

GESTION DES CINEMAS

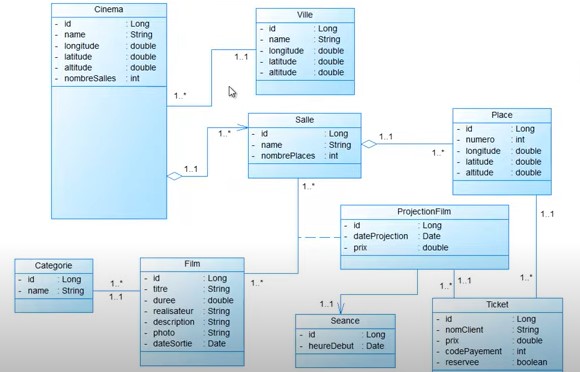
Réalisé par :

ISSALE Oumaima & Knar Imane

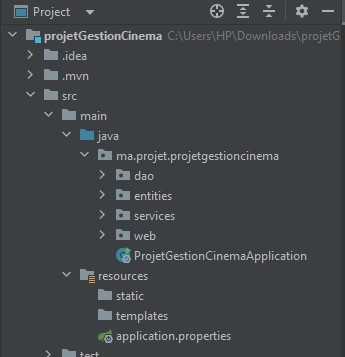
Objectif :

Le but de ce projet est de développer une application pour la gestion des cinémas en utilisant le framework spring (partie Backend) et angular (partie Frontend).

Conception de l’application:

La figure ci-dessous représente la modélisation de notre application sous forme d’un diagramme de classe.

Backend :

 Architecture du projet Backend :

On a divisé le projet en 3 packages :

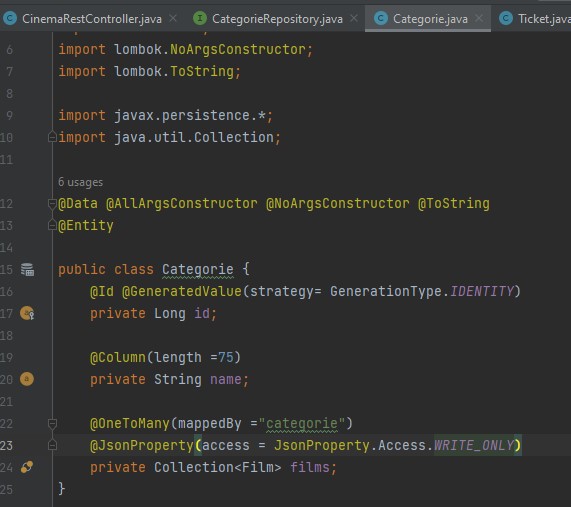
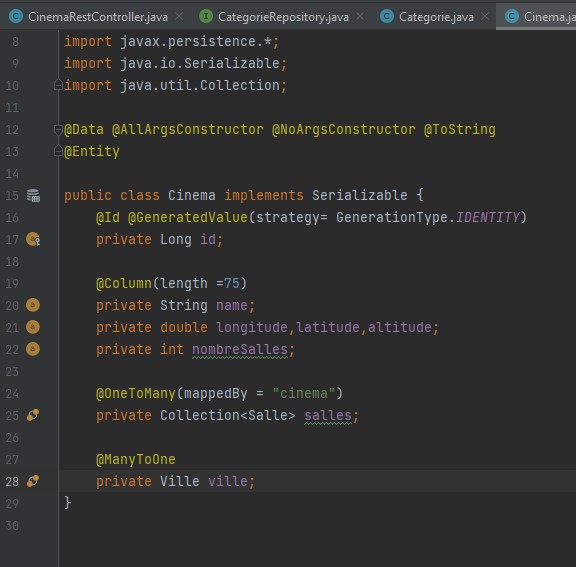
entities : qui représente nos objets qui seront convertis en entités JPA grâce au mapping.

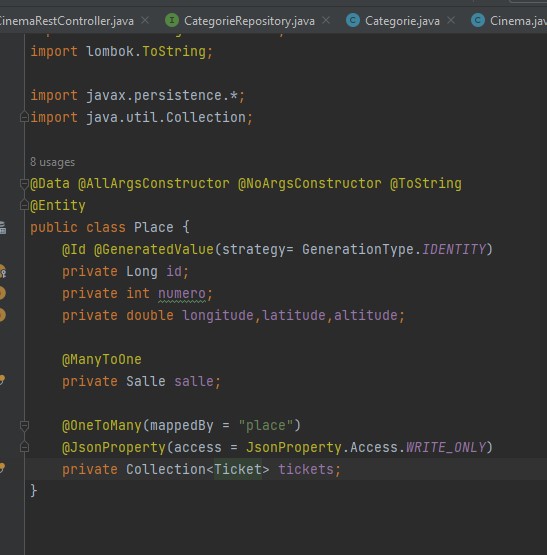
dao : qui contient l’ensemble des interfaces JPA repository .

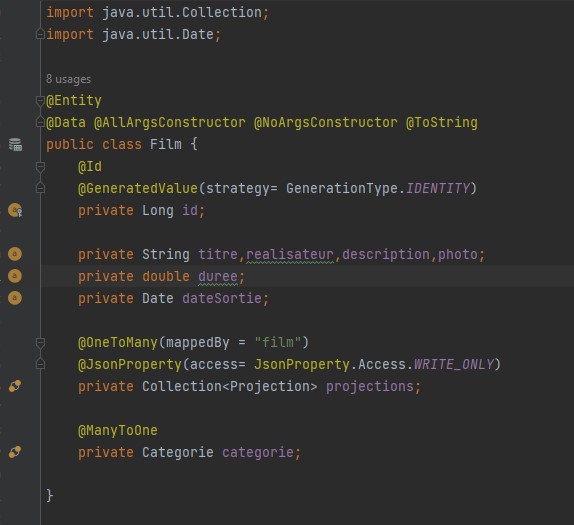
service :qui représente le métier de l’application.

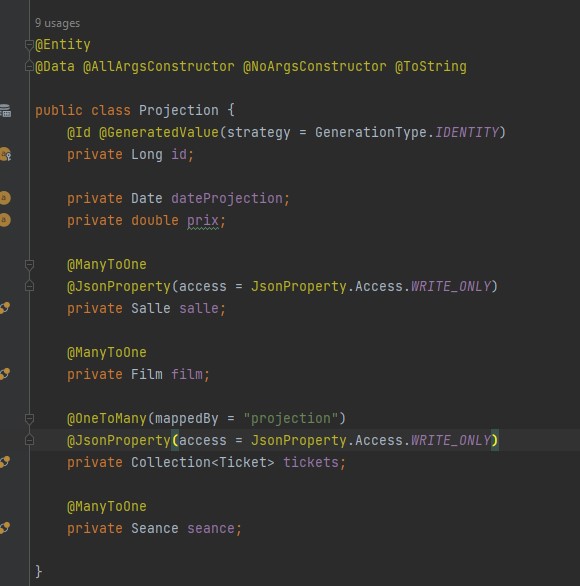
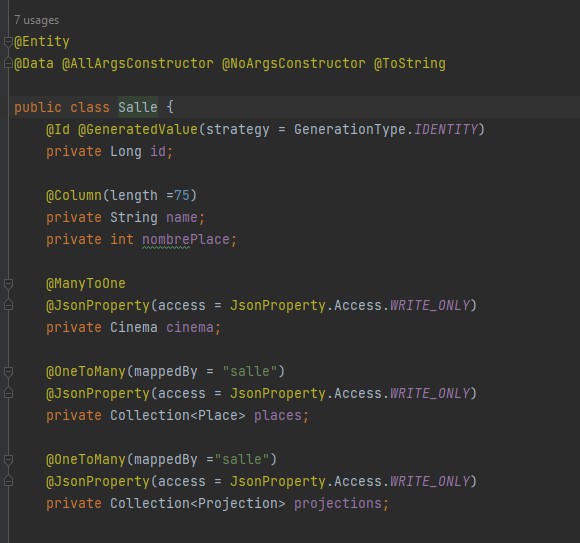
web : qui contient le Controller, responsable de l envoie et la réception des requêtes http.

