

Ecole Marocain Des Science De L'ingenierie

Rapport du projet JEE

**Developpement d'une application web JEE
Pour l'affichage des films
et gestion des cinemas**

Realise par:

Warakib abdelbasset

Voici mon github pour consulter l'application web
<https://github.com/warakibabdelbasset/cinemaJEE>

Contents

Indroduction :	2
Maven project	3
Spring:	4
1. Spring boot	4
2. Spring security	4
Thymeleaf:	4
Angular:	4
Capture d'ecran et commentaire:	4
• DAO	4
• Entities	8
• Security	14
• Service	15
• Web	16
• Thymeleaf	17
• Angular	19
Pom.xml	21
Screen de site web:	23
• User	23
• Admin	26

Indroduction :

JEE (*Java Entreprise Edition*) est la version entreprise de la plate-forme "Java" qui se compose de l'environnement "JSE" ainsi que de nombreuses API et composants destinés à une utilisation "côté serveur" au sein du système d'information de l'entreprise. Il s'agit donc d'une évolution du Java.

Dès son origine, Java a révolutionné plusieurs domaines de l'informatique, que ça soit la téléphonie, l'internet ou les applications d'entreprise. Par la suite, Sun avait réorganisé son offre autour de trois briques :

- Java Micro Edition (JME) qui cible les terminaux portables.
- Java Standard Edition (JSE) qui vise les postes clients.
- Java Entreprise Edition (JEE) qui définit le cadre d'un serveur d'applications et d'intégration.

Maven project

Maven est un outil de construction de projets (build) open source développé par la fondation Apache, initialement pour les besoins du projet Jakarta Turbine. Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java. Il permet notamment :

- d'automatiser certaines tâches : compilation, tests unitaires et déploiement des applications qui composent le projet
- de gérer des dépendances vis-à-vis des bibliothèques nécessaires au projet
- de générer des documentations concernant le projet

Au premier abord, il est facile de croire que Maven fait double emploi avec Ant. Ant et Maven sont tous les deux développés par le groupe Jakarta, ce qui prouve bien que leur utilité n'est pas aussi identique que cela. Ant, dont le but est d'automatiser certaines tâches répétitives, est plus ancien que Maven. Maven propose non seulement les fonctionnalités d'Ant mais en propose de nombreuses autres.

Pour gérer les dépendances du projet vis-à-vis de bibliothèques, Maven utilise un ou plusieurs dépôts qui peuvent être locaux ou distants.

Maven est extensible grâce à un mécanisme de plugins qui permettent d'ajouter des fonctionnalités.

Spring:

Spring est un framework open source pour construire et définir l'infrastructure d'une application Java, dont il facilite le développement et les tests. En 2004, Rod Johnson a écrit le livre Expert One-on-One J2EE Design and Development qui explique les raisons de la création de Spring

1. Spring boot

Spring Boot est un framework qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée en jar , totalement autonome. Ce qui nous intéresse particulièrement, puisque nous essayons de développer des Microservices

2. Spring security

Spring Security est un framework Java / Java EE qui fournit l'authentification, l'autorisation et d'autres fonctionnalités de sécurité pour les applications d'entreprise. Le projet a été lancé fin 2003 sous le nom de «Acegi Security» par Ben Alex, et il a été rendu public sous la licence Apache en mars 2004

Thymeleaf:

Angular:

Capture d'écran et commentaire:

- DAO

Tout d'abord on commence par la partie interface .En va cree les interfaces avec les methodes quand va utiliser pour cree les classes entities,car nous etes dans l'héritage faible qui s'impose sur la relation entre les interface et les class parce que si on veulent faire une maintenance il soit simple a l'intégrer.

Voici quelques screenshots des interfaces:

TP_CLASSE - cinema_project/src/main/java/com/cinema/dao/PlaceRepository.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Project Explorer CategorieRepository.java CinemaRepository.java FilmRepository.java PlaceRepository.java

```
1 package com.cinema.dao;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 @RepositoryRestResource
6 @CrossOrigin("http://localhost:4200")
7 public interface PlaceRepository extends JpaRepository<Place, Long>{
8
9 }
10
11 }
```

cinema_project [boot] [devtools]

src/main/java

- com.cinema
- CinemaProjectApplication.java
- com.cinema.dao
- CategorieRepository.java
- CinemaRepository.java
- FilmRepository.java
- PlaceRepository.java
- ProjectionRepository.java
- SalleRepository.java
- SeanceRepository.java
- TicketRepository.java
- VilleRepository.java

com.cinema.entities

- Categorie.java
- Cinema.java
- Film.java
- Place.java
- Projection.java
- ProjectionProj.java
- Salle.java
- Seance.java
- Ticket.java
- TicketProj.java
- Ville.java

com.cinema.security

- SecurityConfig.java
- SecurityControleur.java

com.cinema.service

- CinemaInitServiceImpl.java
- ICinemaInitService.java

com.cinema.web

src/main/resources

- static
- templates
- categoriesgest.html
- cinemasGest.html
- confirmation.html
- filmsGest.html
- form.html
- formville.html
- index.html
- notAuthorized.html
- placesGest.html
- projectionsGest.html
- salleGest.html
- seancesGest.html
- template1.html

Task List

Find All

Outline

Writable Smart Insert 1:1:0

TP_CLASSE - cinema_project/src/main/java/com/cinema/dao/ProjectionRepository.java - Eclipse IDE

```

1 package com.cinema.dao;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 @RepositoryRestResource
6 @CrossOrigin("http://localhost:4200")
7 public interface ProjectionRepository extends JpaRepository<Projection, Long>{
8
9 }
10
11
12
13
14
15
16

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/dao/SeanceRepository.java - Eclipse IDE

```

1 package com.cinema.dao;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 @RepositoryRestResource
6 @CrossOrigin("http://localhost:4200")
7 public interface SeanceRepository extends JpaRepository<Seance, Long>{
8
9 }
10
11
12
13
14
15
16

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/dao/CategorieRepository.java - Eclipse IDE

```

1 package com.cinema.dao;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5 @RepositoryRestResource
6 @CrossOrigin("http://localhost:4200")
7 public interface CategorieRepository extends JpaRepository<Categorie, Long>{
8
9 }
10
11
12
13
14

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/dao/CinemaRepository.java - Eclipse IDE

```

1 package com.cinema.dao;
2
3 import org.springframework.data.domain.Page;
4
5 @RepositoryRestResource
6 @CrossOrigin("http://localhost:4200")
7 public interface CinemaRepository extends JpaRepository<Cinema, Long>{
8
9     public Page<Cinema> findByNameContains(String keyword, Pageable pageable);
10    public Page<Cinema> findByVilleAndNameContains(Ville ville, String keyword, Pageable pageable);
11    public Page<Cinema> findByVille(Ville currentVille, Pageable pageable);
12
13
14
15
16
17
18
19
20
21
22

```

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/dao/FilmRepository.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Project Explorer Task List
CategorieRepository.java CinemaRepository.java FilmRepository.java
com.cinema.dao
com.cinema.entities
com.cinema.security
com.cinema.service
com.cinema.web
src/main/resources
static
templates
categorieGet.html
cinemasGet.html
confirmation.html
filmsGet.html
form.html
formville.html
index.html
notAuthorized.html
placeGet.html
placesGet.html
salesGet.html
seancesGet.html
template1.html

```

```

package com.cinema.dao;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import org.springframework.web.bind.annotation.CrossOrigin;
import java.util.List;

@Repository
@CrossOrigin(origins = "http://localhost:4200")
public interface FilmRepository extends JpaRepository<Film, Long> {
}

```

• Entities

Puis en passe au deusieme package .En va cree les class qui corespondant aux interfaces et en va initialiser et redefinir les method cree dans les interfaces. Et ce package (entities) va s'occupe de creation des tables on base de donne .

