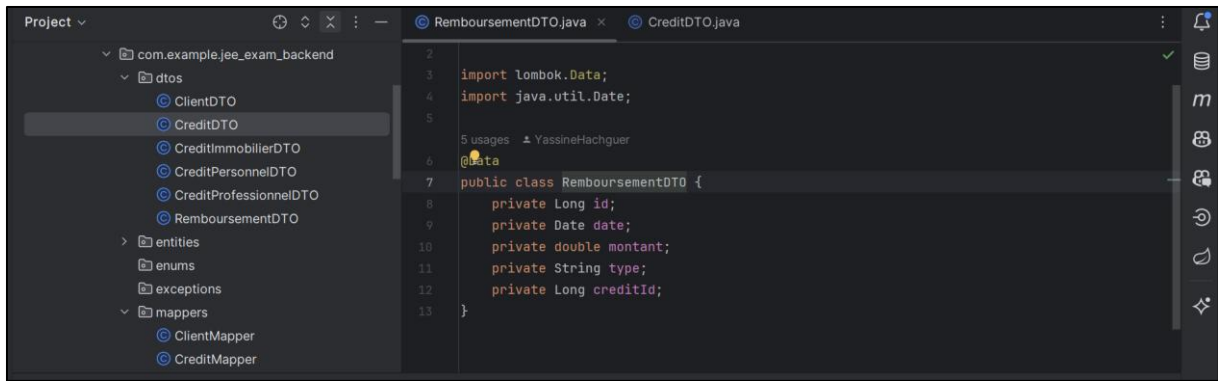
```
public class ClientDTO {
```

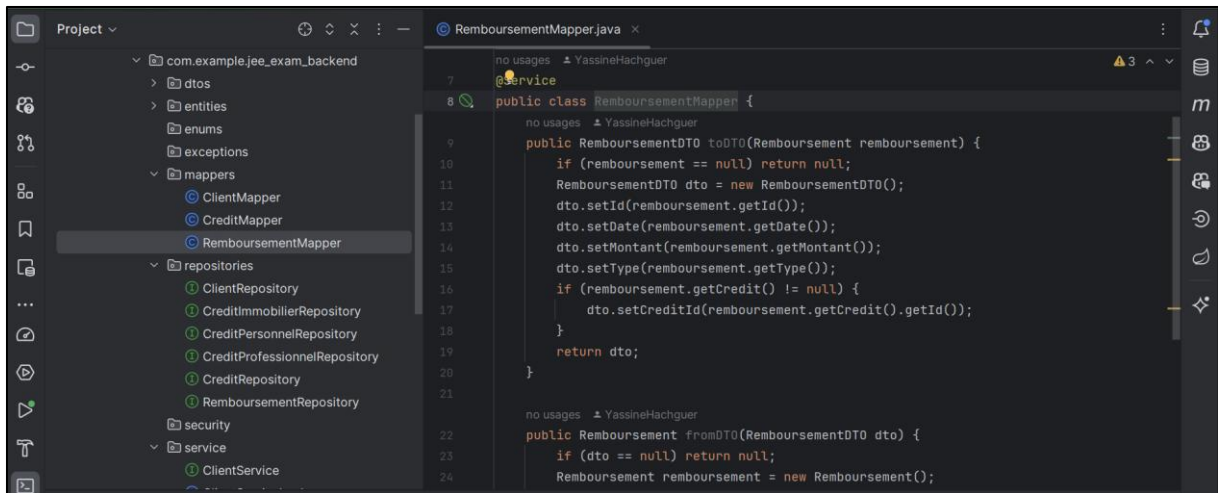
```
    private Long id;
```

```
    private String nom;
```