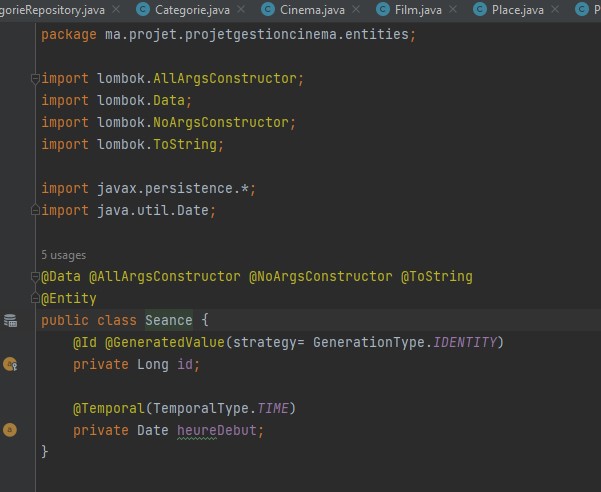
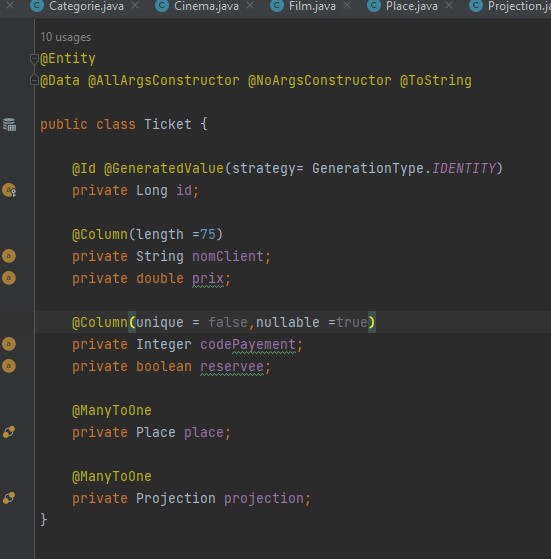
Entités JPA :

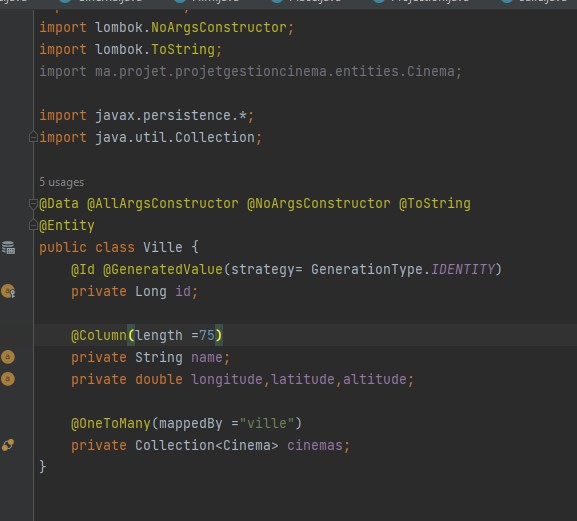






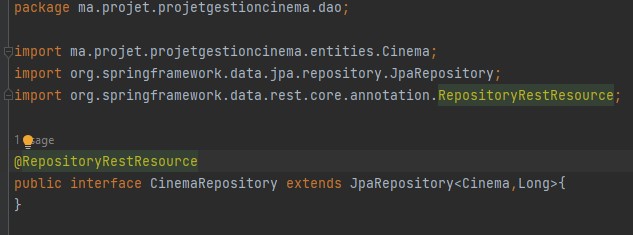
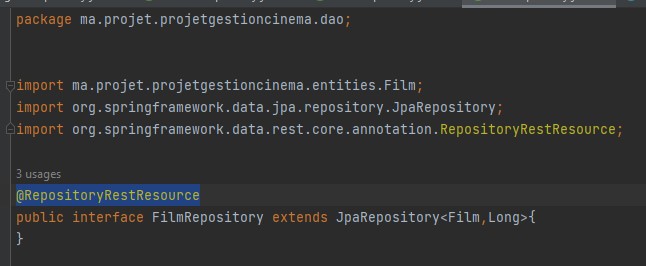


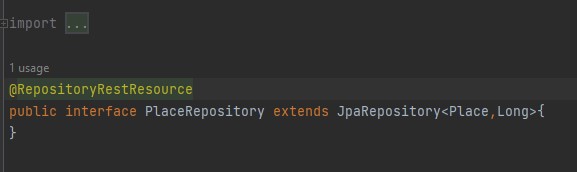
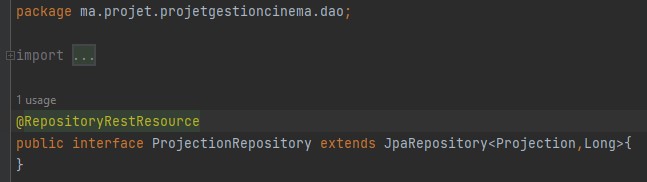


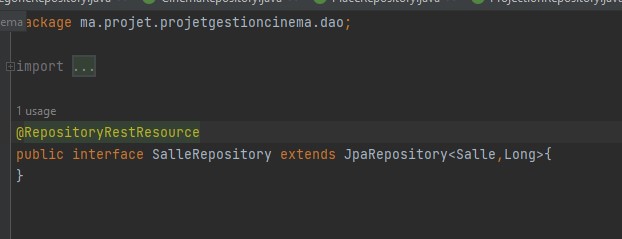
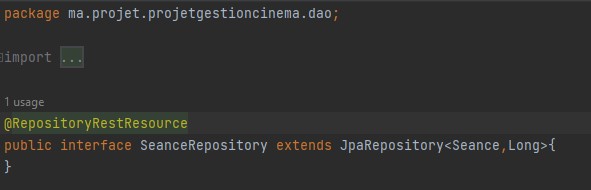


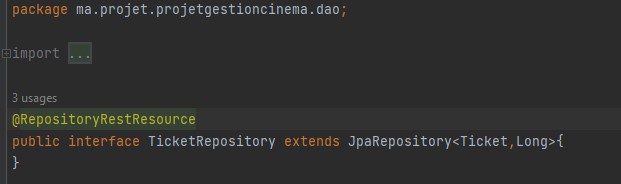
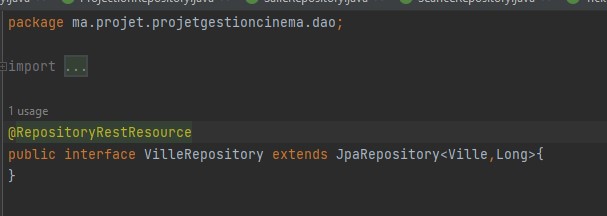
Les entités ci-dessus représentent notre diagramme de classe avec l’ensemble des associations représentées par @OneToMany , @OneToOne , @ManyToOne ou @ManyToMany.

