Spécifications techniques

Projet Cellarium

Février 2026

Axel Vincent Jaunet

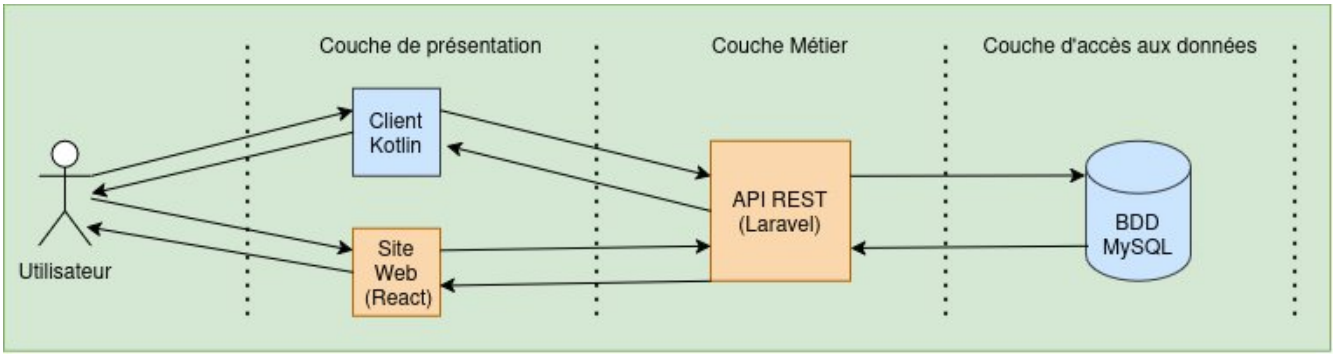
Sommaire

Objectif du document

Ce document constitue le dossier de conception technique du projet Cellarium. Les objectifs du document sont d'apporter plus de détails techniques. Ce document fait suite au cahier des charges et aux spécifications fonctionnelles.

Architecture technique

Pour pouvoir mettre en place une application mobile et une application web, il faudra utiliser l'architecture n-tiers. Cellarium est une petite application, donc il n'y a qu'une seule base de données relationnelle avec MySQL.



Application Web et Mobile

Back-End

Le Back-end est la partie invisible du site en charge du coté logique de l'application mobile. Celui ci sera codé en Php avec le framework Laravel. Il sera en lien direct avec la base de donnée relationnelle

Front-End

Le Front-End est la partie visible du site web. Elle comprend le Front-office et le Back-office. Le framework React sera utilisé, via le package Inertia de Laravel.

Application Mobile

L'application mobile sera développée pour les téléphones Android, au moyen du Framework Kotlin et Jetpack Compose, tous les deux conçues par Google, assurant de meilleures performances dans l'environnent Android.

Base de données

Le système de gestion de bases de données choisi est MySQL. C'est un outil de gestion de bases de données externes à l'API. Elle va contenir les informations nécessaires à toutes les fonctionnalités de l'application, comme les items, utilisateurs, contenants, sources, ...

Dictionnaire des données

Table User		
Nom	**Type**	**Taille**
Id_user	INT	11
firstname	VARCHAR	50
Lastname	VARCHAR	50
Email	VARCHAR	50
Matricule	INT	11
Accept_rgpd	Boolean	
isAdmin	Boolean	
isAdminChief	Boolean	
Password	VARCHAR	50

Table Item		
Nom	**Type**	**Taille**
Id_item	INT	11
name	VARCHAR	50
Total_qty	INT	11
State	BOOLEAN	
Is_stock	BOOLEAN	
Seuil	INT	
Sortorder	INT	

Table Caserne		
Nom	**Type**	**Taille**
Id_caserne	INT	11
city	VARCHAR	50
Postal_code	VARCHAR	5
Code	INT	11

