E-Learning Factory

Conception et Programmation Orientées Objets (C#, .NET)

Octobre 2017

Desamais Florian

Neuts Benoît

Sommaire

Présentation de l’application

Contexte

- Une application mobile
- Permettant aux utilisateurs de suivre des contenus de cours sur leur téléphone (cours écrits, questionnaires, vidéos, des ressources externes)
- Les utilisateurs s'abonnent à des cours ou sont inscrits par un administrateur
- Les cours sont triés dans une arborescence par catégories
- État d'avancement, état de progression de l'utilisateur dans le cours auquel il est inscrit

Un peu de vocabulaire pour comprendre les différents niveaux d’arborescence de l’application

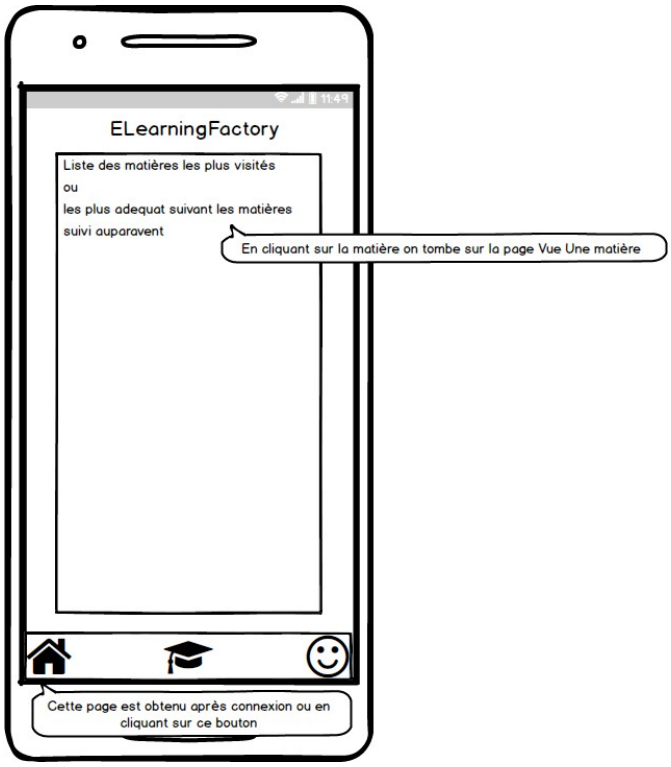
- Un ensemble de cours constitue une progression. Cet enchaînement de cours se situe dans une matière donnée.

- L’unité de base est un cours, c’est-à-dire une séance que pourra suivre l’utilisateur. Ces cours peuvent être sous différentes formes (texte, image, évaluation, etc.).

- La progression peut se subdiviser en chapitres qui sont des sous-ensembles de cours.

