Planète Dashboard – Android

# Architecture

Le schéma suivant montre l’architecture globale de l’application mobile Planète Dashboard.

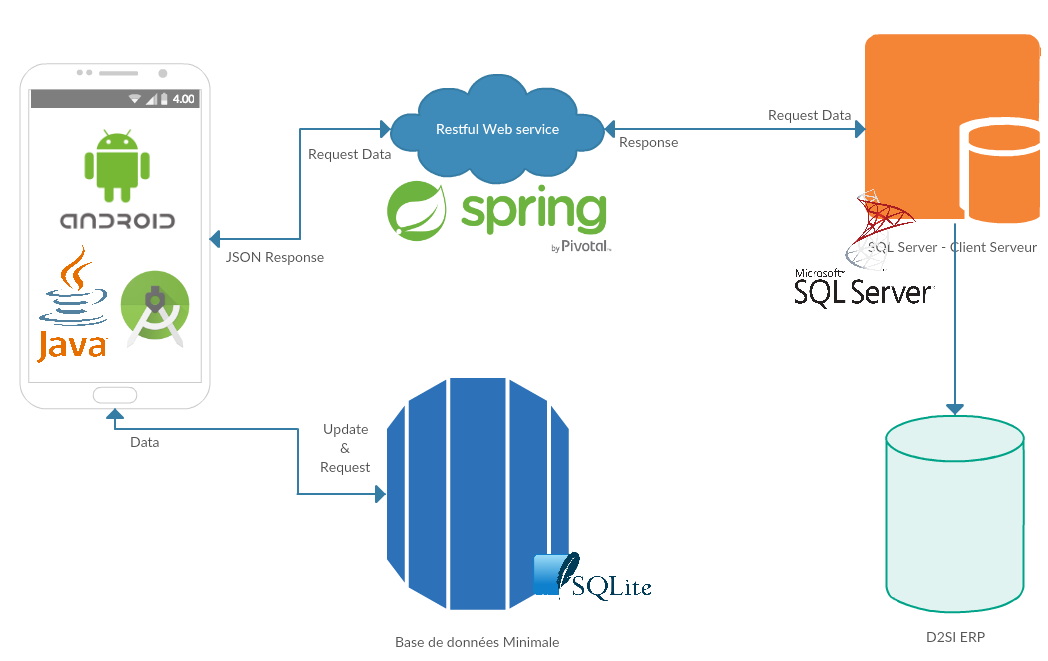


Figure - Architecture fonctionnelle

Planète Dashboard est une application qui se répartie en plusieurs parties :

* Partie Applicative : elle se lance au niveau de smartphone (ayant comme Système d’exploitation Android), permettant de représenter les indicateurs sous forme d’un Dashboard avec des graphes explicites.
* Base de données minimale : portable avec la partie applicative, représentant les données minimales qui permet de lancer l’application. Cette dernière se met à jour régulièrement. Cette dernière est développé avec l’outil Realm une base de données très puissante dédiée pour mobile basée sur SQLite.
* Serveur de synchronisation : Il permet de lier l’application se trouvant sur smartphone avec l’ERP et les bases de données des clients d’une manière efficace. Ainsi, il permet d’unifier l’accès aux données en permettant à n’importe quel utilisateur mobile d’accéder à une base de données spécifiques tout en respectant les contraintes de sécurité. Ce serveur est basé sur Spring une technologie fondée sur une architecture RestFull. Le langage de programmation utilisé à ce niveau est Java.

## Serveur de synchronisation

Le serveur de synchronisation permet concrètement de connecter l’application mobile à l’ERP. Ainsi, il permet de :

* Recevoir des requêtes avec des URL et des paramètres spécifiques.
* Répondre à des requêtes, le format de la réponse est un fichier JSON contenant les réponses, en cas ou la réponse est vide on retourne un tableau de données vide, en cas d’échec le serveur retourne la valeur NULL.
* Connecter à la base de données distante SQL Server et exécuter des requêtes.

### Environnement de développement

Le serveur de synchronisation ayant une architecture REST est développé sous JAVA en se basant sur le Framework Spring, une technologie qui prend le lead sur ce marché grâce à sa puissance, facilité et son module de sécurité.



Figure - Environnement de développement de Serveur de synchronisation

### Diagramme de classe

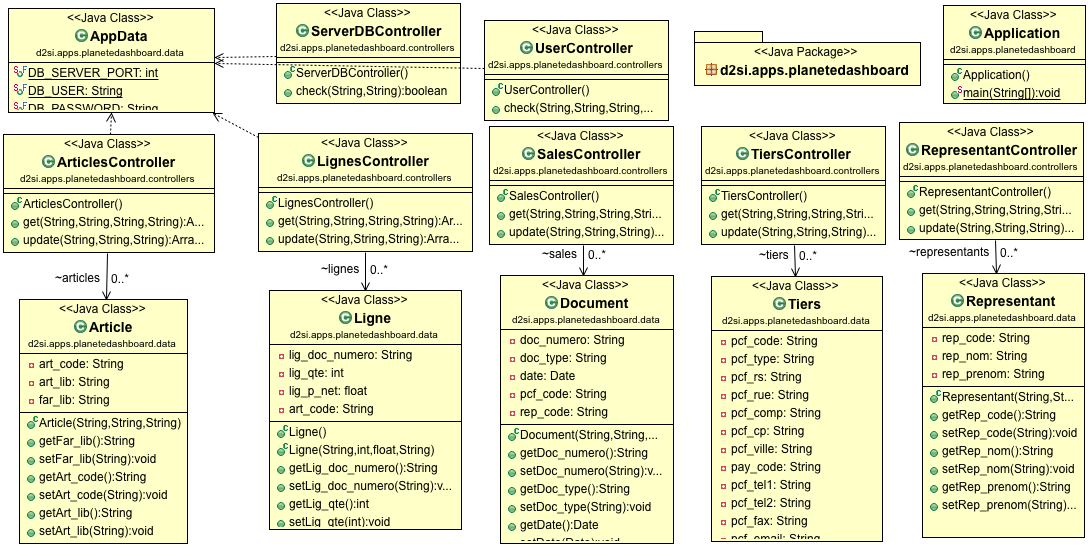
Spring permet via une architecture très simple de mettre en place un serveur recevant des requêtes GET et POST. L’ensemble des classes de serveur de synchronisation est représenté dans le diagramme suivant :