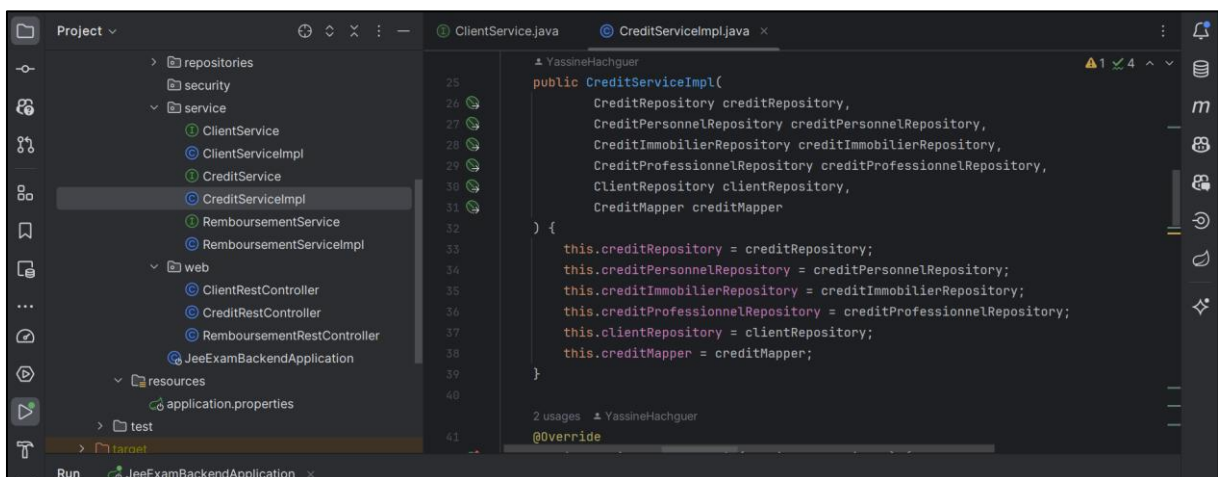
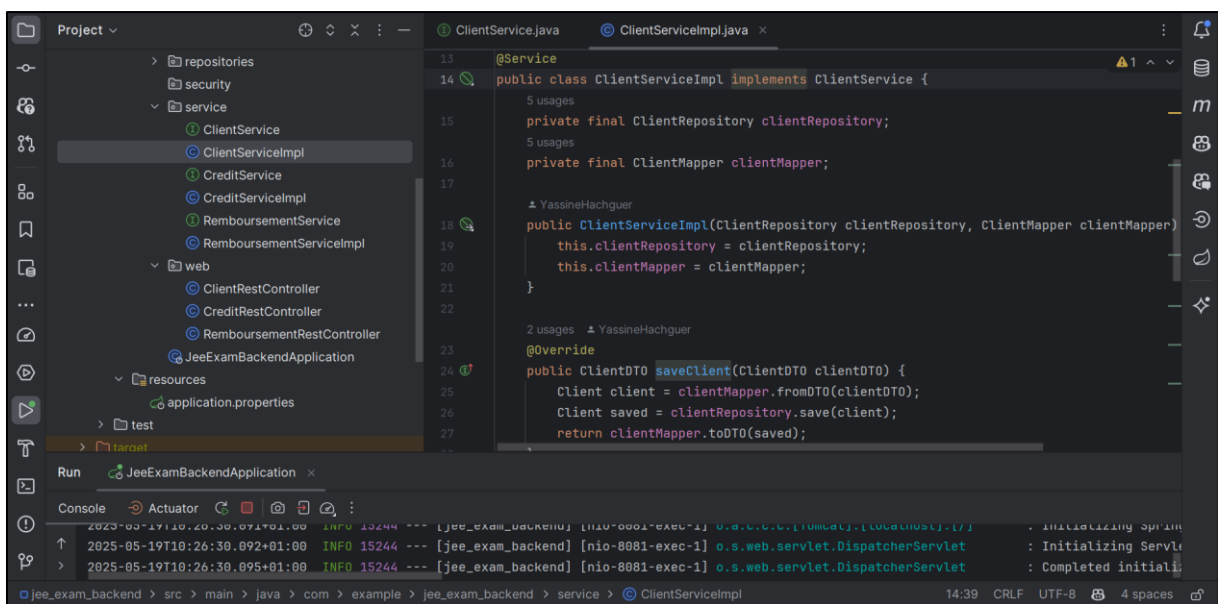
```
    private String email;
```

```
}
```

