Outils :

keycloak :24.0.1

postgres 15

Entity Framework

1. Architecture

* Keycloak et sa BD
* BD projet
* Api interconnectée avec Keycloak et la BD du projet
* Frontend qui consomme l’api

http://localhost:8080/admin

http://localhost:44349/swagger/index.html

FRONTEND

(React JS)

BD Keycloak

Keycloak

BD Projet

(postgres 15)

API

(.net)

1. Installation et configuration de keycloack

Il faut au préalable avoir installé docker sur sa machine (docker desktop dans mon cas) ;

Pour l’installation, j’ai choisi de le faire via docker, avec des images de keycloack 24.0.1 et postgres 15, les configurations nécessaires ont été spécifiées dans le fichier **docker-compose.yml** ;

url\_keycloak : <http://localhost:8080/admin>

username : admin

password : admin

Utilisateurs créés pour test

gdanielcedric : P@ssword2025! (Admin)

amazon : P@ssword2025! (Amazon)

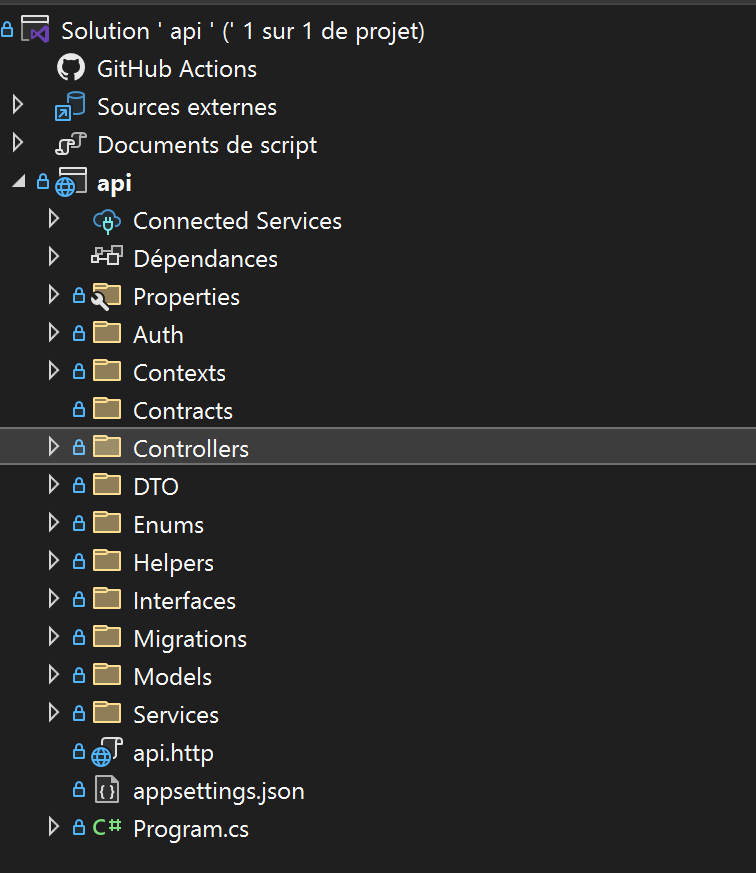
1. Configuration de la base de données

J’ai monté une image de postgres 15 dans docker, à laquelle je me suis connecté pour créer une base de données vide ;

Ensuite pour la migration de la base de données, j’ai utilisé **entity framework**, avec les commandes suivantes :

* **Dotnet ef migrations add « nom\_migration » (pour créer un dossier de migration)**
* **Dotnet ef database update (pour mettre à jour la base de données)**

1. API
2. Structure du projet



1. Auth

Contient une classe statique UserRoles qui défini les différents roles ;

1. Contexts

Contient **ApiDbContext: DbContext,** le context de la base de données du projet, avec les tables à implémenter, le schema à utiliser et les différentes clés ;

1. Controllers

Tous les contrôleurs du projet y sont, notamment le contrôleur « simulation » et « subscriptions »

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

1. DTO

Dossier des objets utilisés en paramètres des contrôleurs et autres services

1. Enums

2 fichier enums pour la liste des garanties et le type de souscriptions

1. Helpers

Contient un fichier Utility qui implémente les fonctions réutilisables

1. Interfaces

Le dossier des interfaces

1. Migrations

Le dossier créé après avoir effectué la commande : **Dotnet ef migrations add « nom\_migration »**

1. Models

Contient tous les modèles, classes associées aux tables de notre base de données

1. Services

Contient les services, qui implémentent les interfaces