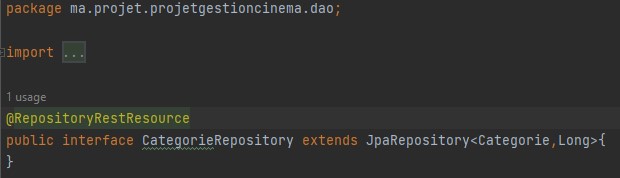
Les interfaces JPA :





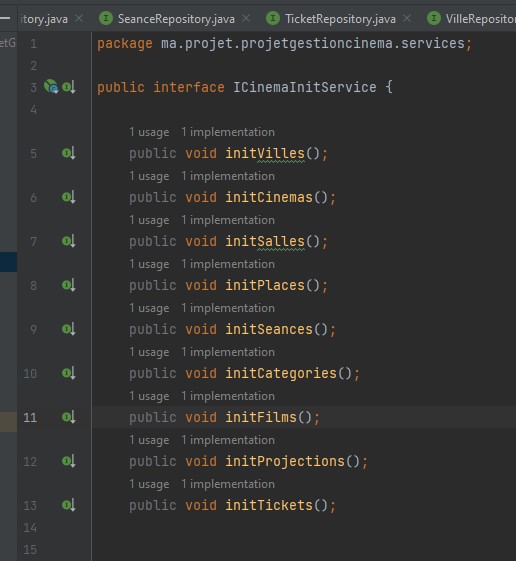




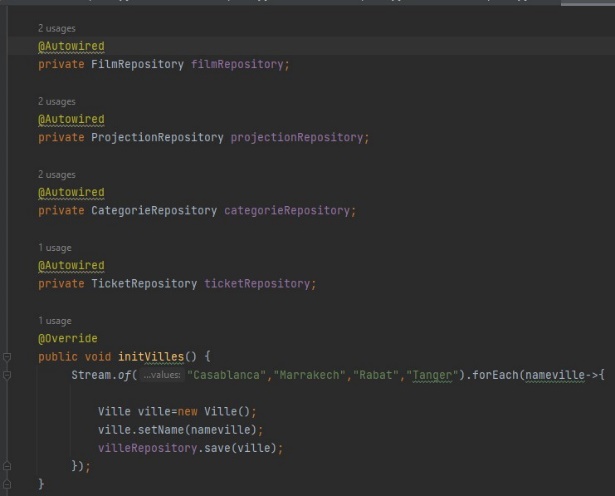


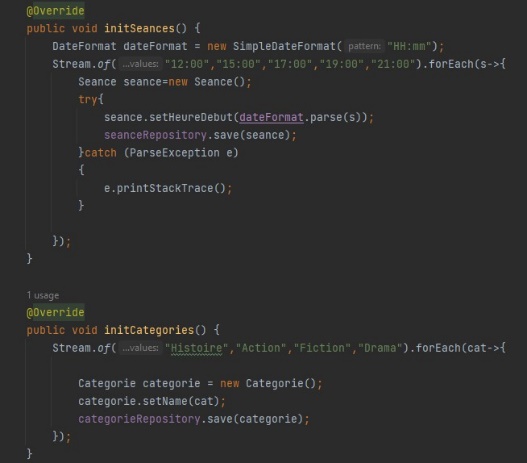
Ces interfaces héritent de l’interface JPA Repository ,qui genère toutes les methodes classique appliquées sur les entités

Si on veut personnaliser une méthode, il suffit de la définir à l’intérieur de l’interface.

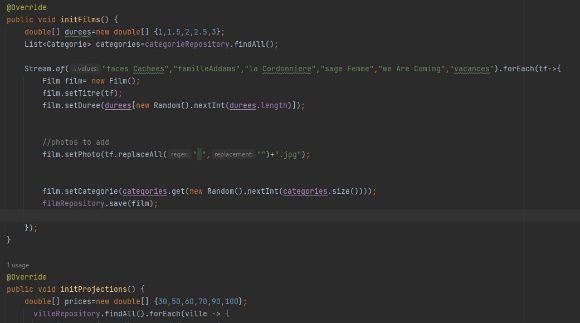
Code metier :

On définit dans cette interfaces toutes les méthodes à implémenter ,relatifs au traitement de l’application.