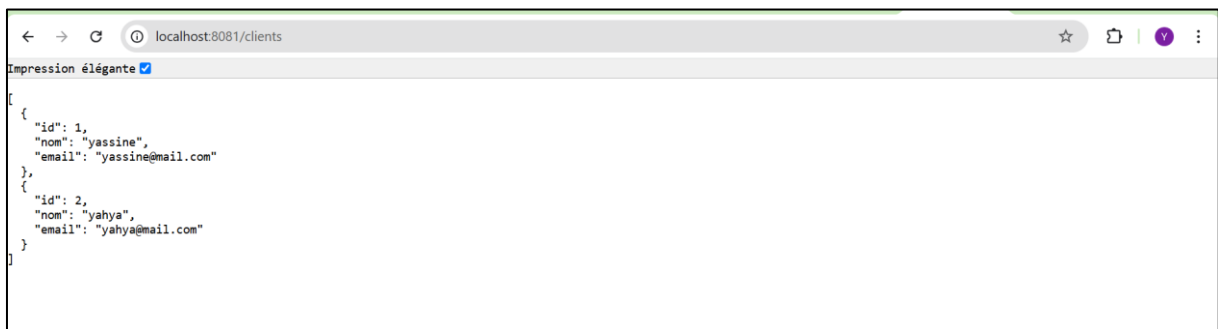


4. Créer les Web services (Rest Controllers) en proposant les fonctionnalités que vous estimez importantes. Tester les REST API en générant la documentations SWAGGER (Open API Doc)



```
17 private final RemboursementRepository remboursementRepository;
18 private final CreditRepository creditRepository;
19 private final RemboursementMapper remboursementMapper;
20
21 // YassineHachguer
22 public RemboursementServiceImpl(
23     RemboursementRepository remboursementRepository,
24     CreditRepository creditRepository,
25     RemboursementMapper remboursementMapper
26 ) {
27     this.remboursementRepository = remboursementRepository;
28     this.creditRepository = creditRepository;
29     this.remboursementMapper = remboursementMapper;
30 }
31
32 // 2 usages
33 @Override
```

```
8 // YassineHachguer
9 @RestController
10 @RequestMapping("/clients")
11 public class ClientRestController {
12     // 6 usages
13     private final ClientService clientService;
14
15     // YassineHachguer
16     public ClientRestController(ClientService clientService) { this.clientService = clientService; }
17
18     // YassineHachguer
19     @GetMapping
20     public List<ClientDTO> getClients() { return clientService.listClients(); }
21
22     // YassineHachguer
23     @GetMapping("/{id}")
24     public ClientDTO getClient(@PathVariable Long id) { return clientService.getClient(id); }
25
26 }
27
```



```
10 @RequestMapping("/credits")
11 public class CreditRestController {
12     // 6 usages
13     private final CreditService creditService;
14
15     // YassineHachguer
16     public CreditRestController(CreditService creditService) { this.creditService = creditService; }
17
18     // YassineHachguer
19     @GetMapping
20     public List<CreditDTO> getCredits() { return creditService.listCredits(); }
21
22     // YassineHachguer
23     @GetMapping("/{id}")
24     public CreditDTO getCredit(@PathVariable Long id) { return creditService.getCredit(id); }
25
26     // YassineHachguer
27     @PostMapping
28     public CreditDTO saveCredit(@RequestBody CreditDTO creditDTO) { return creditService.saveCredit(creditDTO); }
29 }
30