Figure - Diagramme de classe de serveur de synchronisation

L’architecture est simple :

La classe Application permet de lancer le serveur pour recevoir et répondre aux requêtes.

L’ensemble de contrôleurs permettant d’exécuter des requêtes spécifiques pour établir une connexion à la base de données distante et d’extraire des données de cette dernière qui communiquent avec l’ensemble des classes Data qui représentent les données utilisées par l’application.

* **ServerDBController** : Il permet de vérifier l’état d’un serveur contenant une base de données ERP.
* **UserController**: Il permet de vérifier la connexion d’un utilisateur à la base de données distante.
* **ArticlesController**: Il permet de requêter la table Article pour récupérer des articles entre deux dates.
* **LignesController**: Il permet de requêter la table Ligne pour récupérer des lignes entre deux dates.
* **SalesController**: Il permet de requêter la table Document pour récupérer des documents de vente entre deux dates. Ces documents de vente peuvent être :
  + **Ventes Positives**: Les factures et les bons de livraisons non transférés.
  + **Ventes Négatives**: Les avoirs et les bons de retour non transférés.
* **TiersController**: Il permet de requêter la table Tiers pour récupérer des clients entre deux dates.
* **RepresentantsController**: Il permet de requêter la table Representant pour récupérer des représentants entre deux dates.

## Base de données minimale

La base de données minimale permet de garder une vision minimale de la base de données distante. Cette dernière permet d’afficher les données d’une manière asynchrone. Par exemple, dans le cas ou il n’y a pas une connexion internet, la base de données permet de fournir les données de la dernière mise à jour pour une exécution locale.

### Environnement de développement

La base de données minimale se trouve sur la partie mobile. Ainsi, Android fournit une base de données interne SQLite. Realm est une base de données mobile, qui remplace SQLite, qui à la fois simplifie l’utilisation de la base de données par rapport à la complexité de SQLite, mais qui aussi facilite l’accès aux données.



Figure 4 - Environnement de développement de la base de données minimale

### Schéma de la base de données minimale

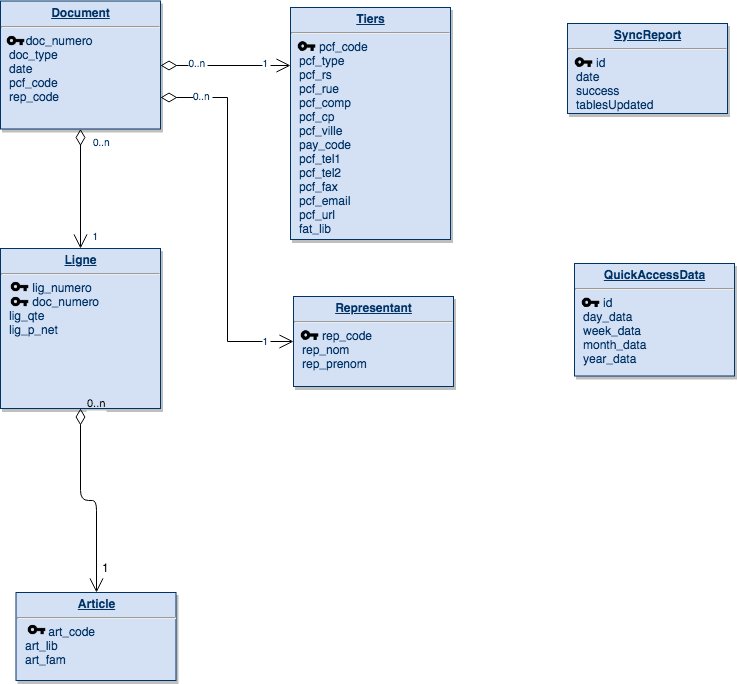


Figure - schéma de la base de données minimale

La base de données minimale contient une version minimale de la base de données de l’ERP en prenant en compte les données nécessaires pour afficher les données applicatives. Ainsi, la base de données minimale contient de plus deux tables principales :

* **SyncReport**: qui permet de stocker les rapports de synchronisation de la base de données avec la date de la synchronisation, son succès et l’ensemble des tables ayant été mise à jour, éventuellement la table qui n’a pas pu être mise à jour.
* **QuickAccessData**: qui permet de stocker à chaque mise à jour les données à afficher pour accélérer le processus d’affichage des données.

## Partie Applicative

La partie applicative développé sous Android en utilisant le langage de programmation Java et l’outil de création des interfaces qui se base sur XML, tout ça sous l’environnement de développement Android Studio.

### Environnement de développement

L’environnement de développement de l’application Android est l’outil puissant natif Android studio.



Figure - Environnement de développement de l'application Android

### Schéma global de l’application