Damien Dussurget, Antoine Ho,

Oussama Lourhmati, Kha Pham, Luke Woodley

**Document d’architecture et de design**

Travail présenté à

Georges Coté

Travail Pratique 2



Collège de Bois-de-Boulogne

Le 9 décembre 2019

Table des matières

[Introduction 3](#_Toc26735703)

[Modèle de la base de données (API) 4](#_Toc26735704)

[Modèle UML (Diagramme de classes) 5](#_Toc26735705)

[Design et maquettes 7](#_Toc26735706)

[Activités 9](#_Toc26735707)

[Architecture du code (MVP) 10](#_Toc26735708)

[Sources 12](#_Toc26735709)

[Auto-évaluation 13](#_Toc26735710)

# Introduction

Ce document présente un projet d’application Android qui consiste à developper une application mobile qui s’appelle « Défi photo » mise en place par la Commission scolaire de la Pointe-de-l'Île (CSPI).

L’application sera destinée à des appareils Android et sera développée à l’aide d’Android Studio. Il est aussi envisagé, si le produit est approuvé, de le déployer également sur IOS. L’objectif est que cette application soit disponible pour tous les OS et n’importe quel étudiant de la CSPI puisse l’installer quel que soit le type de son appareil mobile tel qu’un téléphone cellulaire ou une tablette.

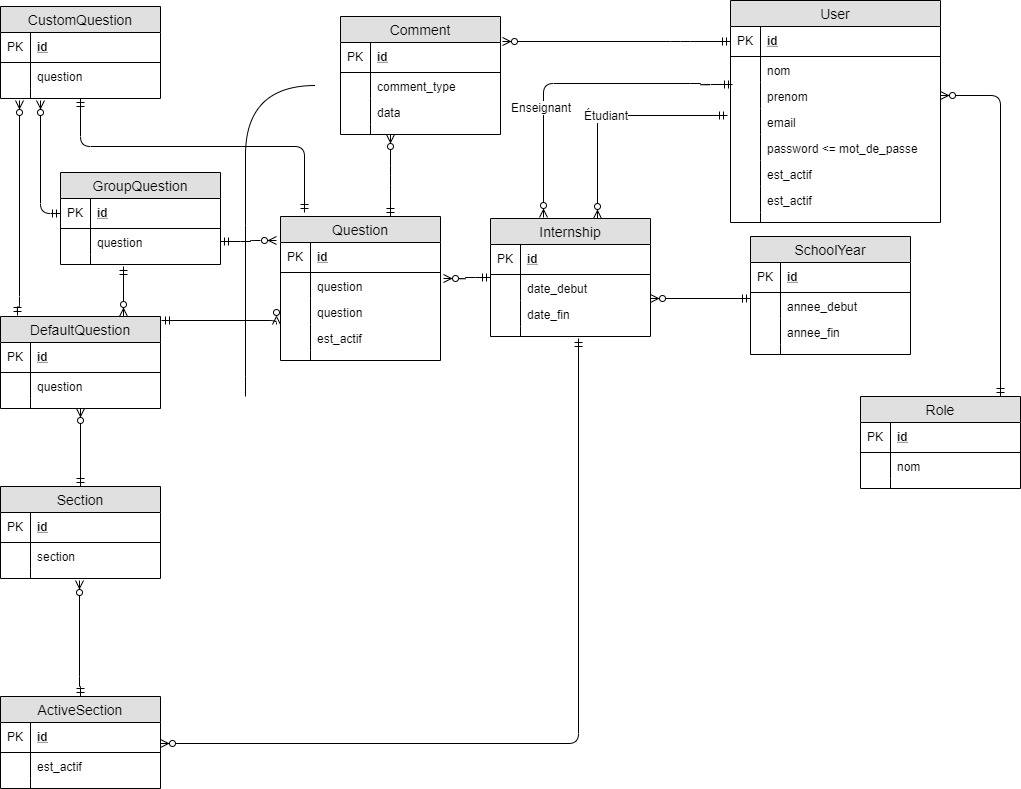
Plus précisément, l'application Défi photo aura comme principale tâche de pouvoir superviser l’avancement des stages des étudiants de la commission scolaire. L’enseignant (superviseur) aura la possibilité de visualiser l’avancement de chacun de ses étudiants, d’approuver leurs tâches et de rédiger des commentaires et poser des questions. L’étudiant aura à son tour la possibilité de prendre des photos pour chaque question et d’y répondre. La page d’accueil de l’étudiant sera divisée en plusieurs catégories, et c’est en cliquant sur une catégorie qu’il pourra accéder aux questions propres à la catégorie.