```

```
localhost:8081/credits

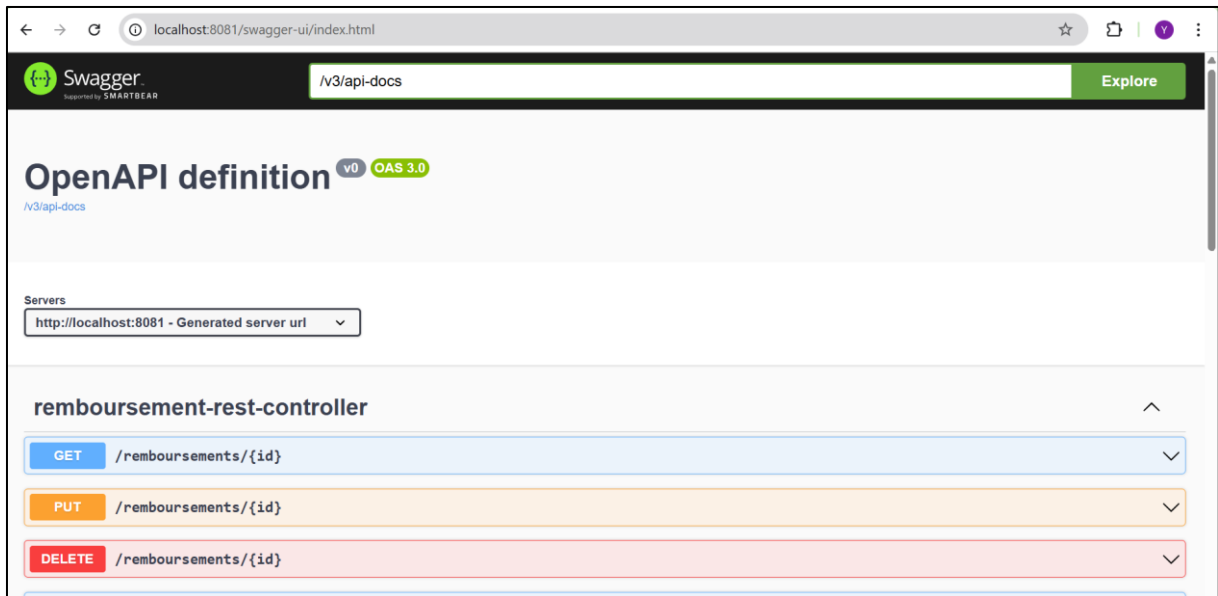
Impression élégante
{
  "id": 1,
  "dateDemande": "2025-05-19T09:26:20.594+00:00",
  "statut": "En cours",
  "dateAcception": null,
  "montant": 50000,
  "dureeRemboursement": 24,
  "tauxInteret": 5.5,
  "clientId": 1,
  "motif": "Achat Voiture"
},
{
  "id": 2,
  "dateDemande": "2025-05-19T09:26:20.602+00:00",
  "statut": "Accepté",
  "dateAcception": "2025-05-19T09:26:20.602+00:00",
  "montant": 300000,
  "dureeRemboursement": 120,
  "tauxInteret": 3.2,
  "clientId": 2,
  "typeBien": "Appartement"
},
{
  "id": 3,
  "dateDemande": "2025-05-19T09:26:20.604+00:00",
  "statut": "Rejeté",
  "dateAcception": null,
  "montant": 150000,
  "dureeRemboursement": 36,
  "tauxInteret": 6,
  "clientId": 2,
  "motif": "Investissement matériel",
  "raisonSociale": "SARL TechPlus"
}
}
```

```
Project
  repositories
  security
  service
  web
    ClientRestController
    CreditRestController
    RemboursementRestController
  JeeExamBackendApplication
    resources
    application.properties
    test
      target
      gitattributes
      gitignore
      mvnw
      mvnw.cmd
      pom.xml

RemboursementRestController.java
10 @RequestMapping("/remboursements")
11 public class RemboursementRestController {
12
13     private final RemboursementService remboursementService;
14
15     @YassineHachguer
16     public RemboursementRestController(RemboursementService remboursementService) {
17         this.remboursementService = remboursementService;
18     }
19
20     @YassineHachguer
21     @GetMapping
22     public List<RemboursementDTO> getRemboursements() { return remboursementService.list
23
24     @YassineHachguer
25     @GetMapping("/{id}")
26     public RemboursementDTO getRemboursement(@PathVariable Long id) {
27         return remboursementService.getRemboursement(id);
28     }
29 }
```

```
localhost:8081/remboursements

Impression élégante
{
  "id": 1,
  "date": "2025-05-19T09:26:20.606+00:00",
  "montant": 2200,
  "type": "Mensualité",
  "creditId": 1
},
{
  "id": 2,
  "date": "2025-05-19T09:26:20.607+00:00",
  "montant": 3000,
  "type": "Remboursement anticipé",
  "creditId": 2
}
}
```