Voici quelque screen shoot de ces classes:

TP_CLASSE - cinema_project/src/main/java/com/cinema/entities/Categorie.java - Eclipse IDE

```

1 package com.cinema.entities;
2
3 import java.util.Collection;
4
5 @Entity
6 @Data
7 @NoArgsConstructor
8 @ToString
9 public class Categorie {
10     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
11     private Long id;
12     @Column(length = 75)
13     private String name;
14     @OneToOne(mappedBy = "categorie")
15     @JsonProperty(access = Access.READ_ONLY)
16     private Collection<Film> films;
17 }
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/entities/Cinema.java - Eclipse IDE

```

1 package com.cinema.entities;
2
3 import java.util.Collection;
4
5 @Entity
6 @Data
7 @NoArgsConstructor
8 @ToString
9 public class Cinema {
10     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
11     private Long id;
12     @Column(length = 75)
13     private String name;
14     private double longitude, latitude, altitude;
15     private int nombreSalles;
16     @OneToOne(mappedBy = "cinema")
17     private Collection<Salle> salles;
18     @ManyToOne
19     private Ville ville;
20 }
21
22
23
24
25
26
27
28
29
30
31
32
33
34

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/entities/Film.java - Eclipse IDE

```

package com.cinema.entities;
import java.sql.Date;
@Entity
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Film {
    @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
    private long id;
    @Column(length = 75)
    private String titre;
    @Column(length = 200)
    private String description;
    @Column(length = 255)
    private String realisateur;
    private Date dateSortie;
    private double duree;
    private String photo;
    @OneToOne(mappedBy="film")
    @JsonIgnore(access=Access.WRITE_ONLY)
    private Collection<Projection> projections;
    @ManyToOne
    private Categorie categorie;
}

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/entities/Place.java - Eclipse IDE

```

package com.cinema.entities;
import java.util.Collection;
@Entity
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Place {
    @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
    private long id;
    private int numero;
    private double longitude, latitude, altitude;
    @ManyToMany
    private Collection<Salle> salles;
    @OneToOne(mappedBy="place")
    @JsonIgnore(access = Access.WRITE_ONLY)
    private Collection<Ticket> tickets;
}

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/entities/Projection.java - Eclipse IDE

```

1 package com.cinema.entities;
2
3 import java.util.collection.*;
4
5 @Entity
6 @Data @NoArgsConstructor @AllArgsConstructor @ToString
7 public class Projection {
8     @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
9     private Long id;
10    private Date dateProjection;
11    private double prix;
12    @ManyToOne(access = Access.WRITE_ONLY)
13    @JsonManagedReference("projection")
14    private Sale sale;
15    @ManyToOne(access = Access.WRITE_ONLY)
16    private Film film;
17    @OneToOne(mappedBy="projection")
18    private Seance seance;
19    @OneToOne(mappedBy="projection")
20    private Ticket ticket;
21 }
22 
```

TP_CLASSE - cinema_project/src/main/java/com/cinema/entities/Salle.java - Eclipse IDE

```

1 package com.cinema.entities;
2
3 import java.util.collection.*;
4
5 @Entity
6 @Data @NoArgsConstructor @AllArgsConstructor @ToString
7 public class Salle {
8     @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
9     private Long id;
10    private String name;
11    private int nombrePlace;
12    @ManyToOneJoinColumn(name="id_cinema")
13    @JsonManagedReference("salle")
14    private Cinema cinema;
15    @OneToOne(mappedBy="salle")
16    private Projection projection;
17    @OneToOne(mappedBy="salle")
18    private Place place;
19    @OneToOne(mappedBy="salle")
20    private Seance seance;
21    @OneToOne(mappedBy="salle")
22    private Ticket ticket;
23    @OneToOne(mappedBy="salle")
24    private Ville ville;
25 }
26 
```

TP_CLASSE - cinema_project/src/main/java/com/cinema/entities/Ticket.java - Eclipse IDE

```

1 package com.cinema.entities;
2
3 import javax.persistence.Column;
4
5 @Entity
6 @Data @NoArgsConstructor @AllArgsConstructor @ToString
7 public class Ticket {
8     @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
9     private Long id;
10    @Column(length = 75)
11    private String nomClient;
12    private double prix;
13    @ManyToOne(mappedBy="ticket")
14    private Salle salle;
15    private Integer codePayment;
16    private boolean reserve;
17    @ManyToMany
18    private Place place;
19    @ManyToMany
20    private Projection projection;
21 }
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

```

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Project Explorer Task List

Find All A

Outline

com.cinema.e ^

Ticket

- getid(): Long
- getNomClient(): String
- getPrix(): double
- isReserve(): boolean
- getPlace(): Place
- getProjection(): Projection
- setPlace(Place place)
- setProjection(Projection projection)
- equals(Object obj): boolean
- canEqual(Object obj): boolean
- hashCode(): int
- toString(): String

TP_CLASSE - cinema_project/src/main/java/com/cinema/entities/Ville.java - Eclipse IDE

```

1 package com.cinema.entities;
2
3 import java.util.Collection;
4
5 @Entity
6 @Data @NoArgsConstructor @AllArgsConstructor @ToString
7 public class Ville {
8     @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
9     private Long id;
10    @Column(length = 75)
11    private String name;
12    private double longitude, latitude, altitude;
13    @OneToMany(mappedBy="ville")
14    private Collection<Cinemas> cinemas;
15 }
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