Dans ce document, nous allons aborder notre diagramme de classes et expliquer la logique derrière celui-ci. Nous présentons aussi quelques exemples d’interfaces ainsi que le design des différents écran choisis afin de montrer à quoi ressembleront les interfaces réelles de l’application. Par la suite, nous aborderons des différentes activités et le mode de communication entre elles. L’architecture du code sera aussi expliquée. Ce qui permet de définir les conventions que nous allons utiliser pendant le développement du projet. Elle permettra aussi une plus facile continuité si une équipe différente tiendra à améliorer ou ajouter des fonctionnalités à notre application.

L’API, qui permet d’accéder à la base de données, nous a été fournie. De ce fait, nous n’aborderons pas en détail le fonctionnement de l’API. En revanche, nous présenterons brièvement le modèle de base données de l’API.

# Modèle de la base de données (API)

Le modèle de base de données à utiliser dans le cadre de ce projet est illustré dans le diagramme ci-dessus

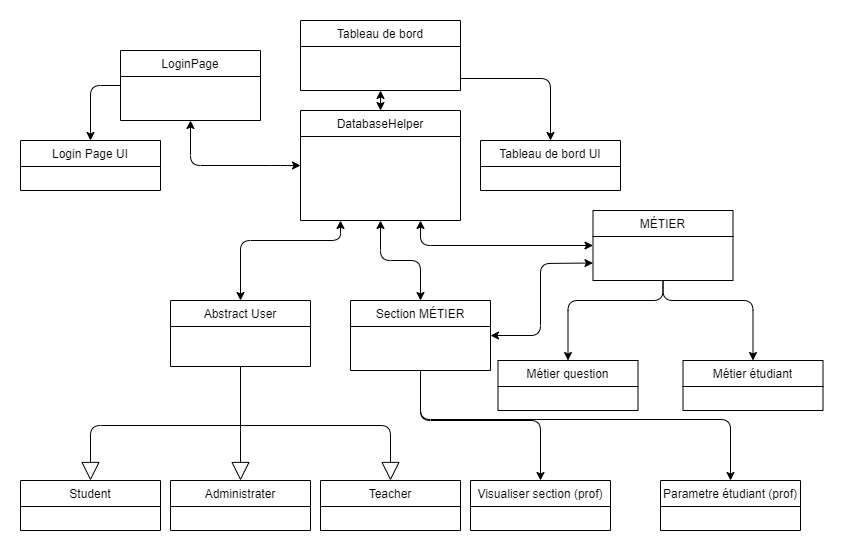


*Figure 1 : Diagramme entité-relation de la base de données*

Ci-dessus, le schéma de la base de données relationnelle à utiliser dans le cadre de ce projet. Il est constitué à base de plusieurs tables interreliées via des jointures de différents types. Brièvement, on peut voir qu’un utilisateur a un rôle (table Rôle), donc soit un administrateur, un enseignant ou un élève. Selon son rôle, l’utilisateur aura la possibilité de soit répondre à une question (table Question) ou soit rédiger un commentaire (table Comment). Plus encore, on peut voir que dans la base de données, il y a une table consacrée à l’année scolaire (SchoolYear), vu que l’enseignant peut choisir l’année scolaire des étudiants qu’il veut superviser. Finalement, on peut remarquer qu’il y a une table GroupQuestion, puisque les questions seront groupées en catégorie (MATIÈRE ET PRODUITS, ÉQUIPEMENTS, TÂCHES ET EXIGENCES, INDIVIDU, ENVIRONNEMENT DE TRAVAIL, RESSOURCES HUMAINES), c’est ce qui forme l’acronyme M.É.T.I.E.R.

# Modèle UML (Diagramme de classes)

Différentes classes ont été définies dans le cadre de ce projet. Le diagramme ci-dessous illustre le modèle UML conçu.



*Figure 2 : Diagramme de classes*

Tel qu’illustré ci-dessus, il y a la classe DatabaseHelper, qui nous aidera à faire les liens entre l’application et l’API de la base de données que nous utilisons. Par exemple, lors de l’ajout d’un commentaire par un professeur, une méthode du DatabaseHelper sera appelée pour stocker l’information dans notre base de données.

