

PlanEte dashboard Android

DOCUMENTATION

May 27, 2018

YOUNES SENNADJ

DSSI

Table des matières

Table des figures 3

Architecture 4

Serveur de synchronisation 4

Environnement de développement 5

Diagramme de classe 5

Base de données minimale 6

Environnement de développement 6

Schéma de la base de données minimale 6

Partie Applicative 7

Environnement de développement 8

Schéma global de l’application 8

UI 8

Activities 9

Fragments 10

Jour 10

Semaine 10

Mois 11

Année 11

Adapters 11

Data 12

Webservice 12

Data 12

HttpGetter 12

DataGetter 12

Background 12

UpdateJob 12

Database 12

SalesController 12

ArticlesController 13

ClientsController 13

RepresentantsController 13

DataBaseHandler 13

DataBaseUpdater 13

SalesFragmentDataSetter 13

Les libraires utilisées 14

# Table des figures

Figure 1 - Architecture fonctionnelle 4

Figure 2 - Environnement de développement de Serveur de synchronisation 5

Figure 3 - Diagramme de classe de serveur de synchronisation 5

Figure 4 - Environnement de développement de la base de données minimale 6

Figure 5 - schéma de la base de données minimale 7

Figure 6 - Environnement de développement de l'application Android 8

Figure 7 - schéma global de l'application 8

Figure 8 - Les activités de l'application 9

# Architecture

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

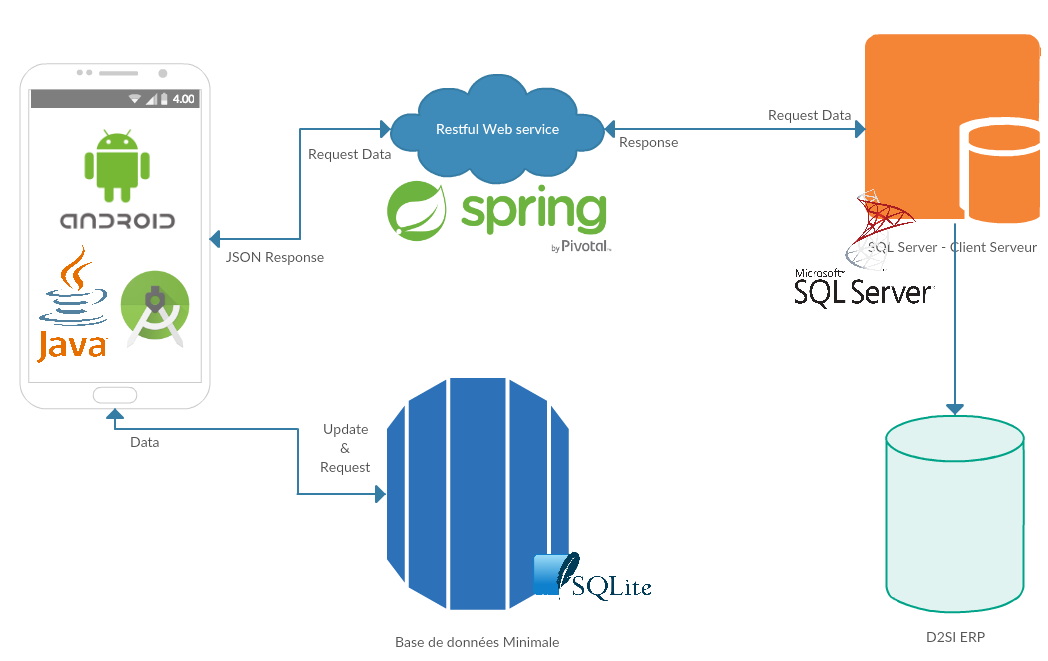


Figure - Architecture fonctionnelle

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

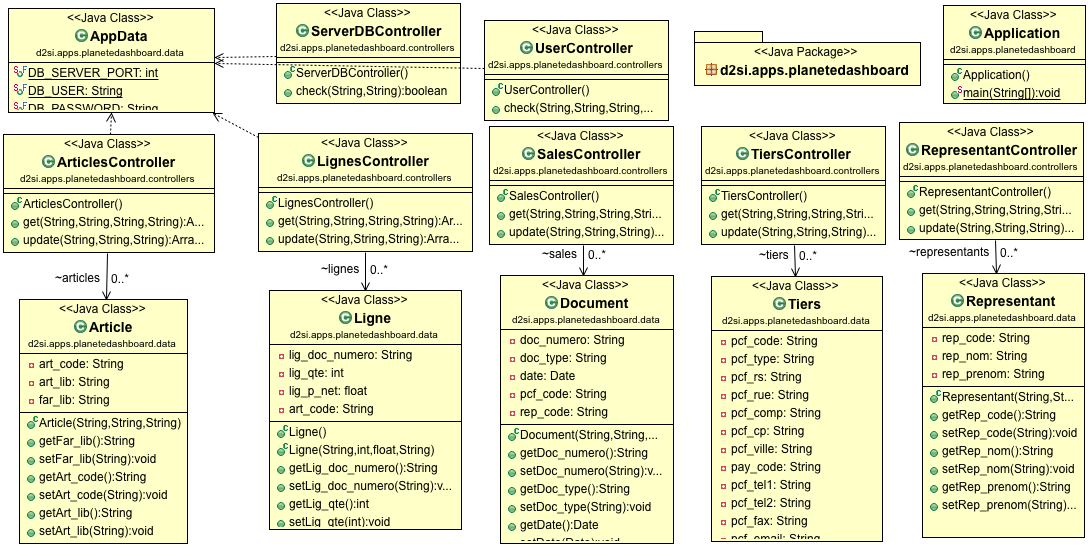
Spring permet via une architecture très simple de mettre en place un serveur recevant des requêtes GET et POST. L’ensemble des classes de serveur de synchronisation est représenté dans le diagramme suivant :

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 - Environnement de développement de la base de données minimale