```

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Project Explorer Task List

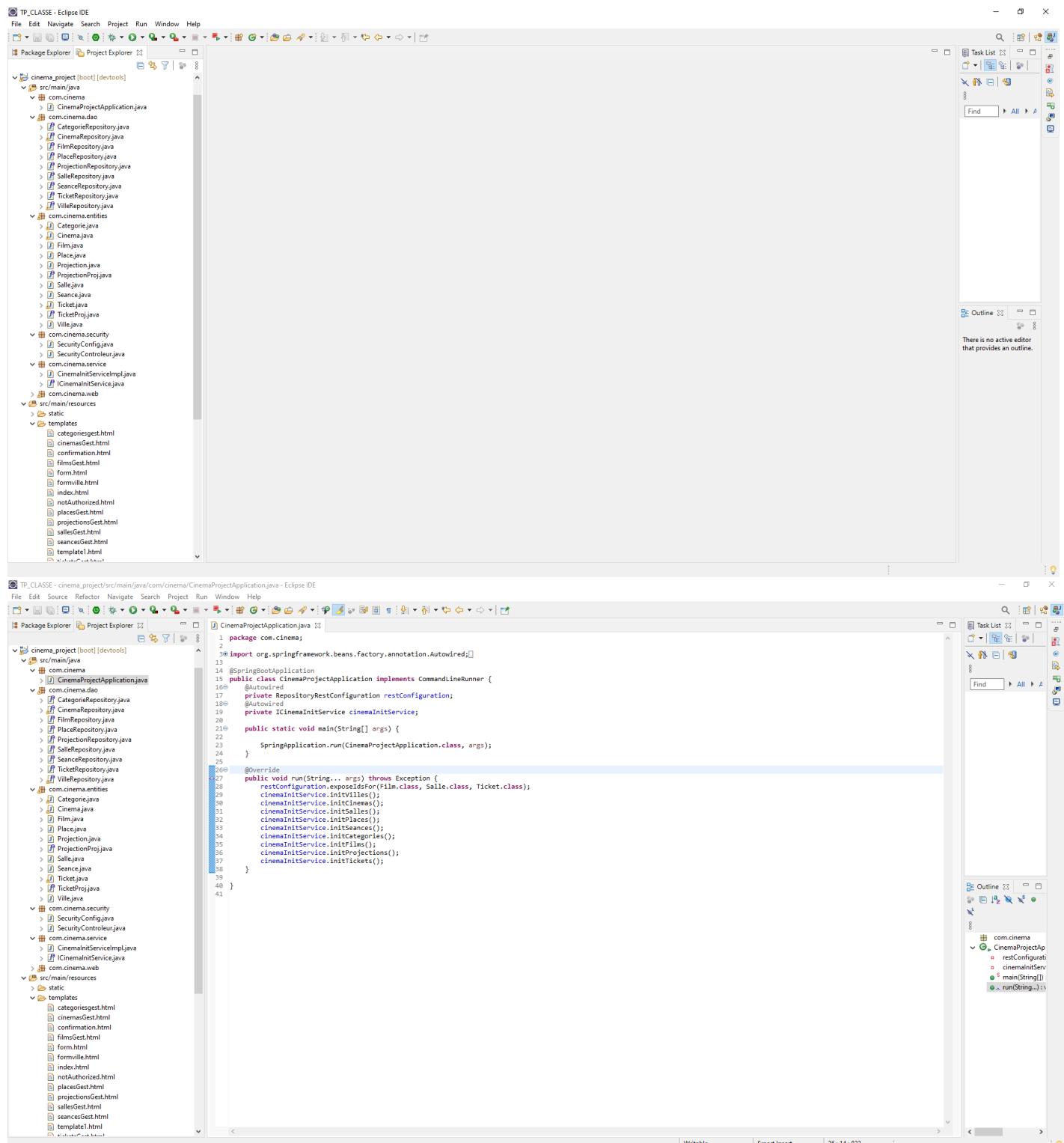
Find All A

Outline

com.cinema.e ^

Ville

- getId(): Long
- getName(): String
- getLongitude(): double
- getLatitude(): double
- getAltitude(): double
- getCinemas(): Collection<Cinemas>
- equals(Object obj): boolean
- canEqual(Object obj): boolean
- hashCode(): int
- toString(): String



● Security

Dans cette partie on est concentre sur spring security qui nous donne la possibilite de cree et gerer les compte pour l'application web .il nous offre une page d'authentification qu'il est securise par spring security comme ont peux modifier l'etat d'authentification et la page themplete.

Voici quelque screen shoot :

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "TP_CLASSE - cinema_project". It includes packages like "com.cinema", "com.cinema.entities", "com.cinema.security", and "com.cinema.service".
- Code Editor:** Displays the content of "SecurityController.java". The code defines a class "SecurityControleur" with a single method "error()".
- Outline View:** On the right, it shows the outline of the current file, listing "SecurityConfig.java" and "SecurityControleur.java".
- Task List:** Shows a single entry: "com.cinema.security.error():String".

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "TP_CLASSE - cinema_project". It includes packages like "com.cinema", "com.cinema.entities", "com.cinema.security", and "com.cinema.service".
- Code Editor:** Displays the content of "SecurityConfig.java". The code extends "WebSecurityConfigurerAdapter" and configures authentication and HTTP security.
- Outline View:** On the right, it shows the outline of the current file, listing "SecurityConfig.java" and "SecurityControleur.java".
- Task List:** Shows multiple entries related to "SecurityConfig.java": "com.cinema.security.SecurityConfig", "configure(Auth)", "configure(Http)", and "passwordEncoder".

● Service

Voici quelques screen shot pour la partie service :

```

package com.cinema.service;

import java.text.SimpleDateFormat;
import javax.annotation.PostConstruct;
import javax.inject.Inject;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.cinema.dao.CategorieRepository;
import com.cinema.dao.CinemaRepository;
import com.cinema.dao.FilmRepository;
import com.cinema.dao.PlaceRepository;
import com.cinema.dao.ProjectionRepository;
import com.cinema.dao.SalleRepository;
import com.cinema.dao.SeanceRepository;
import com.cinema.dao.TicketRepository;
import com.cinema.dao.VilleRepository;
import com.cinema.entities.Categorie;
import com.cinema.entities.Cinema;
import com.cinema.entities.Place;
import com.cinema.entities.Projection;
import com.cinema.entities.Salle;
import com.cinema.entities.Seance;
import com.cinema.entities.Ticket;
import com.cinema.entities.Ville;
import com.cinema.security.SecurityConfigJava;
import com.cinema.service.CinemaInitServiceImpl;
import com.cinema.web.CinemaControllerJava;
import com.cinema.web.CinemaRestControllerJava;
import com.cinema.web.CinemaTicketControllerJava;
import static com.cinema.entities.Categorie.*;
import static com.cinema.entities.Cinema.*;
import static com.cinema.entities.Place.*;
import static com.cinema.entities.Projection.*;
import static com.cinema.entities.Salle.*;
import static com.cinema.entities.Seance.*;
import static com.cinema.entities.Ticket.*;
import static com.cinema.entities.Ville.*;

@Service
@Transactional
public class CinemaInitServiceImpl implements ICinemaInitService {
    @Inject
    private VilleRepository villeRepository;
    @Inject
    private CinemaRepository cinemaRepository;
    @Inject
    private PlaceRepository placeRepository;
    @Inject
    private SeanceRepository seanceRepository;
    @Inject
    private FilmRepository filmRepository;
    @Inject
    private ProjectionRepository projectionRepository;
    @Inject
    private CategorieRepository categorieRepository;
    @Inject
    private TicketRepository ticketRepository;

    @Override
    public void initVilles() {
        Stream.of("Casablanca", "Marrakech", "Rabat", "Tanger").forEach(nameVille -> {
            Ville ville = new Ville();
            ville.setVilleName(nameVille);
            villeRepository.save(ville);
        });
    }

    @Override
    public void initCinemas() {
        villeRepository.findAll().forEach(v -> {
            Stream.of("Megarama", "IMAX", "FOUJOUN", "CHAHRAZAD", "DAOULIZ").forEach(nameCinema -> {
                Cinema cinema = new Cinema();
                cinema.setVille(v);
                cinema.setNombreSalles(3 + (int) (Math.random() * 7));
                cinema.setVille(v);
                cinemaRepository.save(cinema);
            });
        });
    }

    @Override
    public void initPlaces() {
        cinemaRepository.findAll().forEach(cinema -> {
            for (int i = 0; i < cinema.getNombreSalles(); i++) {
                Salle salle = new Salle();
                salle.setName("Salle " + (i + 1));
                salle.setCinema(cinema);
                salle.setNombrePlaces(12 + (int) (Math.random() * 5));
                salleRepository.save(salle);
            }
        });
    }

    @Override
    public void initSalles() {
        cinemaRepository.findAll().forEach(cinema -> {
            for (int i = 0; i < cinema.getNombreSalles(); i++) {
                Salle salle = new Salle();
                salle.setName("Salle " + (i + 1));
                salle.setCinema(cinema);
                salle.setNombrePlaces(12 + (int) (Math.random() * 5));
                salleRepository.save(salle);
            }
        });
    }

    @Override
    public void initCategories() {
        Stream.of("Histoire", "Actions", "Fiction", "Drama").forEach(cat -> {
            Categorie categorie = new Categorie();
            categorie.setName(cat);
            categorieRepository.save(categorie);
        });
    }

    @Override
    public void initFilms() {
        double[] durees = new double[] { 1, 1.5, 2, 2.5, 3 };
        List<Categorie> categories = categorieRepository.findAll();
        Stream.of("vikings.jpg", "yoursineapple.jpg", "naruto.jpg", "yourname.jpg").forEach(titreFilm -> {
            Film film = new Film();
            film.setTitre(titreFilm);
            film.setDuree(durees[new Random().nextInt(durees.length)]);
            film.setPhoto(titreFilm.replaceAll(" ", "") + ".jpg");
            film.setCategorie(categories.get(new Random().nextInt(categories.size()))));
            filmRepository.save(film);
        });
    }

    @Override
    public void initProjections() {
        double[] prices = new double[] { 30, 50, 60, 70, 90, 100 };
        List<Film> films = filmRepository.findAll();
        villeRepository.findAll().forEach(ville -> {
            ville.getPlaces().forEach(place -> {
                place.getSalles().forEach(salle -> {
                    int index = new Random().nextInt(films.size());
                    Film film = films.get(index);
                    Seance seance = new Seance();
                    Projection projection = new Projection();
                    projection.setDate(new Date());
                    projection.setPlace(place);
                    projection.setSalle(salle);
                    projection.setSeance(seance);
                    projectionRepository.save(projection);
                });
            });
        });
    }

    @Override
    public void initTickets() {
        ...
    }
}