Par la suite, on peut observer la classe LoginPage qui fait le lien entre le UI de la Login Page et l’API. Il s’agit de la page de login qui permet l’authentification des utilisateurs ( étudiants, professeurs et administrateurs) et c’est à ce niveau que le rôle de chacun est défini dans l’application.

Nous avons choisi d’utiliser une classe abstraite User (utilisateur) qui sera étendue à trois classes enfants : Student (étudiant), Administrator (administrateur) et Teacher (professeur). Chacun de ces utilisateurs aura accès à des écrans spécifiques à son rôle. Dans le cas d’un étudiant, il n’aura accès qu’à sa fiche tandis qu’un professeur aura accès à l’ensemble des étudiants. De plus, le professeur aura la possibilité par exemple de limiter l’accès à certaines sections à ses étudiants.

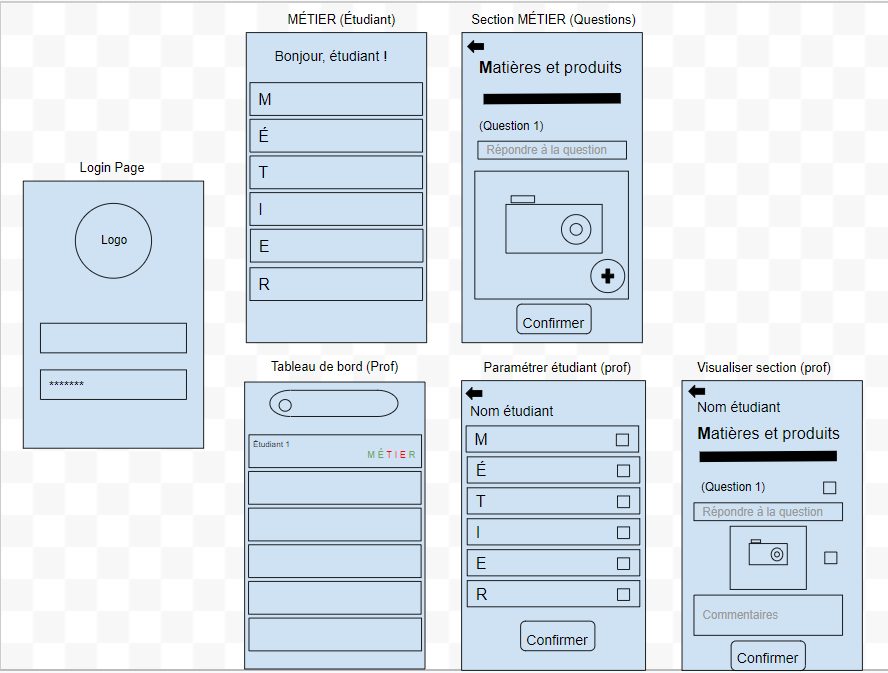
Dans le cas d’un professeur, une fois authentifié, il aura accès au tableau de bord qui représentera une liste des différents étudiants qui sont associés à lui. Il aura aussi accès à une barre de recherche pour trouver plus rapidement un élève en particulier. Les étudiants seront triés par niveau puis par ordre alphabétique.

Parlant de sections, on peut aussi observer les classes Section MÉTIER et MÉTIER. La classe Section MÉTIER représente les sections de l’acronyme MÉTIER.

Par exemple, lorsque l’étudiant va sur la section M il aura accès à la section Matières et produits. Il pourra ainsi visualiser les questions que son professeur lui a posé avec la possibilité de prendre une photo ou en fournissant une photo de son téléphone cellulaire. Il aura aussi la possibilité de mettre un commentaire à celle-ci. Le professeur aura accès à la même vue avec la possibilité de choisir les questions qu’il souhaite poser à ces étudiants.

# Design et maquettes

Afin d’avoir une idée sur l’aspect visuel de l’application, nous avons conçu et élaboré les maquettes suivantes :



*Figure 3 : Interfaces de l’application*

Tout d’abord, il y a une page d’accueil, où l’utilisateur sera invité à s’authentifier. Il y aura le logo de la CSPI et la possibilité de se connecter avec un compte Google.

Dépendamment de son rôle, l’utilisateur sera redirigé soit vers la page d’accueil s’il s’agit d’un étudiant (page M.É.T.I.E.R) ou soit vers le tableau de bord dans le cas d’un enseignant.

La page d’accueil de l’étudiant sera divisée en différentes catégories (formant l’acronyme MÉTIER). Il y aura aussi un message de bienvenue à l’étudiant.