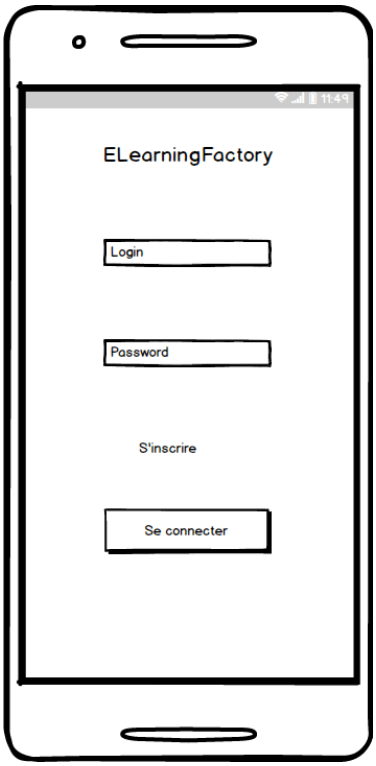
Présentation du design de l’interface

Accueil



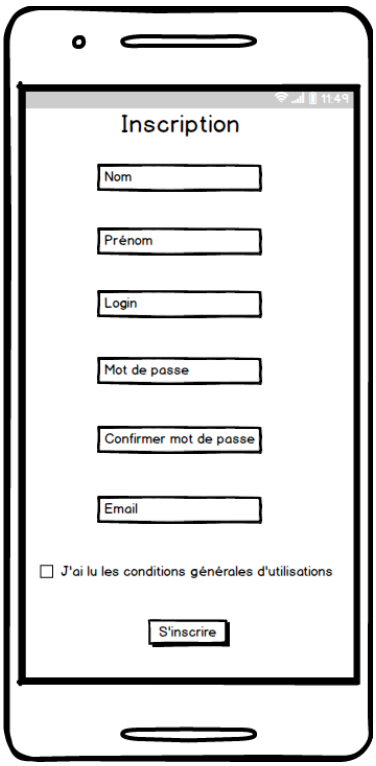
La navigation par catégories ainsi que les résultats de recherche utiliseront cette vue pour s’afficher.

Connexion



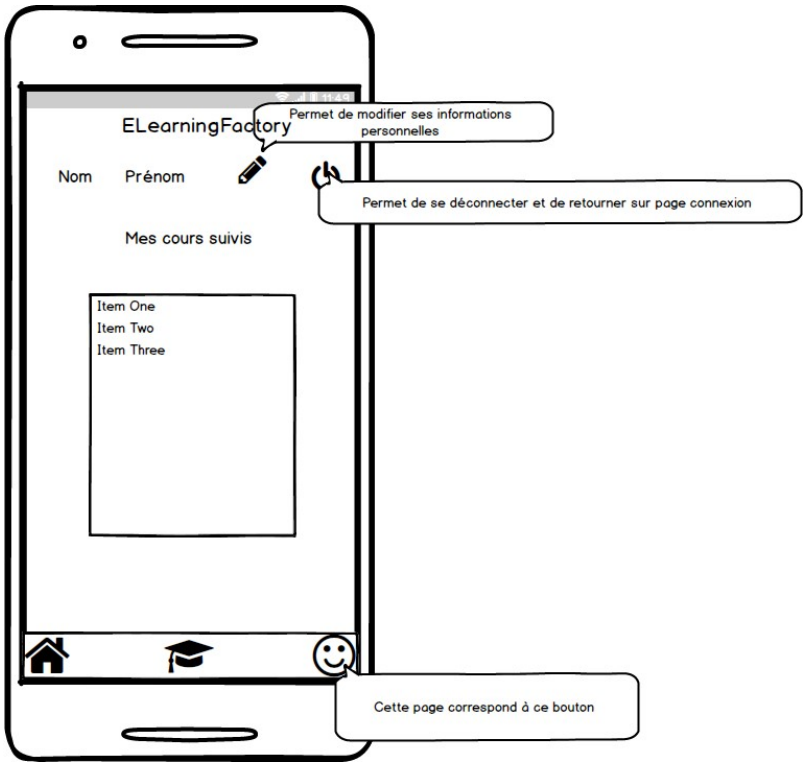
Cette vue permettra aux utilisateurs ayant déjà créé leur compte de se connecter et de retrouver les cours qu'ils auront déjà suivis.

Inscription



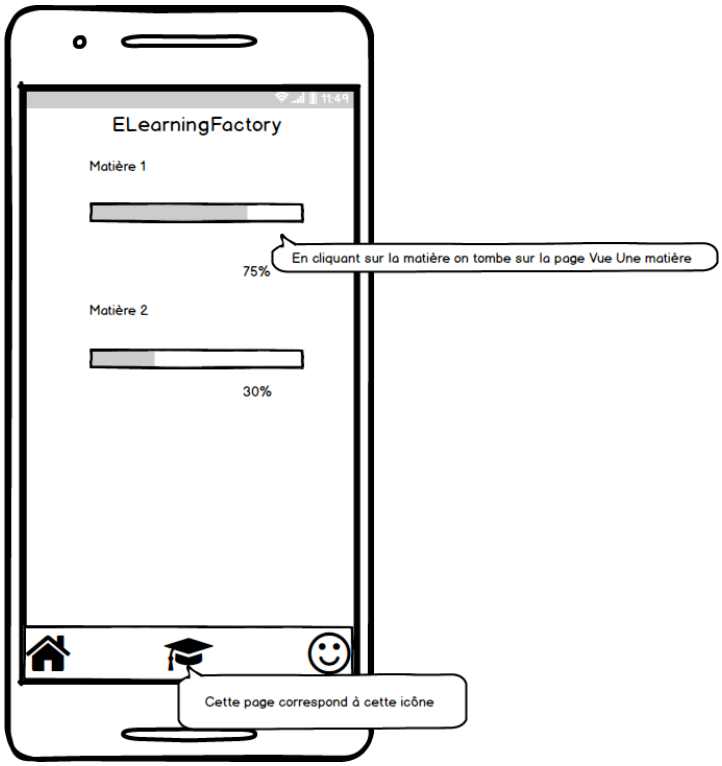
Sur cette vue, les utilisateurs pourront créer leur compte en fournissant leurs informations personnelles et leurs informations d'identification.