```



```

package com.cinema.service;

import java.text.SimpleDateFormat;
import javax.annotation.PostConstruct;
import javax.inject.Inject;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.cinema.dao.CategorieRepository;
import com.cinema.dao.CinemaRepository;
import com.cinema.dao.FilmRepository;
import com.cinema.dao.PlaceRepository;
import com.cinema.dao.ProjectionRepository;
import com.cinema.dao.SalleRepository;
import com.cinema.dao.SeanceRepository;
import com.cinema.dao.TicketRepository;
import com.cinema.dao.VilleRepository;
import com.cinema.entities.Categorie;
import com.cinema.entities.Cinema;
import com.cinema.entities.Place;
import com.cinema.entities.Projection;
import com.cinema.entities.Salle;
import com.cinema.entities.Seance;
import com.cinema.entities.Ticket;
import com.cinema.entities.Ville;
import com.cinema.security.SecurityConfigJava;
import com.cinema.service.CinemaInitServiceImpl;
import com.cinema.web.CinemaControllerJava;
import com.cinema.web.CinemaRestControllerJava;
import com.cinema.web.CinemaTicketControllerJava;
import static com.cinema.entities.Categorie.*;
import static com.cinema.entities.Cinema.*;
import static com.cinema.entities.Place.*;
import static com.cinema.entities.Projection.*;
import static com.cinema.entities.Salle.*;
import static com.cinema.entities.Seance.*;
import static com.cinema.entities.Ticket.*;
import static com.cinema.entities.Ville.*;

@Service
@Transactional
public class CinemaInitServiceImpl implements ICinemaInitService {
    @Inject
    private VilleRepository villeRepository;
    @Inject
    private CinemaRepository cinemaRepository;
    @Inject
    private PlaceRepository placeRepository;
    @Inject
    private SeanceRepository seanceRepository;
    @Inject
    private FilmRepository filmRepository;
    @Inject
    private ProjectionRepository projectionRepository;
    @Inject
    private CategorieRepository categorieRepository;
    @Inject
    private TicketRepository ticketRepository;

    @PostConstruct
    public void initVilles() {
        Stream.of("Casablanca", "Marrakech", "Rabat", "Tanger").forEach(nameVille -> {
            Ville ville = new Ville();
            ville.setVilleName(nameVille);
            villeRepository.save(ville);
        });
    }

    @Override
    public void initCinemas() {
        villeRepository.findAll().forEach(v -> {
            Stream.of("Megarama", "IMAX", "FOUJOUN", "CHAHRAZAD", "DAOULIZ").forEach(nameCinema -> {
                Cinema cinema = new Cinema();
                cinema.setVille(v);
                cinema.setNombreSalles(3 + (int) (Math.random() * 7));
                cinema.setVille(v);
                cinemaRepository.save(cinema);
            });
        });
    }

    @Override
    public void initPlaces() {
        cinemaRepository.findAll().forEach(cinema -> {
            for (int i = 0; i < cinema.getNombreSalles(); i++) {
                Salle salle = new Salle();
                salle.setName("Salle " + (i + 1));
                salle.setCinema(cinema);
                salle.setNombrePlaces(12 + (int) (Math.random() * 5));
                salleRepository.save(salle);
            }
        });
    }

    @Override
    public void initSalles() {
        cinemaRepository.findAll().forEach(cinema -> {
            for (int i = 0; i < cinema.getNombreSalles(); i++) {
                Salle salle = new Salle();
                salle.setName("Salle " + (i + 1));
                salle.setCinema(cinema);
                salle.setNombrePlaces(12 + (int) (Math.random() * 5));
                salleRepository.save(salle);
            }
        });
    }

    @Override
    public void initCategories() {
        Stream.of("Histoire", "Actions", "Fiction", "Drama").forEach(cat -> {
            Categorie categorie = new Categorie();
            categorie.setName(cat);
            categorieRepository.save(categorie);
        });
    }

    @Override
    public void initFilms() {
        double[] durees = new double[] { 1, 1.5, 2, 2.5, 3 };
        List<Categorie> categories = categorieRepository.findAll();
        Stream.of("vikings.jpg", "yoursineapple.jpg", "naruto.jpg", "yourname.jpg").forEach(titreFilm -> {
            Film film = new Film();
            film.setTitre(titreFilm);
            film.setDuree(durees[new Random().nextInt(durees.length)]);
            film.setPhoto(titreFilm.replaceAll(" ", "") + ".jpg");
            film.setCategorie(categories.get(new Random().nextInt(categories.size()))));
            filmRepository.save(film);
        });
    }

    @Override
    public void initProjections() {
        double[] prices = new double[] { 30, 50, 60, 70, 90, 100 };
        List<Film> films = filmRepository.findAll();
        villeRepository.findAll().forEach(ville -> {
            ville.getPlaces().forEach(place -> {
                place.getSalles().forEach(salle -> {
                    int index = new Random().nextInt(films.size());
                    Film film = films.get(index);
                    Seance seance = new Seance();
                    Projection projection = new Projection();
                    projection.setDate(new Date());
                    projection.setPlace(place);
                    projection.setSalle(salle);
                    projection.setSeance(seance);
                    projectionRepository.save(projection);
                });
            });
        });
    }

    @Override
    public void initTickets() {
        ...
    }
}

```

● Web

La partie WEB c'est la partie ou on trouvent le contrôleur qui nous aide à gérer les classes et les méthodes