Lorsque l’étudiant clique sur une catégorie, il sera redirigé vers une nouvelle page (Section MÉTIER) dédiée à la catégorie. Sur cette page, il y aura la barre de progression de l’élève et des fragments. Chaque fragment sera constitué de la question, d’un Edittext pour pouvoir répondre à la question et d’une photo que l'étudiant va sélectionner ou prendre. Lorsque l'étudiant termine de répondre à ses questions, il peut cliquer sur le bouton confirmer qui va le rediriger vers sa page d’accueil.

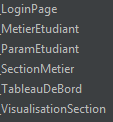
Dans le tableau de bord de l’enseignant, il y aura une liste de tous ses étudiants (sous forme de Recyclerview). Il aura la possibilité de rechercher un étudiant grâce au Search Bar et de trier ses élèves selon l’année scolaire. Une fois que l’enseignant trouve l'élève qu’il veut superviser, il peut cliquer dessus, ce qui le redirige à une nouvelle page où il peut confirmer chacune des catégories de l'élève en cliquant sur le checkbox approprié. Lorsque l’enseignant termine ses confirmations, il peut cliquer sur le bouton confirmer, qui va le ramener au tableau de bord. S’il veut voir une catégorie plus précisément et visualiser les réponses et les photos de l’étudiant, il peut cliquer sur la catégorie et accéder à une nouvelle page. Dans cette nouvelle page, il peut voir la progression de l'élève, confirmer chacune des questions et images, et rédiger un commentaire à l’étudiant.

Nous avons voulu l'application soit la plus intuitive possible, pour faciliter l'expérience à l'étudiant et à l'enseignant. Pour cela, nous n'avons pas surchargé les différentes pages, nous avons juste mis l’essentiel. De plus, nous avons aussi implémenté une Navigation Drawer pour faciliter le tri des élèves dans le tableau de bord, et des questions dans la page d’accueil de l’étudiant (questions triées par catégorie).

Finalement, pour le design, nous avons choisi les couleurs bleu et noir, les couleurs de la CSPI. Dans la page d’authentification, nous avons fait un fond d’écran animé qui change de couleurs. Finalement, pour rendre l’application élégante et belle esthétiquement, nous avons décidé d’utiliser des cardview pour les différents items des recycleview.

# Activités

Les activités principales définies dans le cadre de ce projet sont illustrées ci-dessous.

****

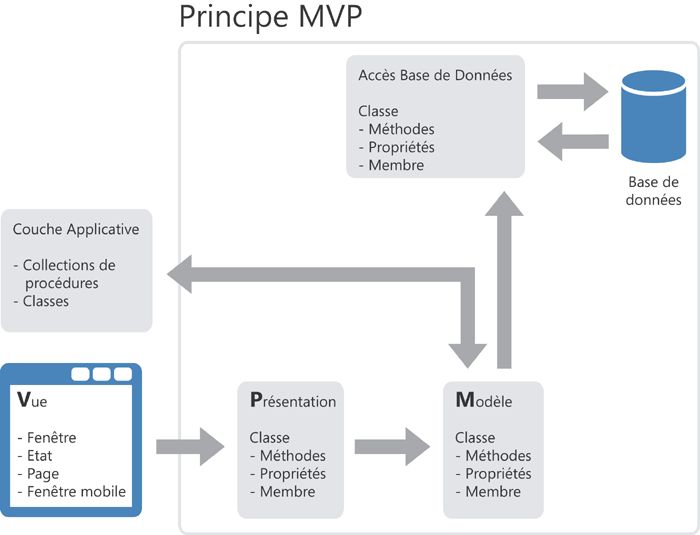
Nos activités reprennent la même structure que nos maquettes. Il y aura donc :

* Une activité consacrée à la page d'authentification (LoginPage)
* Une activité consacrée à la page d’accueil de l’étudiant (MetierEtudiant),
* Une activité liée aux questions auquel l’élève doit répondre (SectionMetier)
* Une activité consacrée au tableau de bord de l’enseifnant (TableauDeBord)
* Une pour que l'enseignant puisse confirmer les sections (ParamEtudiant)
* Une autre pour que l’enseignant puisse visualiser une section en particulier et aux réponses de l’élève (VisualisationSection).

La communication entre les activités et le passage d’une activité à une autre se font avec l’aide des intents et des méthodes StartActivity(), StartActivityForResult() ou putExtra(). Grâce aux intents, on peut par exemple, envoyer les informations d’un élève, d’une activité à une autre sans souci.