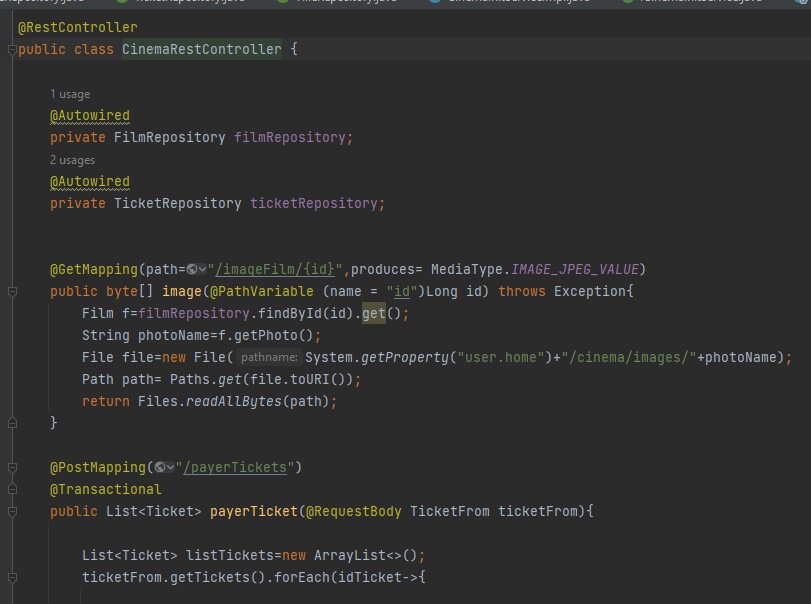




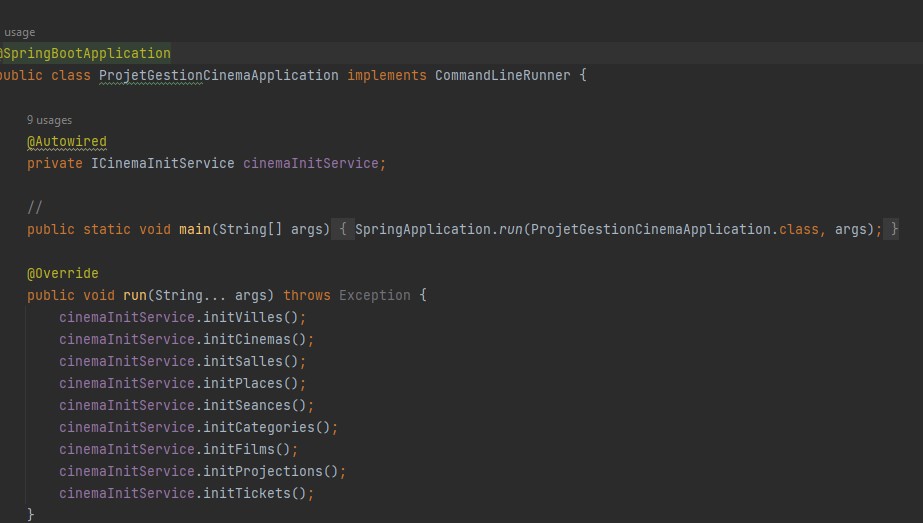
Cette classe CinemaService implémente l’interface ICinemaService et contient les implémentations des méthodes correspondant aux fonctionnalités de l’application.

On déclare les objets interfaces pour communiquer avec la partie dao.

Controller :

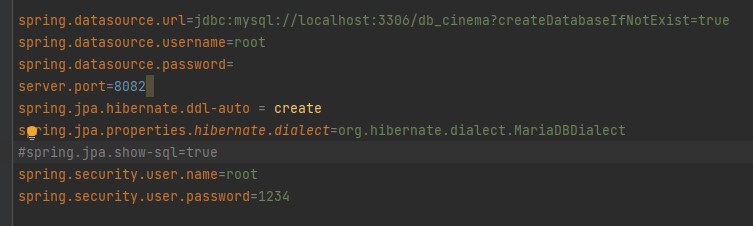


On en définit les méthodes correspondant à chaque type de requête envoyée.

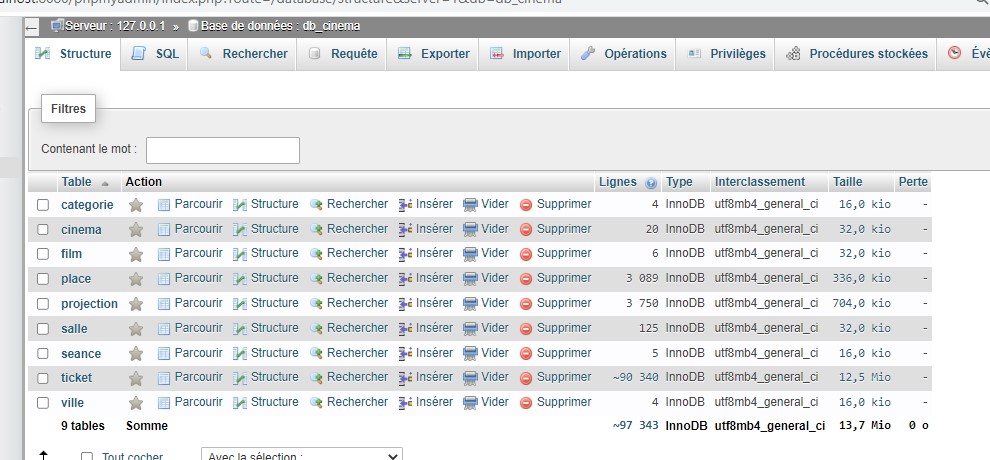
Application main :

Cette classe fait appel au service pour lancer le traitement.

MySQL :



Le fichier application.properties contient notre configuration de la BDD MySQL.

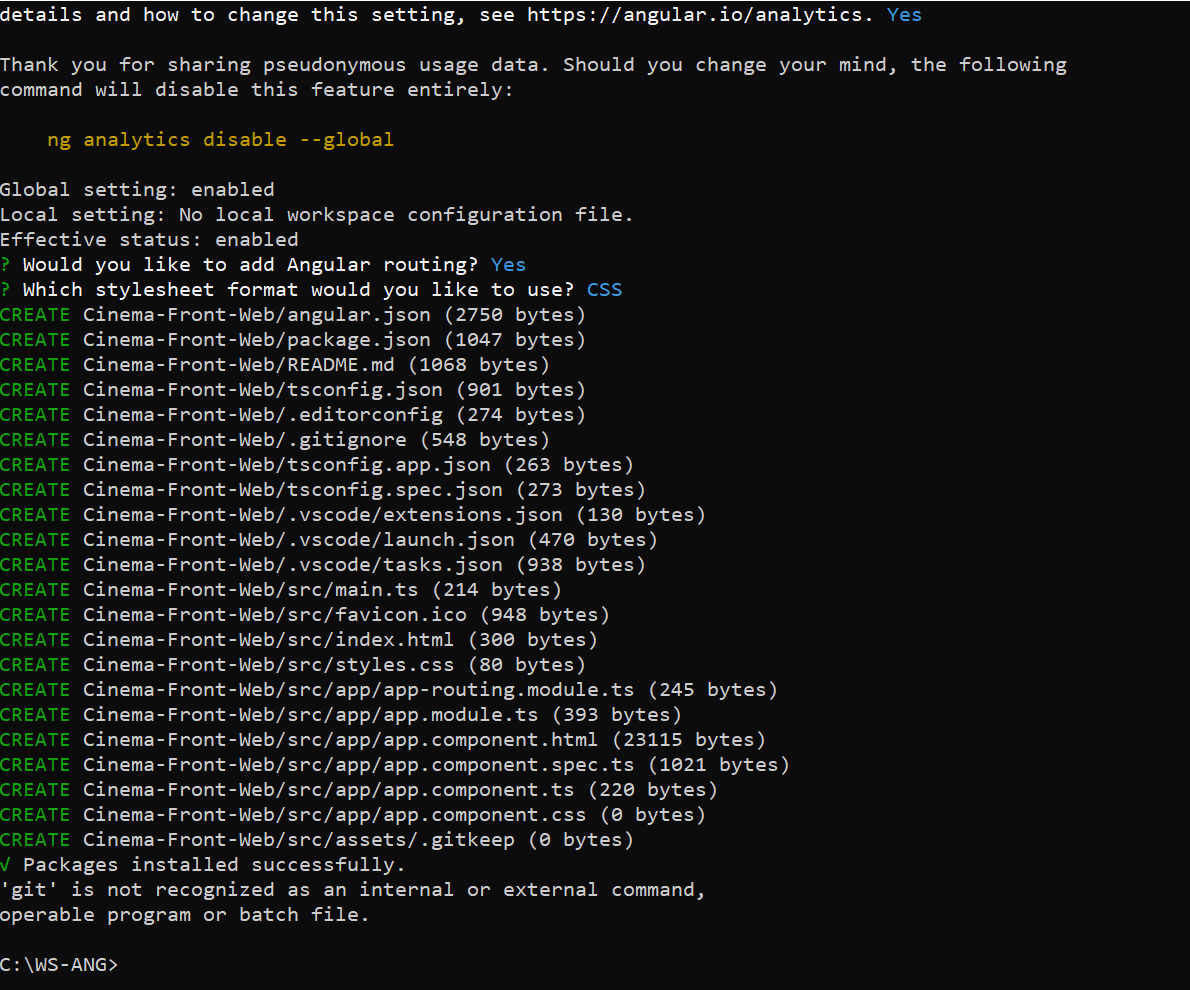


La base de donnée est créée et contient toutes les entités JPA déclarées.