Table Contenant		
Nom	**Type**	**Taille**
Id_contenant	INT	11
name	VARCHAR	50

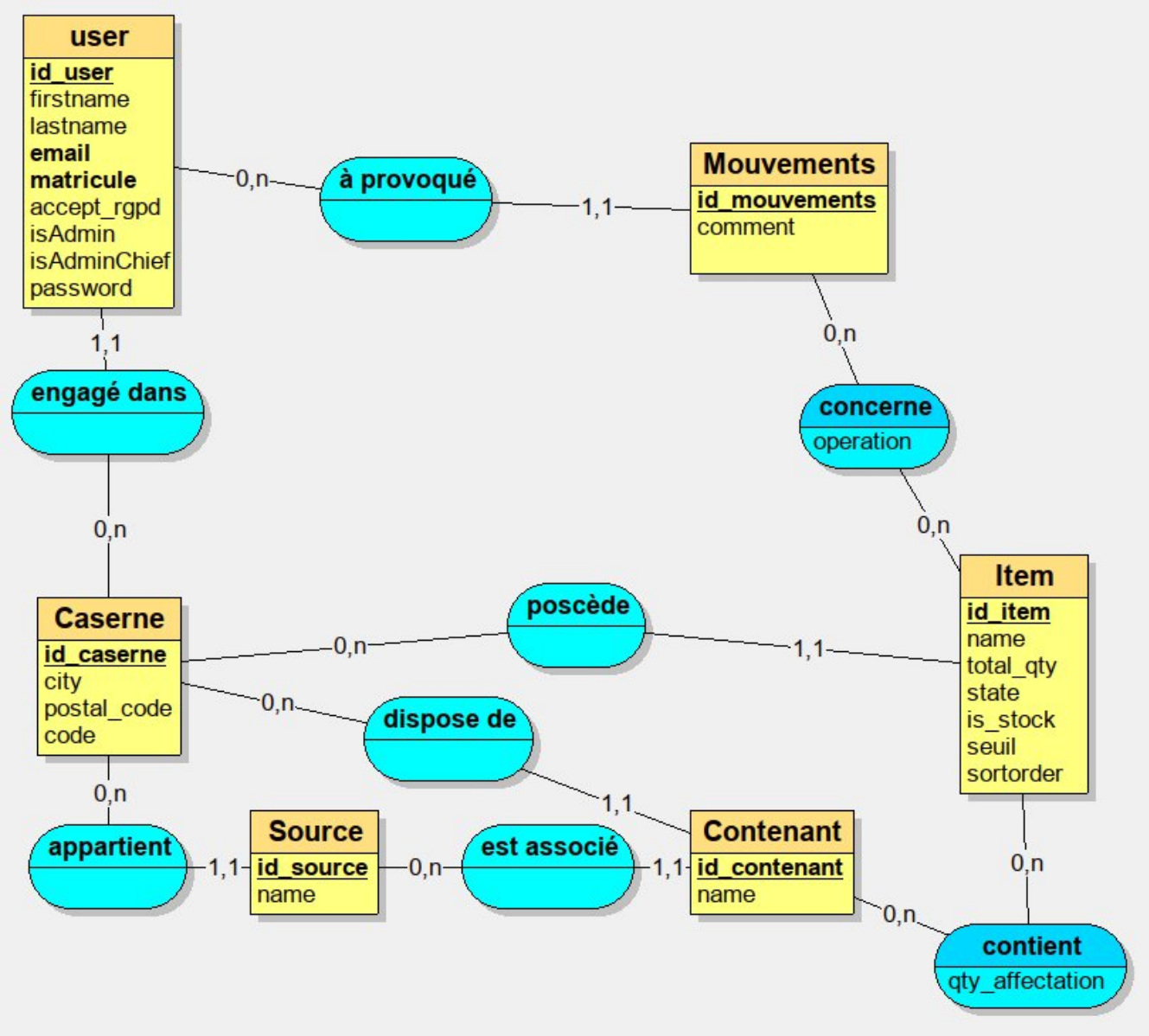
Table Source		
Nom	**Type**	**Taille**
Id_source	INT	11
name	VARCHAR	50

Table Mouvements		
Nom	**Type**	**Taille**
Id_mouvements	INT	11
comment	VARCHAR	50

