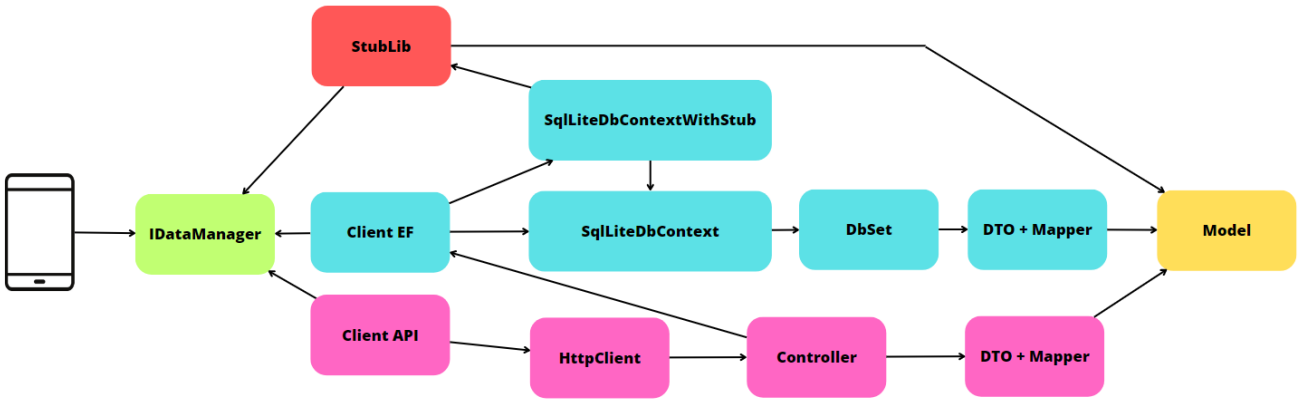
Voici un schéma de l'architecture de notre projet




Tout à gauche de notre schéma nous trouvons un téléphone qui représente l'application MAUI qui nous à été fourni. Tout à droit nous avons un modèle qui nous à été aussi fourni. Pour continuer avec les éléments qui nous ont été fourni nous avons un Stub représenté en rouge et un IDataManager qui lui est de couleur verte. Le IDataManager est une interface. Elle est le point d'entrée vers nos données pour notre application.

Notre client Entity Framework implémente l'interface IDataManager. Ce client est décomposé pour que chaque entité ait son manager. Notre client dépend d'un DbContext. Nous en avons donc créé deux, un SqlLiteDbContext utilisant SqlLite et un SqlLiteDbContextWithStub qui utilise SqlLite et permet d'entrer des données qui sont présentes dans le Stub. Ce DbContext hérite du premier.
Les DbContext que nous avons créé utilisent des DbSet. Ce sont des collections d'entités. Les entités servent à faire le lien entre le modèle et notre base de données. Nous avons dû aussi réaliser des mapper afin de pouvoir convertir les entités en classe de notre modèle et inversement.

Concernant notre client API, il implémente aussi l'interface IDataManager. Pour réaliser les requêtes vers notre API nous utilisons un HttpClient. Les requêtes de notre API sont liées à des controllers. Ce sont eux qui manipulent les données. Ces données sont aussi des DTO afin de ne pas modifier le modèle. Nous avons donc dû aussi réaliser des mapper pour la conversion.

L'utilisation de l'application MAUI peut ainsi être faite par le biais d'un stub, d'une WebAPI ou d'une base de données Entity Framework. Ce changement peut être fait très facilement.