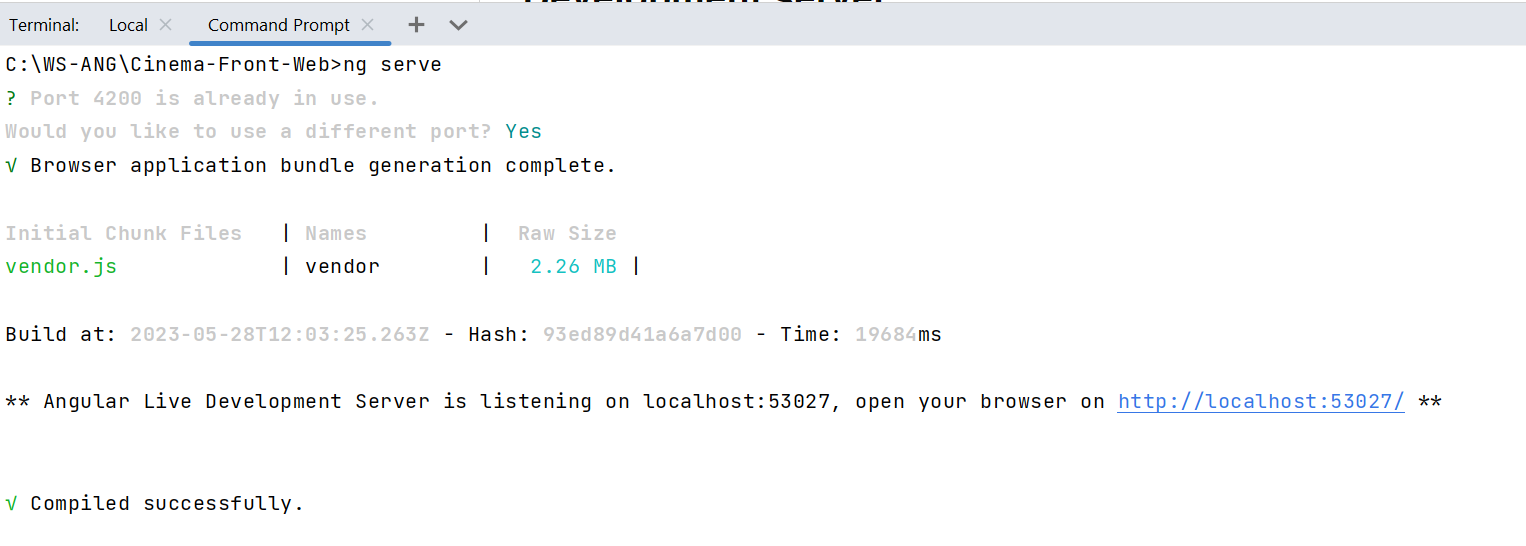
FrontEnd :

Pour la partie FrontEnd, on va d’abord installer Angular avec la commande : npm install -g @angular/cli

Après, on va créer un projet Angular : ng new Cinema-Front-Web

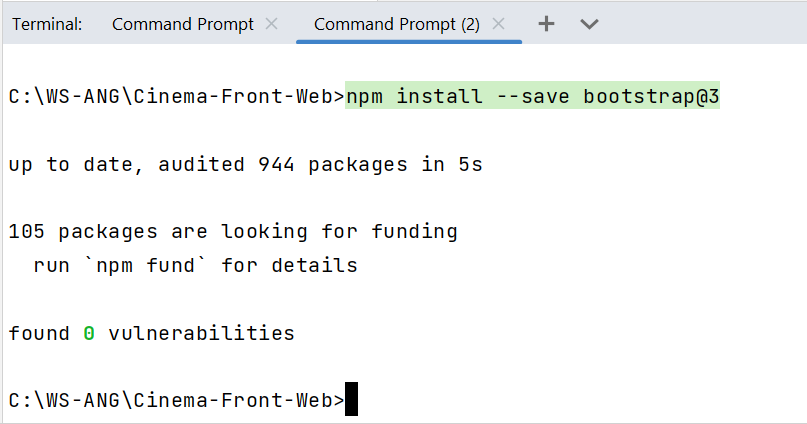


On va tester le projet avec : ng serve

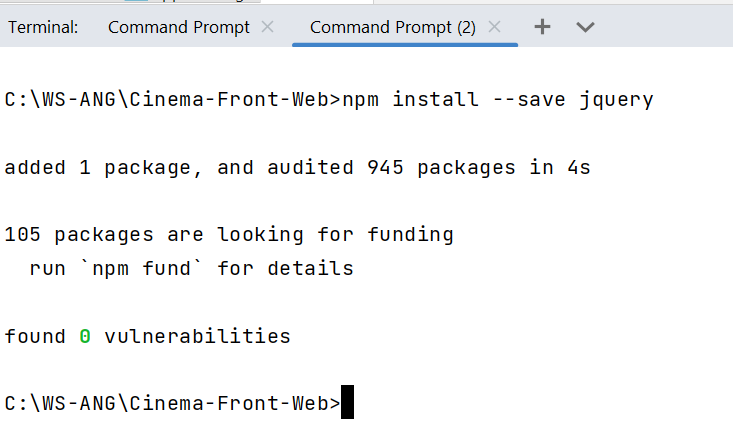


Pour utiliser la barre de navigation, on va d’abord installer Bootstrap et jquery

On va installer la version 3 de Bootstrap avec : npm install --save bootstrap@3



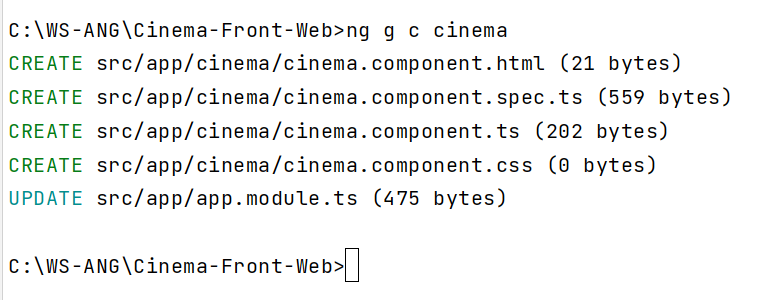
Et on va installer jquery avec : npm install --save jquery



Dans le fichier angular.json, on va ajouter les styles et scripts suivants :



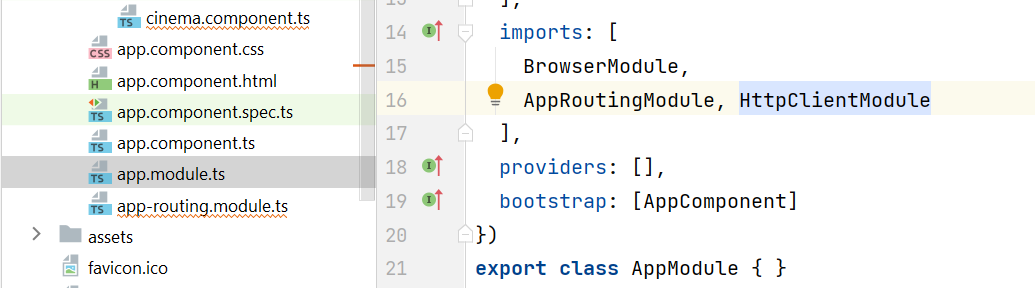
On va créer maintenant un composant web « cinema » qu’on va afficher :



On va commencer par afficher la liste des villes .