Modélisation de la base de données

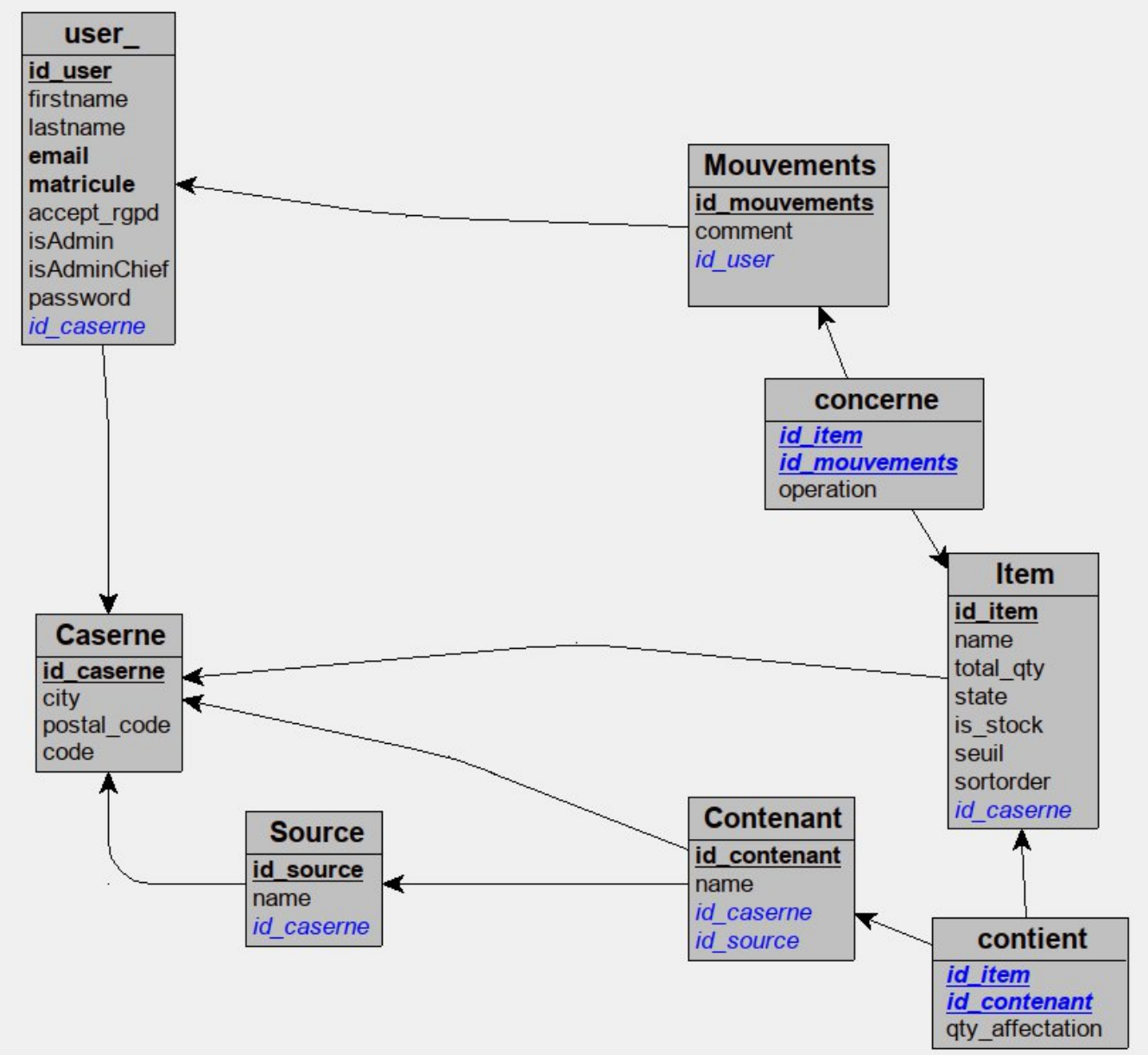
Le MCD

Le modèle conceptuel de base de données représente les données issues du dictionnaire, sous forme d'entité en ajoutant des détails sur les relations qui existent entre elles.