### Schéma de la base de données minimale

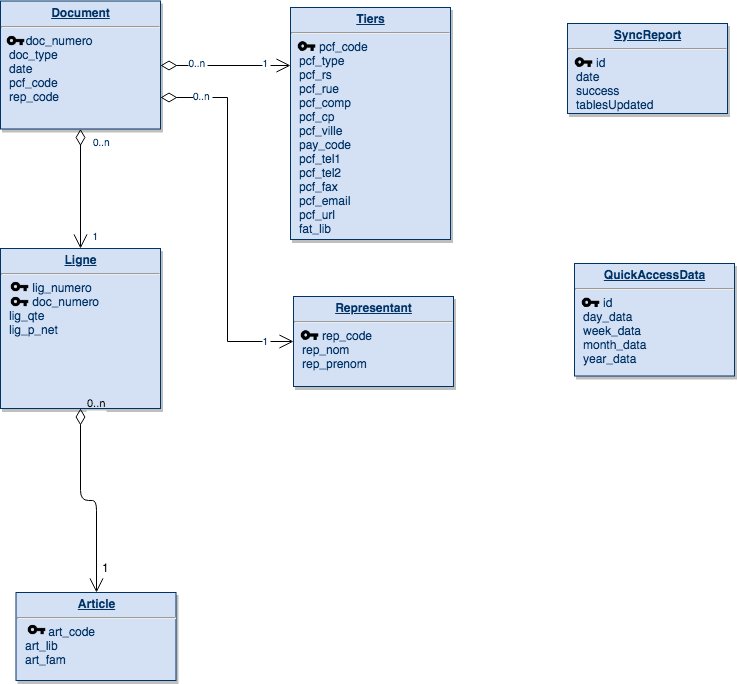


Figure - schéma de la base de données minimale

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

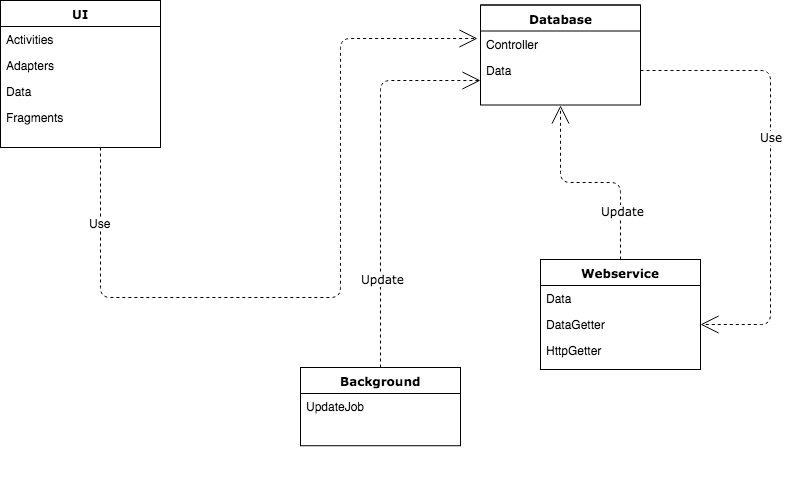


Figure - schéma global de l'application

L’application se divise principalement en 4 éléments globales :