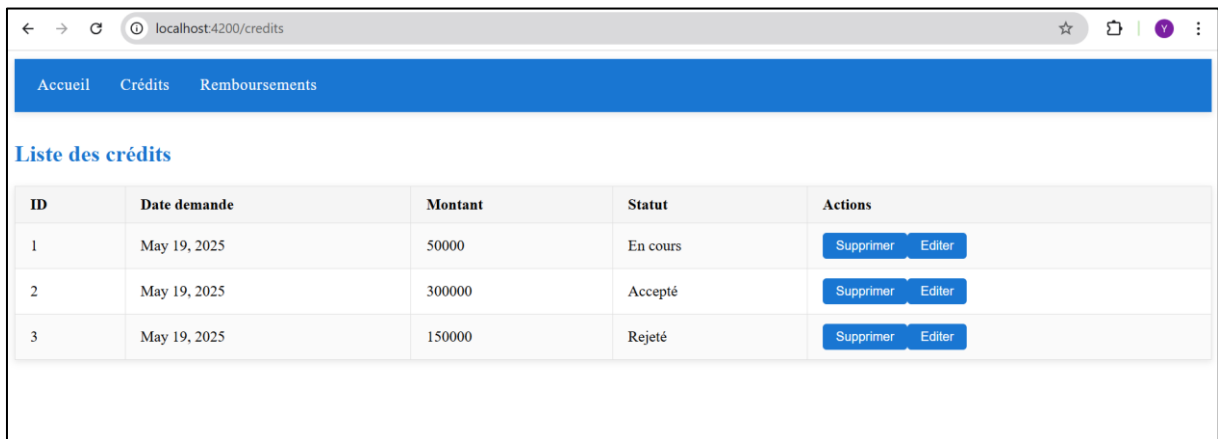
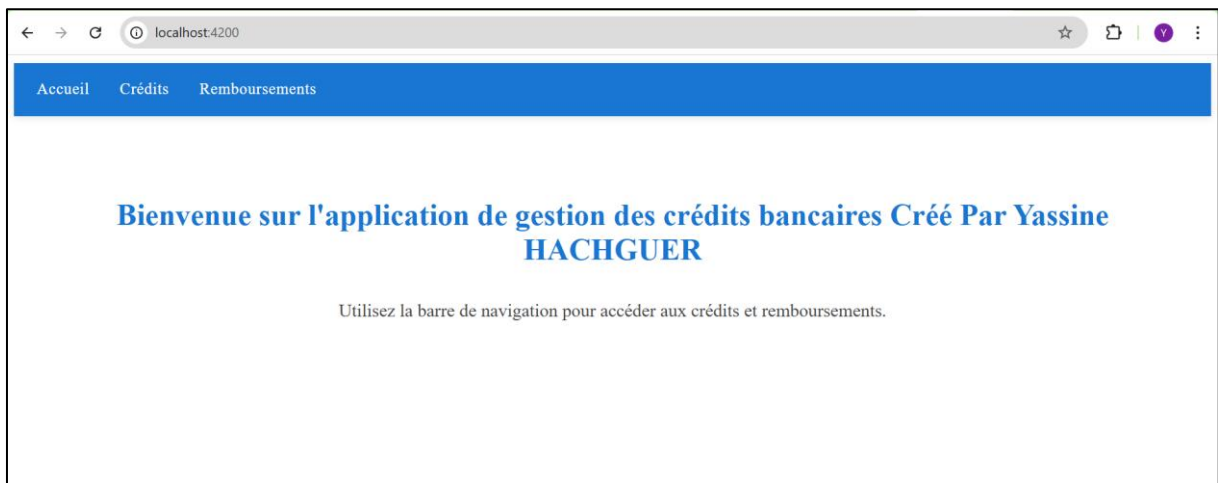
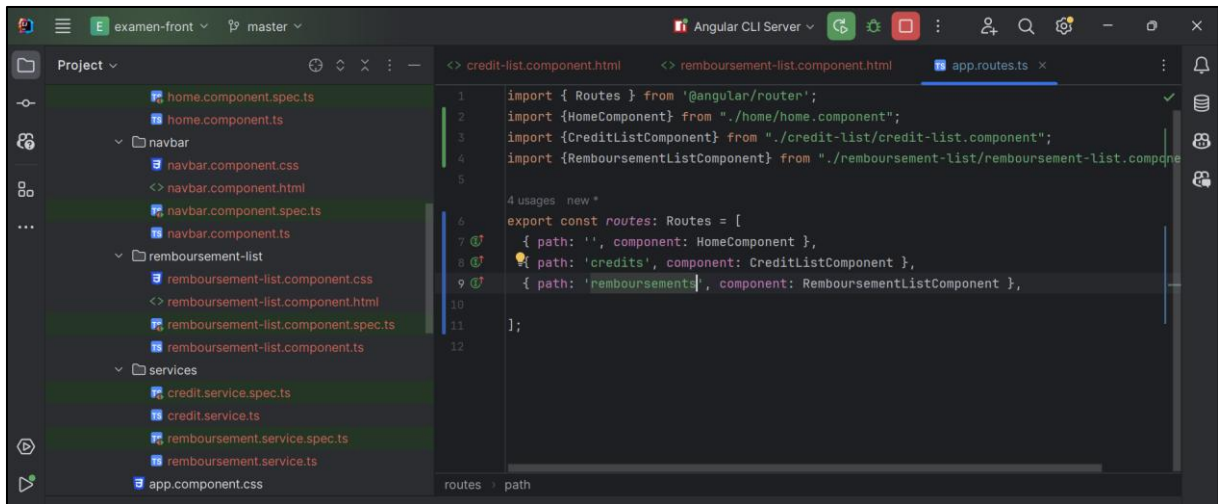


5. Proposer une application frontend en utilisant Angular Framework

Pour la structure du projet on a 3 composants :

Home, Navbar, crédit-list et remboursement-list

Et pour gérer les requêtes on a 2 services



<div> Accueil Crédits Remboursements </div>				
<div>Liste des remboursements</div>				
ID	Date	Montant	Type	Actions
1	May 19, 2025	2200	Mensualité	<div>Supprimer</div> <div>Editer</div>
2	May 19, 2025	3000	Remboursement anticipé	<div>Supprimer</div> <div>Editer</div>

- Sécuriser d'accès aux applications backend et frontend application en se basant sur Spring Security et Json Web Token avec un système d'authentification des utilisateurs avec 3 types de rôles «ROLE_CLIENT», « ROLE_EMPLOYE » et « ROLE_ADMIN » en choisissant des autorisations appropriées à ses rôles**