# Architecture du code (MVP)

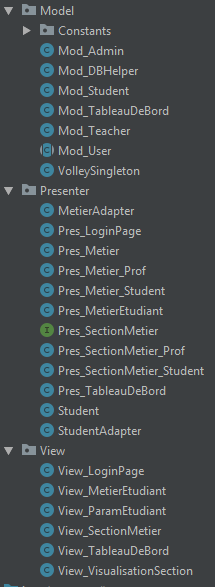
L’architecture du code MVP a été conçue selon le pattern illustré dans le diagramme suivant



Source: <https://doc.pcsoft.fr/fr-FR/?1000021496>

Pour l’architecture de notre code, nous avons choisi le Design Pattern MVP (modèle, vue, présentation). Tel que mentionné dans le lien ci-dessus, l’idée de cette architecture est que nos différentes interfaces utilisateurs soient découpées et structurées en plusieurs couches. De cette manière, notre code derrière l’interface (backend) sera séparé du code qui manipule directement l’interface. Grâce à cette architecture, la maintenance et l’évolution du code seront plus facile.

Pour résumer le schéma ci-dessus, le modèle ne va jamais directement communiquer avec la vue (l’interface utilisateur). C’est le présentateur qui va faire la communication entre le modèle et la vue. Le modèle (Model) contient les données à afficher dans notre application est correspond à l’API que nous utilisons. Le présentateur (Presenter) quant à lui contient la logique concernant les actions effectuées par l’utilisateur. En somme, il est là pour recueillir les données et/ou les actions de l’utilisateur dans l’application et de les transmettre au model. Le présentateur a pour but de faire le lien entre l’interface graphique et le modèle. La vue (View) contient la présentation de l’interface graphique. Elle est supposée de ne contenir que du code en rapport avec l’esthétique et la disposition de l’interface.



Comme on peut le voir ici, au niveau du projet, nous avons respecté le principe de MVP. Pour cela, nous avons créé 3 packages différents, un pour la vue (View), un pour la présentation (Presenter) et pour le modèle (Model). Nous avons bien segmenté notre code en différentes parties (ex: **View**\_LoginPage, **Pres**\_LoginPage), et que nous avons placé dans le bon package.

# Sources

**Liens:**

* <https://developer.android.com/training/basics/firstapp/starting-activity>
* <https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-pr%C3%A9sentation>
* <https://fr.wikipedia.org/wiki/Diagramme_de_classes>
* <https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes>
* <https://www.youtube.com/channel/UC_Fh8kvtkVPkeihBs42jGcA>
* <http://download.androidhive.info/login>
* <https://codinginflow.com/>
* <https://doc.pcsoft.fr/fr-FR/?1000021496>

**Documents:**

* CAHIER DES CHARGES APPLICATION « DÉFI PHOTOS » – VERSION 3 20 août 2019
* CAHIER DES CHARGES APPLICATION « DÉFI PHOTOS » – VERSION 4 1er novembre 2019
* DÉFI PHOTO - Cahier de l’élève
* DÉFI PHOTO – J’OBSERVE ET J’AGIS Guide à destination des enseignants
* Le défi photo (PowerPoint)
* Travail Pratique II Version 1.0 Par Georges Côté

# Auto-évaluation

**Points forts :**

* Taches de priorité 1 accomplies (pages)
* Implémentation de l’API pour la page d’authentification et la déconnexion
* Utilisation de l’API pour la page visualiser section métier (enseignant) et tableau de bord
* Pages esthétiques et intuitives
* Code structuré (MVP: model, view, presenter)

Nous avons manqué de temps, donc nous ne pouvions développer l’ensemble de l’application

**Points faibles :**

* L’API n’a pas été implémenté pour toutes les pages
* Taches de priorité 2 et 3 non accomplies

Nous avons à plusieurs reprises travaillé en pair-programming, mais voici, en général, nos contributions individuelles.

**Contributions individuelles :**

* Damien Dussurget : Document, Page Paramétrer Étudiant, Débogage Page Tableau de Bord
* Antoine Ho : Page Visualiser Section (Prof), DataBaseHelper avec API (mineur), interface Page Login, Navigation Drawer
* Oussama Lourhmati : Document, Correction du document, Page Tableau de Bord, interface Page Login
* Kha Pham : DataBaseHelper avec l’API (majeur), authentification utilisateur
* Luke Woodley : Page METIER, Page Section METIER