#### UI

Elle regroupe l’ensemble des classes utilisées pour la création de l’interface et elle se compose en 4 parties principales.

##### Activities

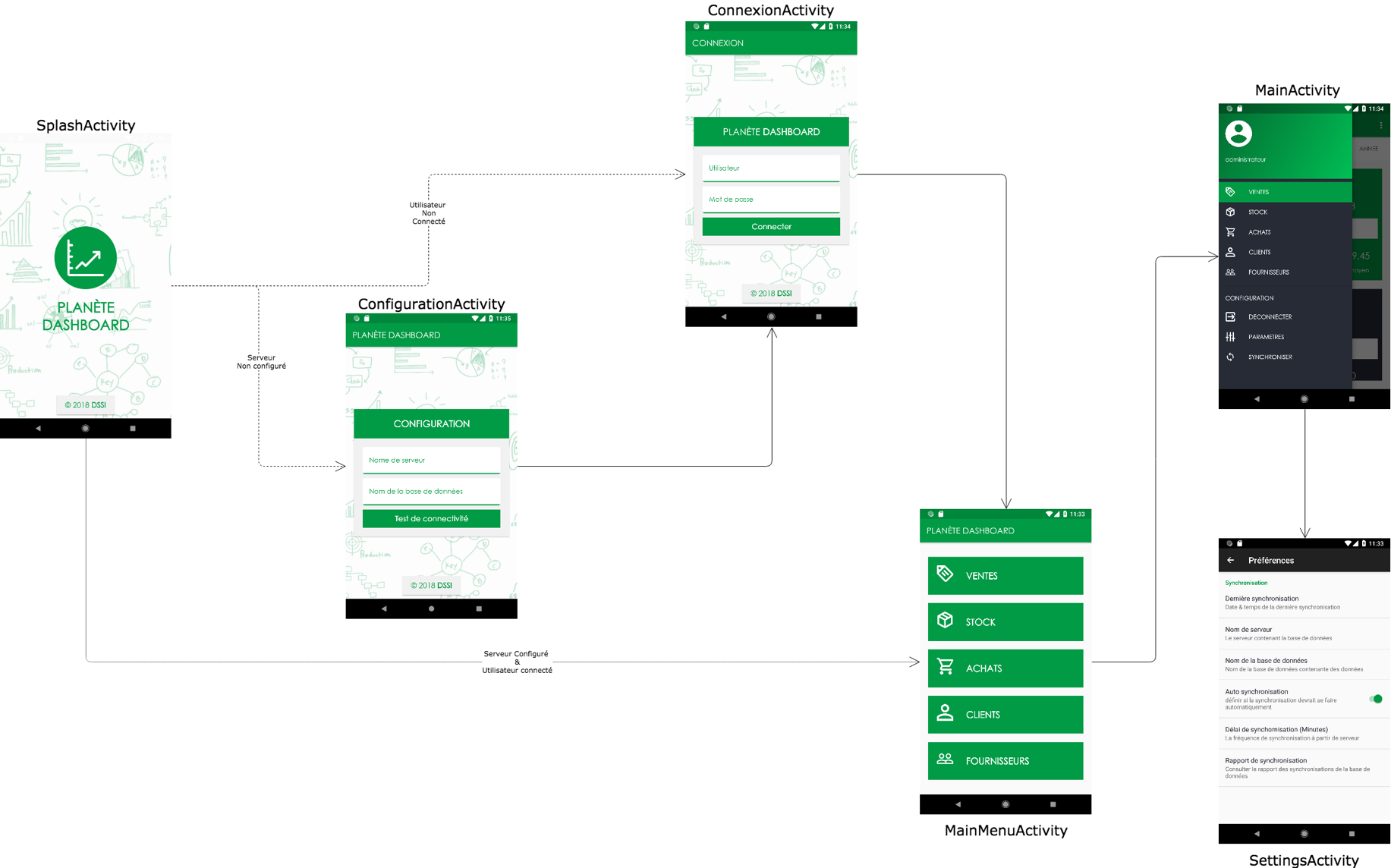
Elle contient les activités qui représentent une instance qui permet d’être affiché sur l’écran. Voici un schéma de séquences globale.