Voici quelques screen shot :

```

84     model.addAttribute("cinemas", cinemas.getConten());
85     model.addAttribute("pages", new int[cinemas.getTotalPages()]);
86     model.addAttribute("currentPage", page);
87     model.addAttribute("keyword", keyword);
88     model.addAttribute("ville", ville);
89     return "cinemasGet";
90   }
91   @GetMapping("/sallesGet")
92   public String salles(Model model) {
93     List<Salle> salles = salleRepository.findAll();
94     model.addAttribute("salles", salles);
95     return "sallesGet";
96   }
97   @GetMapping("/seancesGet")
98   public String seances(Model model) {
99     List<Seance> seances = seanceRepository.findAll();
100    model.addAttribute("seances", seances);
101    return "seancesGet";
102  }
103  @GetMapping("/filmsGet")
104  public String films(Model model) {
105    List<Film> films = filmRepository.findAll();
106    model.addAttribute("films", films);
107    return "filmsGet";
108  }
109  @GetMapping("/categoriesGet")
110  public String categories(Model model) {
111    List<Categorie> categories = categorieRepository.findAll();
112    model.addAttribute("categories", categories);
113    return "categoriesGet";
114  }
115  @GetMapping("/projectionsGet")
116  public String projections(Model model) {
117    List<Projection> projections = projectionRepository.findAll();
118    model.addAttribute("projections", projections);
119    return "projectionsGet";
120  }
121  @GetMapping("/ticketsGet")
122  public String tickets(Model model) {
123    List<Ticket> tickets = ticketRepository.findAll();
124    model.addAttribute("tickets", tickets);
125    return "ticketsGet";
126  }
127  @GetMapping("/placesGet")
128  public String places(Model model) {
129    List<Place> places = placeRepository.findAll();
130    model.addAttribute("places", places);
131    return "placesGet";
132  }
133  @GetMapping("/form")
134  public String form(Model model) {
135    model.addAttribute("film", new Film());
136    return "form";
137  }
138  @PostMapping("savefile")
139  public String savefile(Film film) {
140    filmRepository.save(film);
141    return "confirmation";
142  }
143}

```

```

1 package com.cinema.web;
2
3 import java.util.List;
4
5 @Controller
6 public class CinemaController {
7
8   @Autowired
9   private VilleRepository villeRepository;
10  @Autowired
11  private CinemaRepository cinemaRepository;
12  @Autowired
13  private SalleRepository salleRepository;
14  @Autowired
15  private PlaceRepository placeRepository;
16  @Autowired
17  private SeanceRepository seanceRepository;
18  @Autowired
19  private FilmRepository filmRepository;
20  @Autowired
21  private ProjectionRepository projectionRepository;
22  @Autowired
23  private CategorieRepository categorieRepository;
24  @Autowired
25  private TicketRepository ticketRepository;
26
27  @GetMapping("/index")
28  public String index() {
29    return "index";
30  }
31  @GetMapping("/villesGet")
32  public String villes(Model model,
33    @RequestParam(name="page", defaultValue = "0") int page,
34    @RequestParam(name="size", defaultValue = "5") int size,
35    @RequestParam(name="keyword", defaultValue = "", required = false) String keyword) {
36    Page<Ville> villes = villeRepository.findByNameContains(keyword, PageRequest.of(page,size));
37    model.addAttribute("villes", villes.getConten());
38    model.addAttribute("pages", new int[villes.getTotalPages()]);
39    model.addAttribute("currentPage", page);
40    model.addAttribute("keyword", keyword);
41    return "villesGet";
42  }
43  @GetMapping("/cinemasGet")
44  public String cinemas(Model model,
45    @RequestParam(name="page", defaultValue = "0") int page,
46    @RequestParam(name="size", defaultValue = "5") int size,
47    @RequestParam(name="ville", defaultValue = "1") Long ville,
48    @RequestParam(name="keyword", defaultValue = "", required = false) String keyword) {
49    Ville currentVille = villeRepository.findById(ville).get();
50    //Page<Cinema> cinemas = cinemaRepository.findByNameAndVille(keyword, currentVille, PageRequest.of(page,size));
51    Page<Cinema> cinemas = cinemaRepository.findByVilleNameContains(keyword, PageRequest.of(page,size));
52    model.addAttribute("villes", villes.getConten());
53    model.addAttribute("cinemas", cinemas.getConten());
54    model.addAttribute("pages", new int[cinemas.getTotalPages()]);
55    model.addAttribute("currentPage", page);
56    model.addAttribute("ville", ville);
57    return "cinemasGet";
58  }

```

TP_CLASSE - cinema_project/src/main/java/com/cinema/web/CinemaRestController.java - Eclipse IDE

```

1 package com.cinema.web;
2
3 import java.io.File;
4
5 @RestController
6 @CrossOrigin("http://localhost:4200")
7 public class CinemaRestController {
8     @Autowired
9     private FilmRepository filmRepository;
10    @Autowired
11    private TicketRepository ticketRepository;
12    @GetMapping(path = "/imagefile/{id}", produces = MediaType.IMAGE_JPEG_VALUE)
13    public byte[] image(@PathVariable(name = "id")long id) throws Exception {
14        public byte[] image(@PathVariable("id")long id) throws Exception {
15            Film f = filmRepository.findById(id).get();
16            String photoName = f.getPhoto();
17            File file = new File(System.getProperty("user.home")+"/cinema/images/"+photoName);
18            Path path = Paths.get(file.toURI());
19            return Files.readAllBytes(path);
20        }
21    }
22
23    @PostMapping("/payTicket")
24    @Transactional
25    public List<Ticket> payTicket(@RequestBody TicketForm ticketForm){
26        List<Ticket> listTickets = new ArrayList<Ticket>();
27        ticketForm.getTickets().forEach(idTicket -> {
28            Ticket ticket = ticketRepository.findById(idTicket).get();
29            ticket.setNomClient(ticketForm.getNomClient());
30            ticket.setReserve(true);
31            ticketRepository.save(ticket);
32            listTickets.add(ticket);
33        });
34        return listTickets;
35    }
36
37    @Data
38    class TicketForm{
39        private String nomClient;
40        private List<Long> tickets = new ArrayList<>();
41        private Integer codePayment;
42    }
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68

```

- Thymeleaf

Thymeleaf est un moteur de modèle Java côté serveur moderne pour les environnements Web et autonomes. L'objectif principal de Thymeleaf est d'apporter des modèles naturels élégants à votre flux de travail de développement - du HTML qui peut être correctement affiché dans les navigateurs et également fonctionner comme des prototypes statiques, permettant une collaboration plus étroite dans les équipes de développement. Avec des modules pour Spring Framework, une multitude d'intégrations avec vos outils préférés et la possibilité de brancher vos propres fonctionnalités, Thymeleaf est idéal pour le développement Web JVM HTML5 moderne - bien qu'il puisse faire beaucoup plus.

Voici quelques exemples :

TP_CLASSE - cinema_project/src/main/resources/templates/villesGest.html - Eclipse IDE

```

1 | !DOCTYPE html
2 | <html xmlns="http://www.thymeleaf.org" xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout" layout:decorator="template1" xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
3 | <head>
4 |   <meta charset="utf-8">
5 |   <title>Insert title here</title>
6 |   <link rel="stylesheet" type="text/css"
7 |     href="/webjars/bootstrap/4.1.3/css/bootstrap.min.css">
8 | </head>
9 | <body>
10|   <div layout:fragment="pageContent">
11|     <div class="container mt-5">
12|       <div class="card">
13|         <div class="card-header">Liste des Villes</div>
14|         <div class="card-body">
15|           <form method="get" th:action="@{/villesgest}">
16|             <div class="form-group">
17|               <label>Name</label> <input type="text" name="keyword" th:value="${keyword}">
18|               <button class="btn btn-primary">Search</button>
19|             </div>
20|           </form>
21|           <table class="table">
22|             <thead>
23|               <tr>
24|                 <th>ID</th>
25|                 <th>Name</th>
26|                 <th>Cinemas</th>
27|               </tr>
28|             </thead>
29|             <tbody>
30|               <tr>
31|                 <td>${v.id}</td>
32|                 <td>${v.name}</td>
33|                 <td><a class="btn btn-success" th:href="@{/cinemasgest(ville=${v.id})}">Show more</a></td>
34|                 <td><a class="btn btn-warning" th:href="@{/editville(id=${v.id})}">edit</a></td>
35|               </tr>
36|             </tbody>
37|           </table>
38|
39|           <ul class="nav nav-pills">
40|             <li th:each="v:${villes}">
41|               <li th:each="page, status:${pages}">
42|                 <th:if test="currentPage==status.index ? 'btn btn-primary' : 'btn'">
43|                   <th:href="#">${villesgest(page=${page}, status.index, keyword=${keyword})}</th:href>
44|                 <th:text>${status.index}</th:text>
45|               </li>
46|             </li>
47|           </ul>
48|
49|         </div>
50|       </div>
51|     </div>