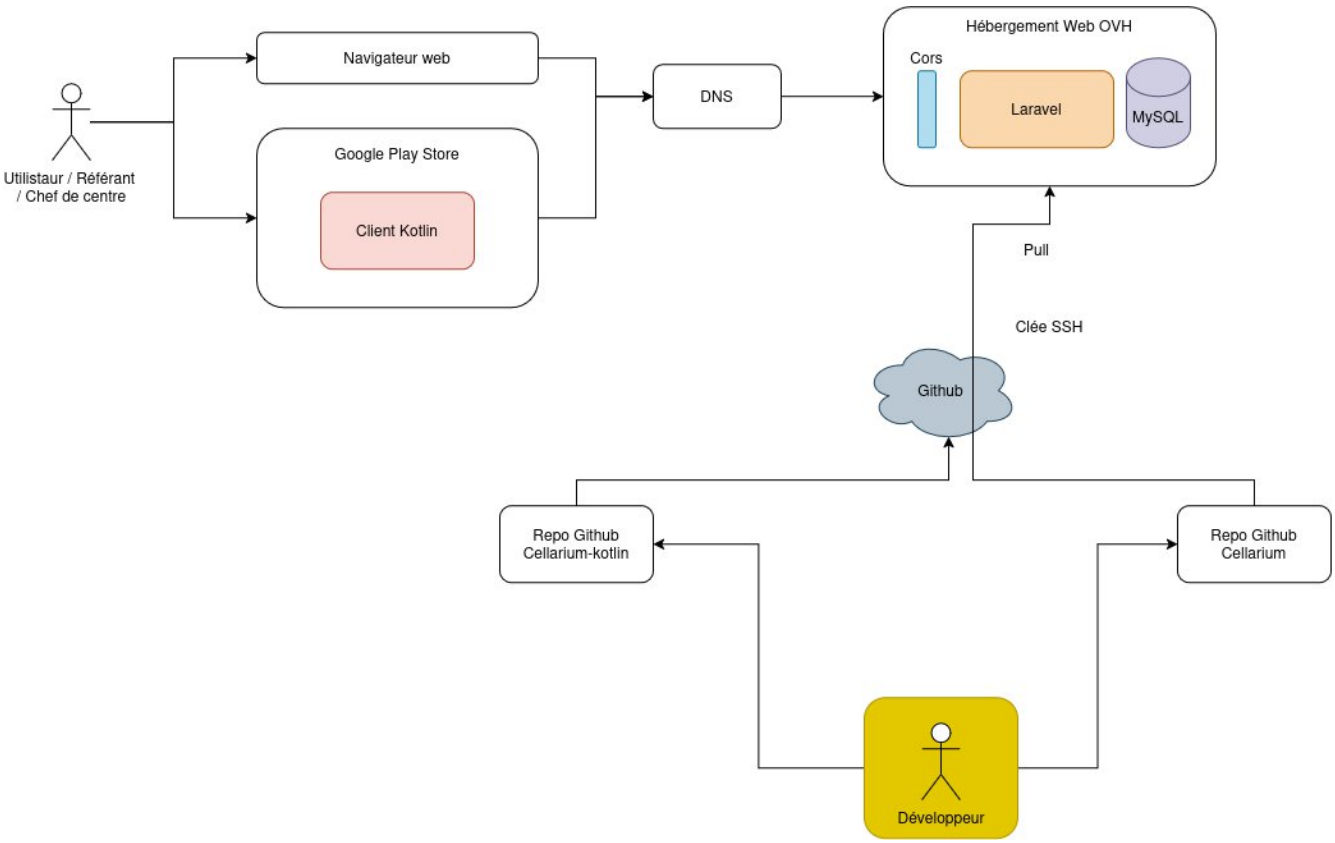
Le MLD

Le modèle logique des données est issu du MCD. Cependant, ce n'est plus des entités, mais des tables. Dedans sont représenter les clés primaires et étrangères



Architecture de déploiement

Voici l'architecture de déploiement de Sentinelle. Elle explique la disposition des composants du système sur les infrastructures physiques et les outils utilisés pour la mise en production.



Serveur Web

Pour le déploiement, j'utiliserai l'hébergeur infinityfree pour héberger l'API Laravel. En plus de cela, j'ajouterai le nom de domaine « pharmacie-sp-saintmartin.kesug.com », ainsi que le certificat SSL ; le coût annuel est de 0.00€ TTC.

Architecture Logicielle

Les différentes couches seront réalisées dans des dossiers distincts. La partie Api sera intitulée « app/Http/Controllers/Api », la partie couche applicative pour l'application Android s'intitulera « App ».

Les deux seront dans un dossier nommé « Cellarium ».

Interface de programmation application web

Les sources et les versions du projet sont gérées par Git. L'application web est construite selon le pattern MVC avec le framework Laravel. La couche modèle donne accès aux données dans la couche de données. Les contrôleurs comprendront toutes les méthodes qui permettent de réaliser les différentes fonctions du CRUD. Les routes serviront à associer des méthodes à des urls.

Couche de présentation

Les sources et les versions du projet sont également générées par Git. J'utiliserai l'architecture par défaut générée par Laravel. Ces fichiers seront faits en PHP, SCSS et JSX.

Sécurité et Authentification

L'authentification des utilisateurs se fera avec l'API. Pour ce projet, j'utiliserai un token JWT. Le client recevra un token JWT généré par le SDK Kotlin et devra le transmettre à l'API, qui l'interrogera pour vérifier la validité de ce dernier.

Les mots de passes seront hachés dans la base de données.

Informations supplémentaires

Environnement

Pour mener à bien ce projet, l'environnement choisi est le suivant :

- L'IDE pour le client kotlin : Android Studio
- L'IDE pour l'API et le client web : PhpStorm
- Gestion des versions : Git relié à un compte Github
- Os en local : Ubuntu

Procédure de livraison

Pour livrer le projet, l'API et l'application web seront déployées par un GitHub Action sur le serveur infinityFree. Le code source sera donc stocké dans un repository. Ainsi, lorsqu'un commit sera effectué et sera mergé sur la branche main, les modifications se déploieront automatiquement sur le serveur.

Le client Kotlin sera également stocké dans un repository, mais sera déployé manuellement sur le Google Play Store. En effet, il n'existe pas de pipeline automatique, car il est nécessaire de compiler le projet avec une signature numérique et de remplir un formulaire pour chaque mise à jours.