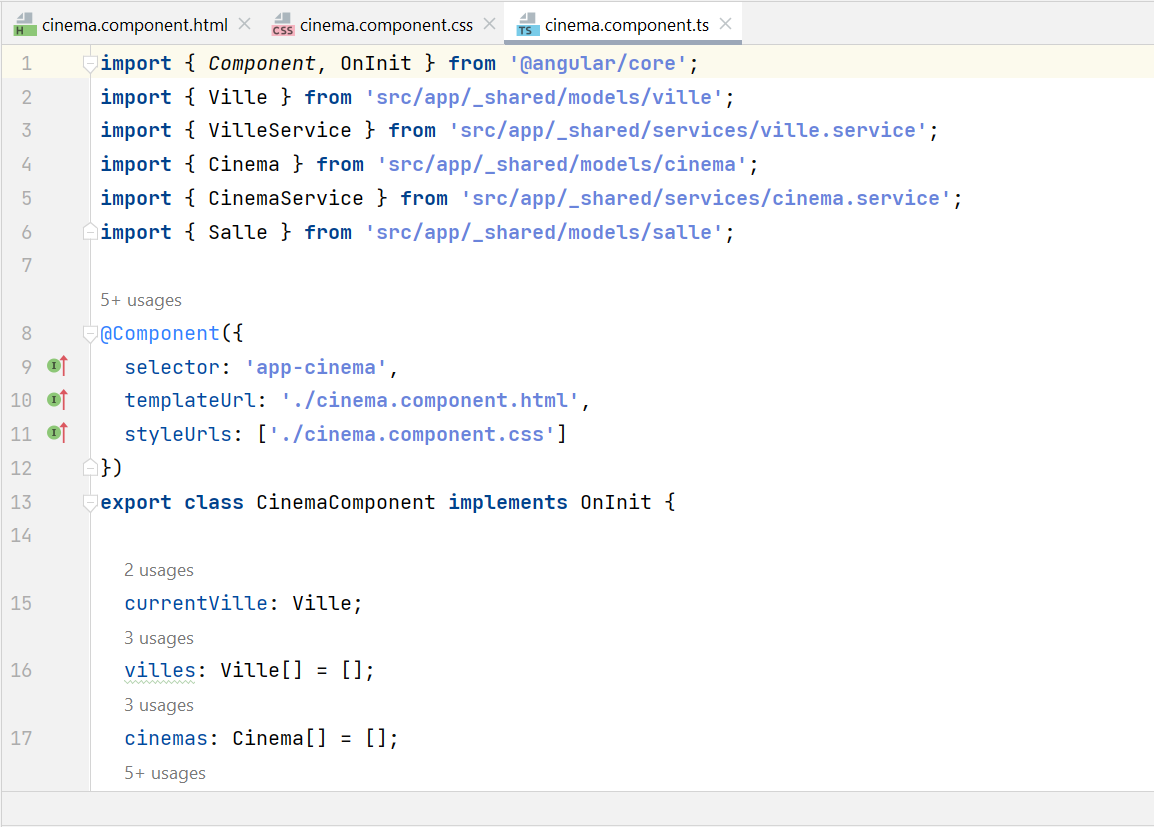
Dans cinema.components.ts , on va envoyer une requête http vers la partie Backend pour récupérer la liste des villes .

Alor on aura besoin d’utiliser le service HttpClientModule, on va l’importer :



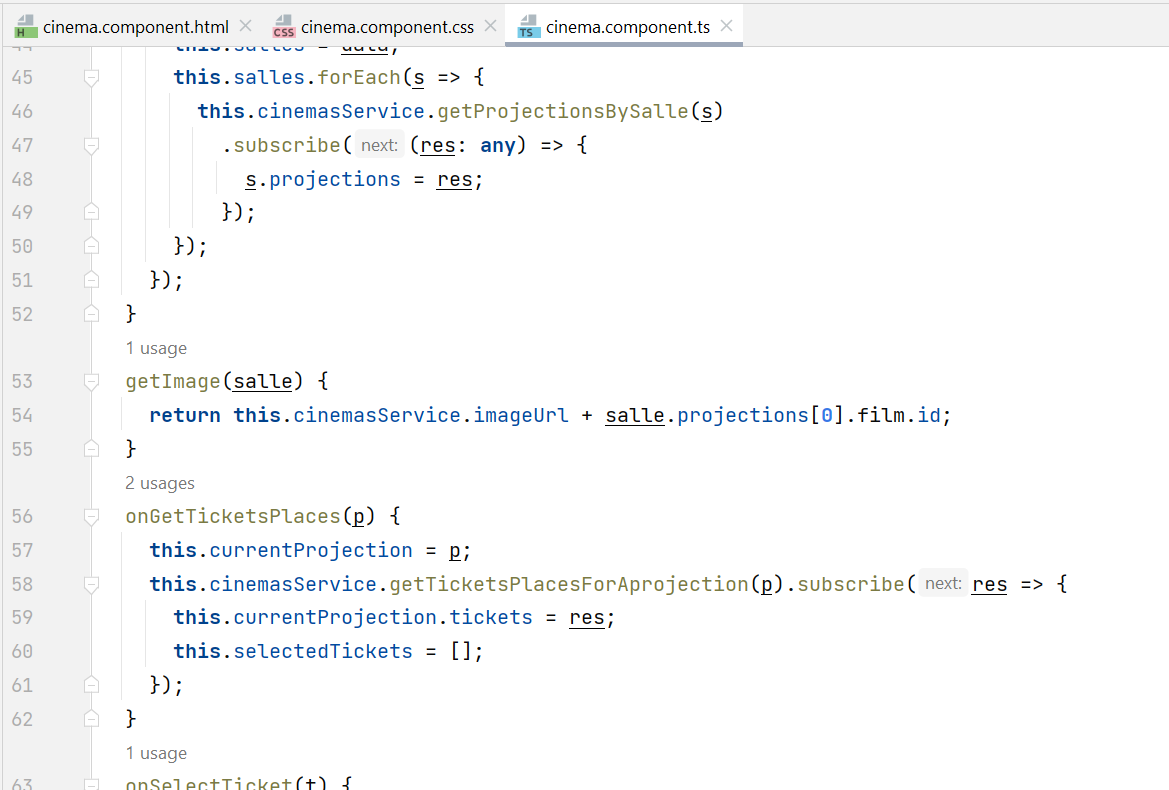
Dès que le composant CinemaComponent est chargé, il va exécuter la méthode ngOnInit() , on envoie une requête vers la parties Backend, on récupère la liste des villes, et on les stockes dans le tableau des villes.