```

TP_CLASSE - cinema_project/src/main/resources/templates/formville.html - Eclipse IDE

```

1 | !DOCTYPE html
2 | <html xmlns="http://www.thymeleaf.org" xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout" layout:decorator="template1">
3 | <head>
4 |   <meta charset="utf-8">
5 |   <title>Insert title here</title>
6 |   <body>
7 |     <div layout:fragment="pageContent">
8|       <h1>Add ville</h1>
9|       <div class="container">
10|         <form th:action="@{/saveville}" method="post">
11|           <div class="form-group">
12|             <label class="control-label" id=":id"><label>
13|               <label class="control-label" th:text="${ville.id}"><label>
14|               <input type="hidden" name="id" class="form-control" th:value="${ville.id}">
15|             </label>
16|           </div>
17|           <div class="form-group">
18|             <label class="control-label" name="name"><label>
19|               <input type="text" name="name" class="form-control" th:value="${ville.name}">
20|             </label>
21|           </div>
22|           <button class="btn btn-success" type="submit" save>Save</button>
23|         </form>
24|       </div>
25|     </div>
26|   </div>
27| </body>
28| </html>
29| 
```

```

<!DOCTYPE html>
<html xmlns="http://www.ultraq.net.nz/thymeleaf/layout"
      xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity5">
    <head>
        <meta charset="utf-8"/>
        <title>Insert title here</title>
        <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/4.1.3/css/bootstrap.min.css"/>
        <script src="/webjars/jquery/3.4.1/jquery.min.js"></script>
        <script src="/webjars/bootstrap/4.1.3/js/bootstrap.min.js"></script>
    </head>
    <body>
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <a class="navbar-brand" href="#">Navbar</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item active">
                        <a class="nav-link" th:href="@{villegest}">Home <span class="sr-only">(current)</span></a>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Link</a>
                    </li>
                    <li class="nav-item dropdown">
                        <a href="#" class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" sec:authorize="hasRole('ADMIN')">navbarDropdown</a>
                        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                            <a class="dropdown-item" th:href="@{form}">Add film</a>
                            <a class="dropdown-item" href="#">Add cinema</a>
                            <a class="dropdown-item" href="#">Add ville</a>
                        </div>
                    </li>
                </ul>
                <li class="nav-item dropdown">
                    <a href="#" class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" sec:authentication="name">navbarDropdown</a>
                    <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                        <a href="#" class="dropdown-item" href="#" sec:authorize="(isAuthenticated())" href="#">Login</a>
                        <a class="dropdown-item" href="#">Profile</a>
                        <a class="dropdown-item" th:href="@{logout}">Logout</a>
                    </div>
                </li>
            </div>
        </nav>
        <section layout:fragment="pageContent">
        </section>
    </body>

```

● Angular

Angular (communément appelé "Angular 2+" ou "Angular v2 et plus")^{2,3} est un cadriel (framework) côté client, open source, basé sur TypeScript, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète de AngularJS, cadriel construit par la même équipe.

Voici quelque screen shoot :

IntelliJ IDEA screenshot showing two tabs open:

- Top Tab:** Cinema-Front-web - cinema.service.spec.ts
- Bottom Tab:** Cinema-Front-web - cinema.component.ts

Project Structure:

```

Cinema-Front-web
  |- src
    |- app
      |- cinema
        |- cinema.component.css
        |- cinema.component.html
        |- cinema.component.spec.ts
        |- cinema.component.ts
      |- services
        |- cinema.service.spec.ts
        |- cinema.service.ts
        |- app.component.css
        |- app.component.html
        |- app.component.spec.ts
        |- app.component.ts
        |- app.module.ts
        |- app-routing.module.ts
    |- assets
    |- environments
    |- favicon.ico
    |- index.html
    |- main.ts
    |- polyfills.ts
    |- styles.css
    |- tests.ts
    |- editorconfig
    |- gitignore
    |- angular.json
    |- browserlist
    |- karma.conf.js
    |- package.json
    |- package-lock.json
    |- README.md
    |- tsconfig.app.json
    |- tsconfig.json
    |- tsconfig.spec.json
    |- tslint.json
  |- External Libraries
  |- Scratches and Consoles

```

Code Editor (Top Tab: cinema.service.spec.ts):

```

import { TestBed } from '@angular/core/testing';
import { CinemaService } from './cinema.service';

describe('CinemaService', () => {
  let service: CinemaService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(CinemaService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
})

```

Code Editor (Bottom Tab: cinema.component.ts):

```

import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { CinemaService } from '../services/cinema.service';

@Component({
  selector: 'app-cinema',
  templateUrl: './cinema.component.html',
  styleUrls: ['./cinema.component.css']
})
export class CinemaComponent implements OnInit {
  public villes: public Cinemas;
  public currentVille;
  public currentCinema;
  public currentProjection;
  public salles;
  public selectedTickets: any[];

  constructor(public cinemaService:CinemaService ) {}

  ngOnInit(): void {
    this.cinemaService.getVilles().subscribe( next: data=>{
      this.villes=data;
    }, error: err =>{
      console.log(err);
    })
  }

  onGetCinemas(v: any) {
    this.currentVille=v;
    this.salles=undefined;
    this.cinemaService.getCinemas(v).
    subscribe( next: data=>{
      this.Cinemas=data;
    }, error: err =>{
      console.log(err);
    })
  }
}

```

```

4  @Injectable({
5    providedIn: 'root'
6  })
7  export class CinemaService {
8
9    public host:string='http://localhost:8885'
10   constructor(public http:HttpClient) {}
11   public getVilles(){
12     return this.http.get(` ${this.host} /villes`)
13   }
14
15   getCinemas(v) {
16     return this.http.get(v._links.cinemas.href);
17   }
18
19   getSales(c){
20     return this.http.get(c._links.Sales.href);
21   }
22
23   getProjections(sale: any) {
24     let url=sale._links.projections.href.replace("{?projection}","");
25     return this.http.get(` ${url} ?projection=p1`);
26   }
27
28   getTicketsPlaces(p) {
29     let url=p._links.tickets.href.replace("{?projection}","");
30     return this.http.get(` ${url} ?projection=ticketPro`);
31   }
32
33   payeTickets(dataform: any) {
34     return this.http.post(` ${this.host} /payerTickets`,dataform);
35   }
36
37 }

```

Pom.xml

Un modèle d'objet de projet ou POM est l'unité de travail fondamentale de Maven. Il s'agit d'un fichier XML qui contient des informations sur le projet et les détails de configuration utilisés par Maven pour générer le projet. Il contient des valeurs par défaut pour la plupart des projets. Par exemple, le répertoire de construction, qui est la cible; le répertoire source, qui est src / main / java; le répertoire source de test, qui est src / test / java; etc. Lors de l'exécution d'une tâche ou d'un objectif, Maven recherche le POM dans le répertoire actuel. Il lit le POM, obtient les informations de configuration nécessaires, puis exécute l'objectif.