Figure - Les activités de l'application

L’ensemble des activités se présente comme suit :

* **SplashActivity**: elle permet de
  + Afficher le Slash Screen contenant le logo et le nom de l’application.
  + Naviguer vers la bonne activité : configuration de serveur si ce dernier n’a jamais été configuré, connexion de l’utilisateur si ce dernier est déconnecté ou inexistant, vers le menu principal de l’application si le serveur et l’utilisateur ont été bien configurés.
* **ConfigurationActivity** : elle permet de
  + Introduire le nom de serveur et le nom de la base de données à utiliser par l’application.
  + Tester la connectivité de serveur et de la base introduite et de naviguer vers l’activité de connexion en cas de succès.
* **LoginActivity** : elle permet de
  + Introduire le nom de l’utilisateur et le mot de passe à utiliser par l’application.
  + Vérifier les champs d’authentification introduits par l’utilisateur.
  + En cas de succès, l’activité automatiquement lance une mise à jour des données qui permet de recueillir les données des deux dernières années nécessaires pour l’application.
  + En cas de succès, naviguer vers l’activité de menu principal.
* **MainMenuActivity**: elle permet simplement de naviguer vers l’activité principale en spécifiant quel fragment à afficher.
* **MainActivity**: c’est l’activité principale de l’application, elle permet de :
  + Afficher le Dashboard des ventes, stocks, achats, clients et fournisseurs.
  + Déconnecter l’utilisateur.
  + Naviguer vers l’activité des paramètres.
  + Synchroniser la base de données manuellement.
* **SettingActivity**: elle permet d’afficher la dernière date de mise à jour réussite de la base de données et le rapport global des synchronisations, elle permet aussi de configurer des paramètres globaux de l’application :
  + Le nom de serveur.
  + Le nom de la base de données.
  + Permet la synchronisation automatique.
  + Spécifier le délai de la synchronisation périodique.

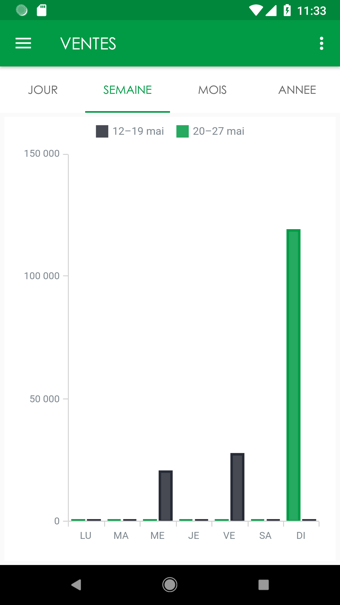
##### Fragments

Le fragment en étant un élément dynamique de représentation des vues. Pour la partie des ventes, on a 4 fragments principales, un affichage par jour, semaine, mois et année.