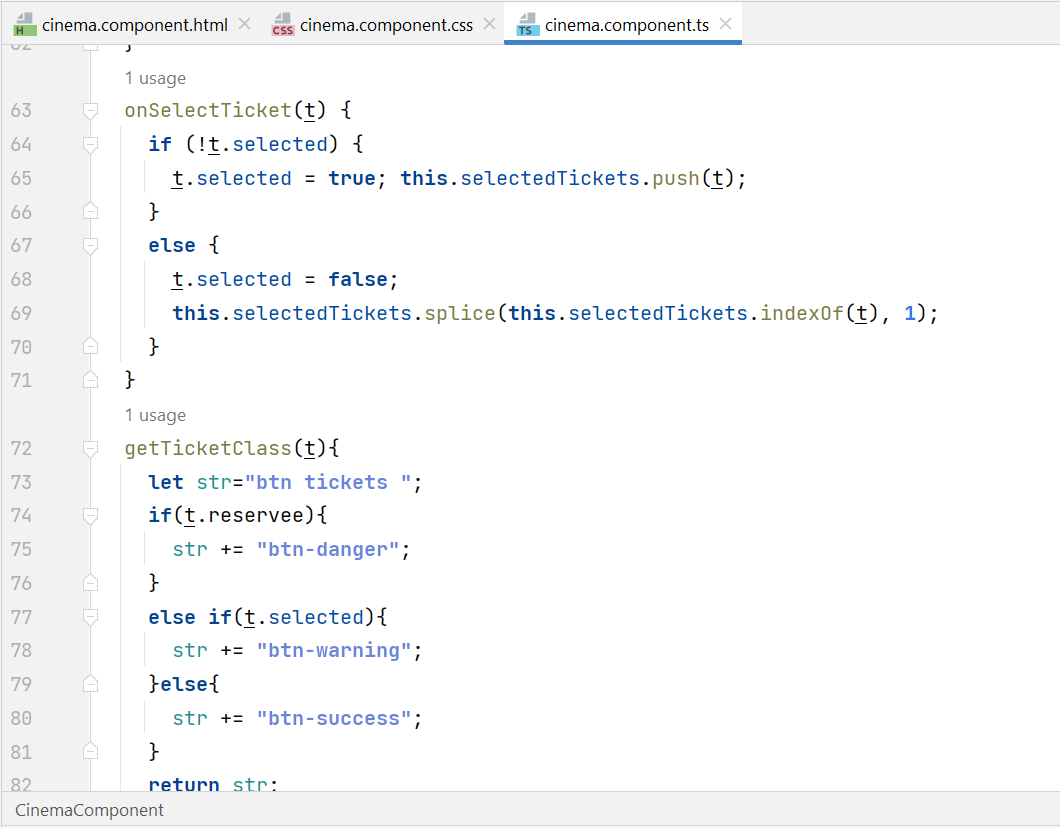
Dans le fichier cinema.component.ts on va mettre tout les services.

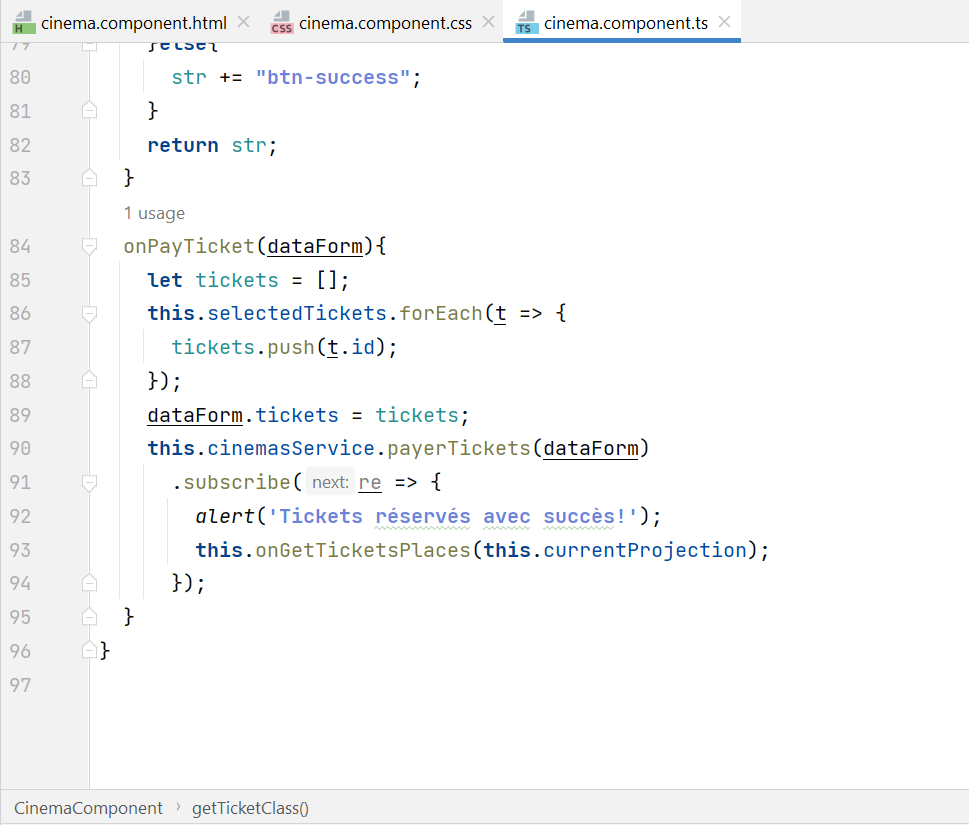












Pour pouvoir afficher la liste des villes, on va le faire dans cinema.component.html .

Dans ce fichier, on a fait une boucle sur les salles, pour chaque salle on a une boucle sur les projections.

Pour chaque projection on a des tickets qui contient le nom du client, prix du ticket, code de payement, le nombre de la place.

Quand on clique sur un ou plusieurs tickets, ils vont être en orange. Càd que tout les tickets sur lesquelles le client clique càd qu’il veut les acheter.

Après, le client va saisir le nom et le mode de payment.et il va cliquer sur le bouton payer tickets.

Une fois que le paiement est effectué, les tickets achetés vont être en gris.