Voici quelque screen shoot ce fichier:

TP_CLASSE - cinema_project/pom.xml - Eclipse IDE

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.cinema.security</groupId>
    <artifactId>SecurityConfigJava</artifactId>
    <version>0.1-SNAPSHOT</version>
    <name>cinema project</name>
    <description>Cinema management</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.2.7.RELEASE</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>com.uneez</groupId>
            <artifactId>cinema_project</artifactId>
            <version>0.1-SNAPSHOT</version>
            <name>cinema project</name>
            <description>Cinema management</description>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
            <version>2.2.7.RELEASE</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
            <version>2.2.7.RELEASE</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <version>2.2.7.RELEASE</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>bootstrap4-jss</artifactId>
            <version>4.1.3</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.thymeleaf.extras</groupId>
            <artifactId>thymeleaf-extras-springsecurity5</artifactId>
            <version>3.0.4.RELEASE</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <version>2.2.7.RELEASE</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>1.4.196</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.junit.vintage</groupId>
            <artifactId>junit-vintage-engine</artifactId>
            <version>5.4.1</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>nz.net.ultrq</groupId>
            <artifactId>thymeleaf-layout-dialect</artifactId>
            <version>3.0.2</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>jquery</artifactId>
            <version>3.4.1</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>mysql:mysql-connector-java</groupId>
            <version>8.0.15</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
            <version>2.2.7.RELEASE</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <version>5.3.1</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>2.2.7.RELEASE</version>
            <type>plugin</type>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>2.2.7.RELEASE</version>
            </plugin>
        </plugins>
    </build>

```

TP_CLASSE - cinema_project/pom.xml - Eclipse IDE

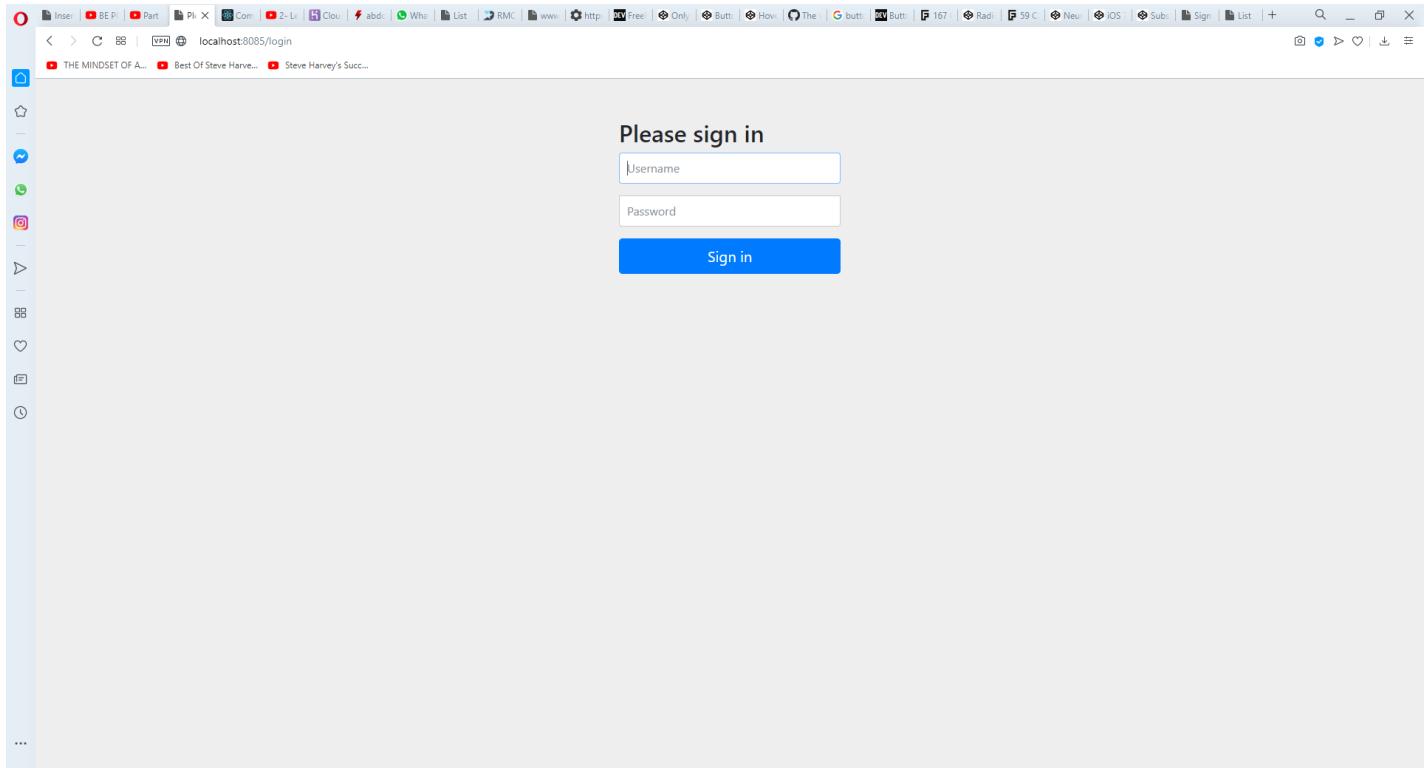
```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.cinema.security</groupId>
    <artifactId>SecurityConfigJava</artifactId>
    <version>0.1-SNAPSHOT</version>
    <name>cinema project</name>
    <description>Cinema management</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-rest</artifactId>
            <version>2.2.7.RELEASE</version>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>nz.net.ultrq</groupId>
            <artifactId>thymeleaf-layout-dialect</artifactId>