###### ../../../../../../../Desktop/Screenshot_1527417191.pngJour

Il permet de mettre en place une comparaison des ventes entre aujourd’hui et hier selon plusieurs indicateurs :

* Le chiffre d’affaire
* La quantité vendue (Positives et négatives)
* Le nombre de ventes
* Le prix moyen des ventes



###### Semaine

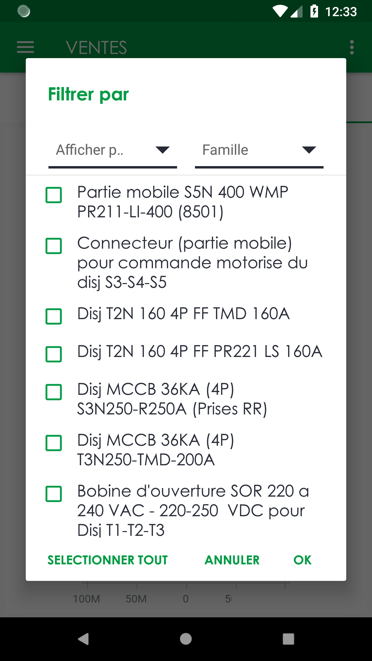
Il permet de montrer une comparaison entre la semaine actuelle et la semaine passée en terme de chiffre d’affaire représenté jour par jour.

###### ../../../../../../../Desktop/Screenshot_1527417211.pngMois

Il permet de montrer une comparaison entre le mois actuel et la mois passé en terme de chiffre d’affaire représenté jour par jour.

###### ../../../../../../../Desktop/Screenshot_1527417217.pngAnnée

Il permet de montrer une comparaison entre l’année actuelle et l’année passée en terme de chiffre d’affaire représenté mois par mois.



##### Adapters

On a utilisé pour l’application un seul adapter **ChoiceRecyclerAdapter** permettant d’afficher une liste avec un bouton à choix comme élément principale. Ce dernier a été utiliser pour l’affichage de filtre des données.

##### Data

Dans la partie data on a utilisé 3 classes, 2 classes qui permet de généraliser les chartes :

* **CustomBarChart**: Un graphe qui permet de présenter une charte avec des barres
* **CustomColumnChart**: Un graphe qui permet de présenter une charte avec des barres horizontales.
* **FilterCheckBox**: Une classe qui représente une case à coucher avec un texte et un booléen qui représente l’état de la case.

#### Webservice

##### Data

Représente les données qui sont implémenté dans la base de données mais avec une spécificité qui leur permettent d’être reçu à partir de serveur de synchronisation.

##### HttpGetter

Un ensemble de classe qui permet de recevoir et de mettre à jour la base de données. Ainsi, il permet de récupérer les données entre deux dates précises.

##### DataGetter

Cette activité a un rôle clé. Elle permet de :

* Connecter avec le serveur de données, recueillir des données et mettre à jour la base de données.
* Vérifier l’existence de l’utilisateur.
* Requêter le serveur distant, récupère les données pour la 1ere fois des deux dernières années.
* Requêter le serveur distant pour récupérer les données entre la dernière date de mise à jour et maintenant et mettre à jour la base de données.
* Gérer les erreurs en cas de problème.
* Générer le rapport de synchronisation et le stocker dans la base.
* Exécuter un bout de code après avoir terminer de requêter le serveur distant.

#### Background

Cette partie permet de synchroniser la base de données et de la mettre à jour en background sans besoin de l’interface.

##### UpdateJob

Il exécute un job périodiquement pour mettre à jour la base de données en arrière plan. Ce job ainsi permet de recalculer les données à afficher pour le prochain lancement de l’activité des ventes.

#### Database

L’ensemble des données a été présentées au préalable. Dans ce qui suit nous présentons les contrôleurs qui requête la base de données.