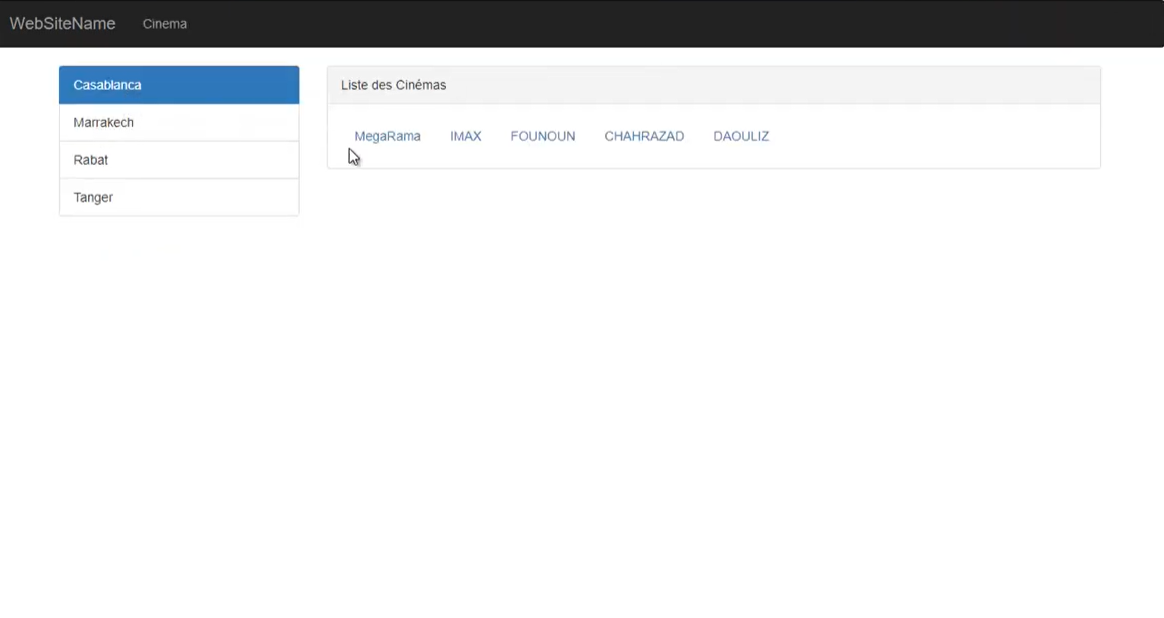


Pour fixer la tailles des boutons, on va faire des modifications au niveau de cinema.component.css

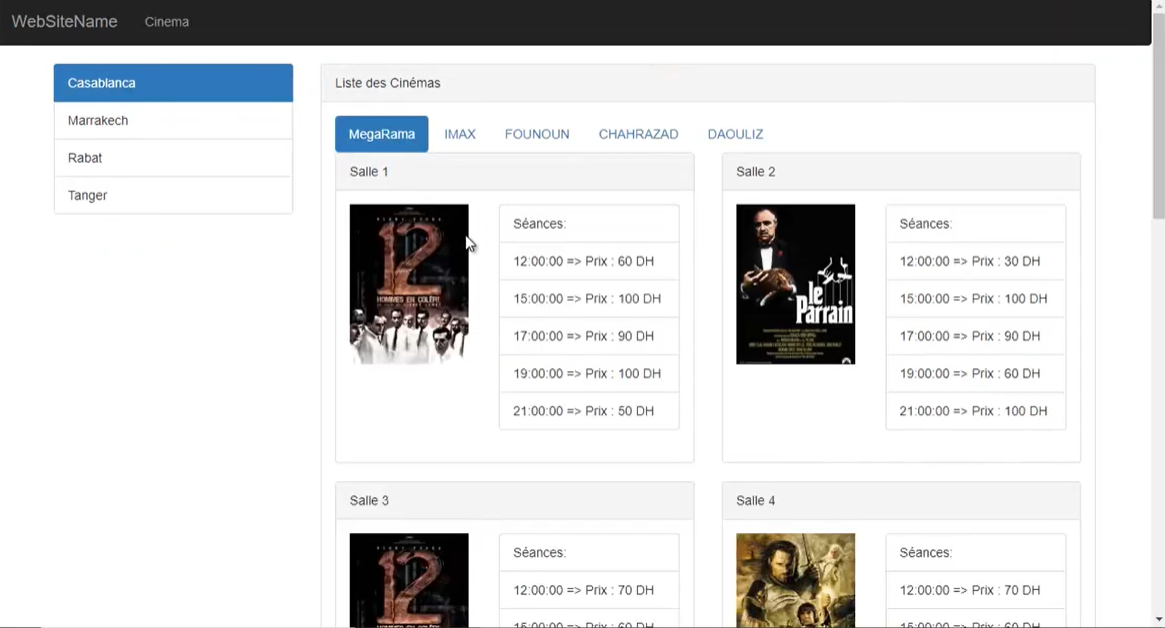


Voici le résultat :

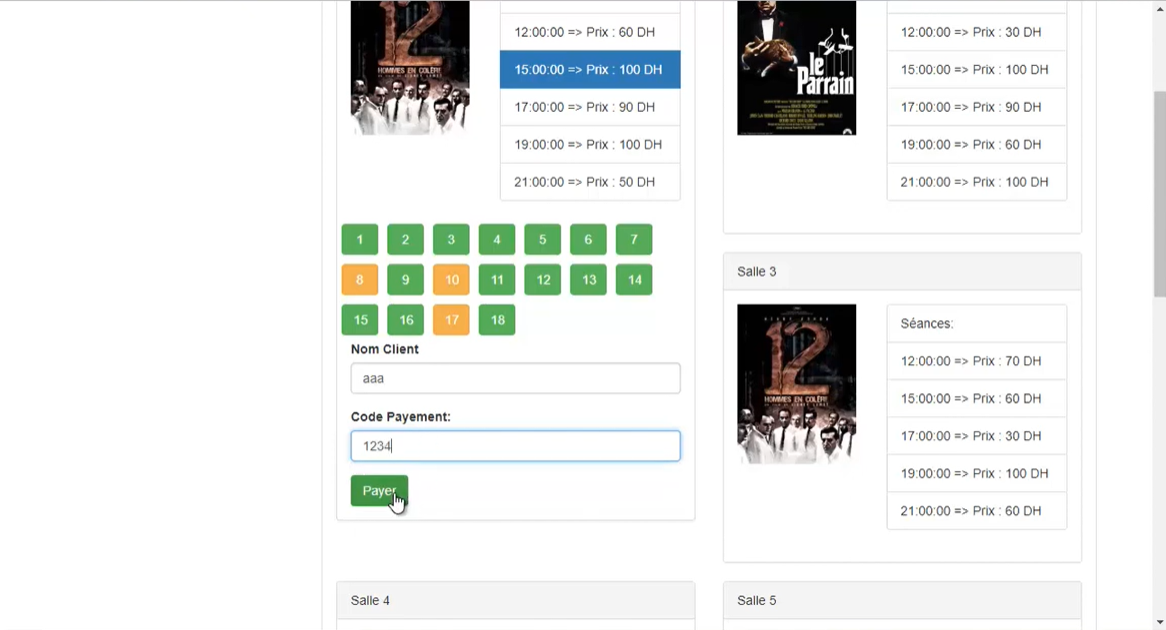
Quand on clique sur une ville, on peut choisir le cinéma qu’on veut.



Après qu’on a choisit le cinéma, on peut choisir le film .



On choisit la séances qu’on veut, et après on sélectionne le ou les tickets qu’on veut acheter , et on saisit le nom et code de payement.



Après on va avoir un message qui nous indique que les tickets sont réservés avec succès.