            <version>3.0.2</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>jquery</artifactId>
            <version>3.4.1</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>mysql:mysql-connector-java</groupId>
            <version>8.0.15</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
            <version>2.2.7.RELEASE</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <version>5.3.1</version>
            <scope>test</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>2.2.7.RELEASE</version>
            <type>plugin</type>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>2.2.7.RELEASE</version>
            </plugin>
        </plugins>
    </build>

```

Screen de site web:

- User



Would you like the password manager to save the password for "localhost:8085"?

Save Never user1

Navbar Home Link

Liste des Villes

ID	Name	Cinemas
1	Casablanca	Show more
2	Marrakech	Show more
3	Rabat	Show more
4	Tanger	Show more
5	bensliman	Show more

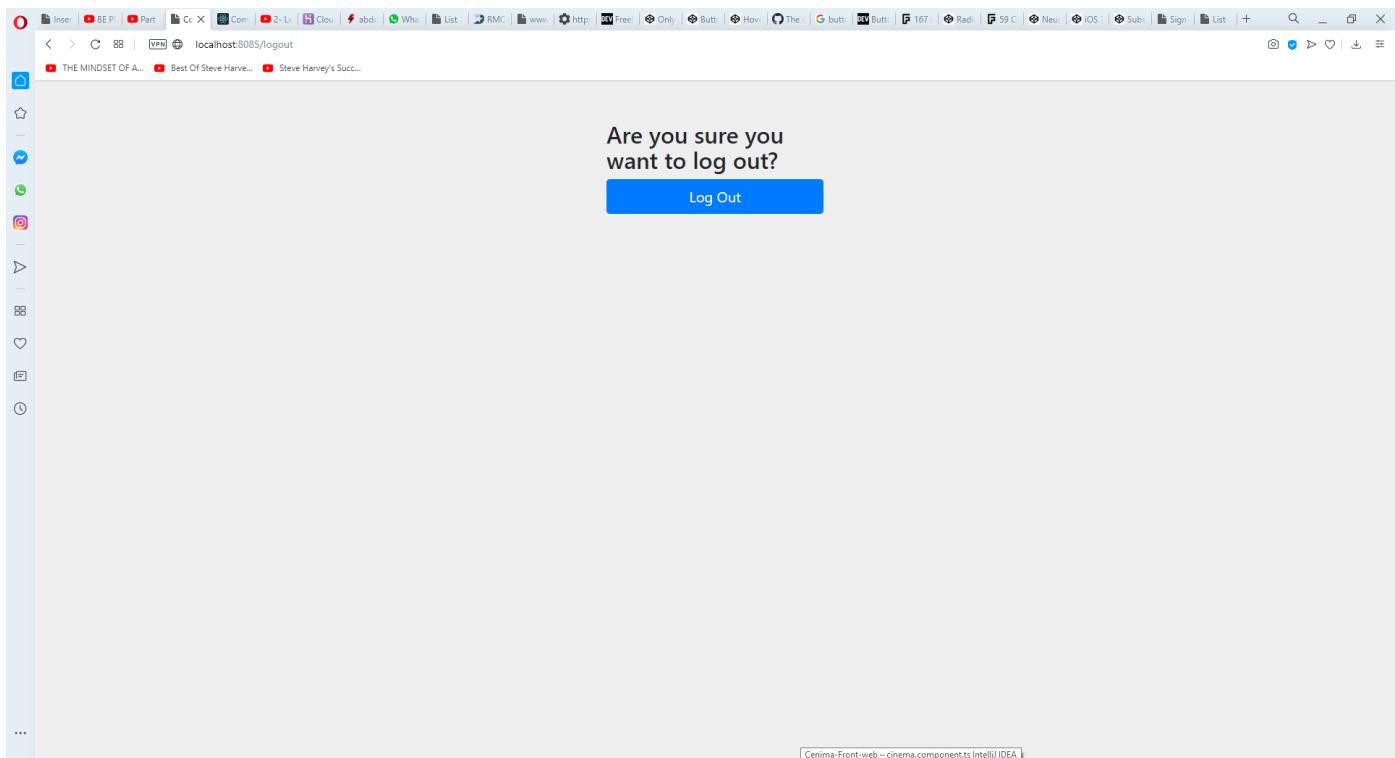
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Cinema-Front-web - cinema.component.ts IntelliJ IDEA

Liste des Cinemas

ID	Name	Nombre salles	Ville
1	MegaRama	6	Casablanca
2	IMAX	8	Casablanca
3	FOOUNOUN	5	Casablanca
4	CHAHRAZAD	5	Casablanca
5	DAOULIZ	7	Casablanca

0 1 2 3 4 5 6 7 8 9 10 11



● Admin

The screenshot displays two vertically stacked windows from a web browser, likely Chrome, showing an admin interface for a movie database.

Top Window: This window shows a "Please sign in" page. It features a "Username" input field, a "Password" input field, and a blue "Sign in" button. The URL in the address bar is `localhost:8085/login`.

Bottom Window: This window shows an "Add film" form. The form includes fields for "Titre" (Title), "Description", and "Durée" (Duration) with a value of "0.0". A green "save" button is at the bottom. The URL in the address bar is `localhost:8085/form`. The top navigation bar of this window includes "Navbar", "Home", "Link", "Modification", and a user dropdown set to "admin".

A status bar at the bottom of the bottom window indicates: "Cinema-Front-web - cinema.component.ts intelliJ IDEA".

Add ville

Id: 1

Name:

save

Modification success

Screenshot of a web application showing a list of cities. The page title is "Liste des Villes".

The table has columns: ID, Name, and Cinemas.

Data:

ID	Name	Cinemas
1	Casablanca	Show more
2	Marrakech	Show more
3	Rabat	Show more
4	Tanger	Show more
5	bensliman	Show more

Pagination: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

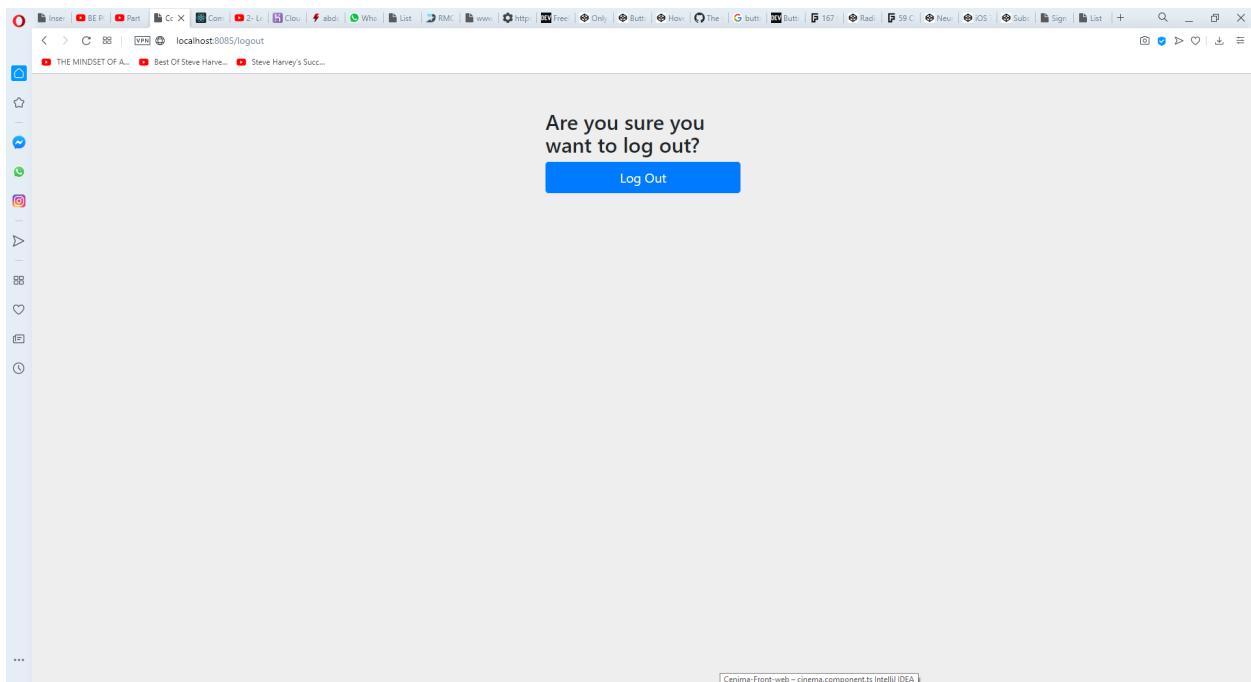
Screenshot of a web application showing a list of cities. The page title is "Liste des Villes".

The table has columns: ID, Name, and Cinemas.

Data:

ID	Name	Cinemas
1	Casablanca	Show more
2	Marrakech	Show more
3	Rabat	Show more
4	Tanger	Show more
5	bensliman	Show more

Pagination: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16



Cinema-Front-web - cinema.component.ts IntelliJ IDEA