##### SalesController

Il permet de

* Calculer le chiffre d’affaire entre deux dates.
* Calculer le chiffre d’affaire des ventes positives et négatives entre deux dates.
* Calculer la quantité des articles vendus.
* Calculer le nombre des ventes.
* Calculer le prix moyen des ventes.
* Calculer les chiffres d’affaire de jour.
* Calculer les chiffres d’affaire de la semaine.
* Calculer les chiffres d’affaire de mois.
* Calculer les chiffres d’affaire de l’année.
* Mettre à jour les données de la base pour l’affichage rapide.

##### ArticlesController

Il permet de

* Récupérer tout les articles.
* Récupérer les noms ou les identifiants des articles.
* Récupérer les familles des articles.
* Récupérer une sous liste d’articles.

##### ClientsController

Il permet de

* Récupérer tout les clients.
* Récupérer les noms ou les identifiants des clients.
* Récupérer les familles des clients.
* Récupérer une sous liste des clients.

##### RepresentantsController

Il permet de

* Récupérer tout les représentants.
* Récupérer les noms ou les identifiants des représentants.
* Récupérer une sous liste des représentants.

##### DataBaseHandler

Il permet d’insérer un ensemble de données dans la base en asynchrone et notifie l’interface à la fin de l’écriture.

##### DataBaseUpdater

De même, Il permet de mettre à jour un ensemble de données dans la base en asynchrone et notifie l’interface à la fin de l’écriture.

##### SalesFragmentDataSetter

Il permet de mettre à jour les données à afficher dans l’activité des ventes en asynchrone sans interrompre l’interface UI.

### Les libraires utilisées

Développer dans un environnement open source a un grand avantage, surtout avec la contribution de grand nombre de développeurs autour de monde. Du coup, l’utilisation des bibliothèques sous Android facilite beaucoup la tâche. Dans le tableau suivant, nous présentons les libraires utilisées dans ce projet.

|  |  |  |
| --- | --- | --- |
| Librairie | Rôle | Lien |
| Realm | Base de données mobile facile à exploiter plus souple que SQLite la BDD par défaut de Android. | <https://realm.io/docs/java/latest/> |
| Spring | Bibliothèque qui permet de connecter au serveur de synchronisation développé avec Spring. | <https://spring.io/guides/gs/consuming-rest-android/> |
| AndroidJob | Elle permet de créer des jobs qui s’exécute en arrière plan même quand l’application ne s’exécute pas. | <https://github.com/evernote/android-job> |
| Anychart | Bibliothèque de graphes contenant un nombre élargi de graphe très facile à utiliser. | <https://github.com/AnyChart/AnyChart-Android> |
| ButterKnife | Bibliothèque qui permet facilement de binder les vues et leurs évènements. | <https://github.com/JakeWharton/butterknife> |
| MaterialDrawer | Un menu qui se fait glisser par la gauche comme celui de Google et qui respecte le Material design. | <https://github.com/mikepenz/MaterialDrawer> |
| Iconics | Permet d’utiliser les icones de plusieurs fonts connus comme étant des ressources drawables. | <https://github.com/mikepenz/Android-Iconics> |
| TypefaceHelper | Permet de modifier le font par défaut de l’application. | <https://github.com/norbsoft/android-typeface-helper> |
| MaterialRipple | Permet de générer l’effet Material de clique. Un effet qui respecte les recommandation de Google en terme de design. | <https://github.com/balysv/material-ripple> |
| TextFieldBoxes | Bibliothèque de barre à saisie avec un design qui répond aux spécifications Material. | <https://github.com/HITGIF/TextFieldBoxes> |
| MaterialDialog | Bibliothèque des dialogues avec un design Material. | <https://github.com/afollestad/material-dialogs> |
| MaterialSpinner | Bibliothèque de liste déroulante avec un design Material. | <https://github.com/ganfra/MaterialSpinner> |
| MaterialPreferences | Bibliothèque permettant de générer les paramètres avec un design Material. | <https://github.com/ferrannp/material-preferences> |