Mon compte



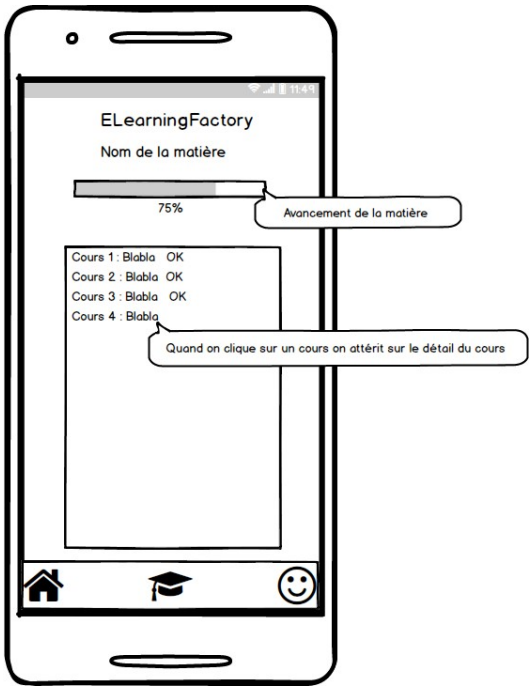
Une fois connecté, l'utilisateur connecté est renvoyé sur cette page. Sur cette vue, l'utilisateur pourra accéder à son compte et au détail des cours auxquels il a participé ou s'est inscrit. Il pourra modifier les coordonnées enregistrées dans l'application et se déconnecter.

Mes cours suivis



Il pourra voir de manière plus détaillée grâce à cette vue, son pourcentage d'avancement sur ses différentes progressions.

Vue principale de la matière (détail des séances)



Une progression sera affichée selon les différentes séances la composant comme présenté ci-dessus.

Vue de la séance (différents types : vidéo, écrit, QCM, etc.)

Le cours se présentera de manière très sobre : un titre, son contenu. S'ajoutera la possibilité de naviguer aux cours suivant ou précédent directement sur la vue du cours.

Personas

→ *visiteur non inscrit*

Il a accès aux ressources mais ne peut pas s'inscrire à une progression générale et donc pouvoir suivre l'état de son avancement dans une thématique donnée.

L'inconvénient pour cet utilisateur est de ne pas pouvoir sauvegarder sa progression directement dans l'application.

→ *étudiant ayant créé son compte*

L'utilisateur inscrit peut suivre sa progression. C'est-à-dire que lorsqu'il va changer de terminal, il peut se connecter grâce à ses identifiants et retrouver son avancement dans un ensemble de cours.

Dans un premier temps, pour des questions de temps, nous orienterons nos développements UI pour les étudiants et non pas sur les personnes qui seraient amenées à publier ou créer des cours. Cependant, voici ce que le persona enseignant pourrait présenter.

→ *enseignant*

Il pourra par la suite (cette fonctionnalité et ce rôle ne sont pas prévus dans cette version) :

- créer de nouveaux cours et des progressions.
- évaluer l'état de la progression d'un groupe ou d'un étudiant
- créer des groupes et des étudiants

Cas utilisateurs (*use cases*)

→ *visiteur non inscrit*

Il a accès aux ressources mais ne peut s'inscrire à une progression et ainsi se repérer au sein de l'application sur les cours qu'il a déjà suivis.

L'application lui propose de :

- s'inscrire
- rechercher une matières
- participer à un cours

→ *visiteur inscrit*

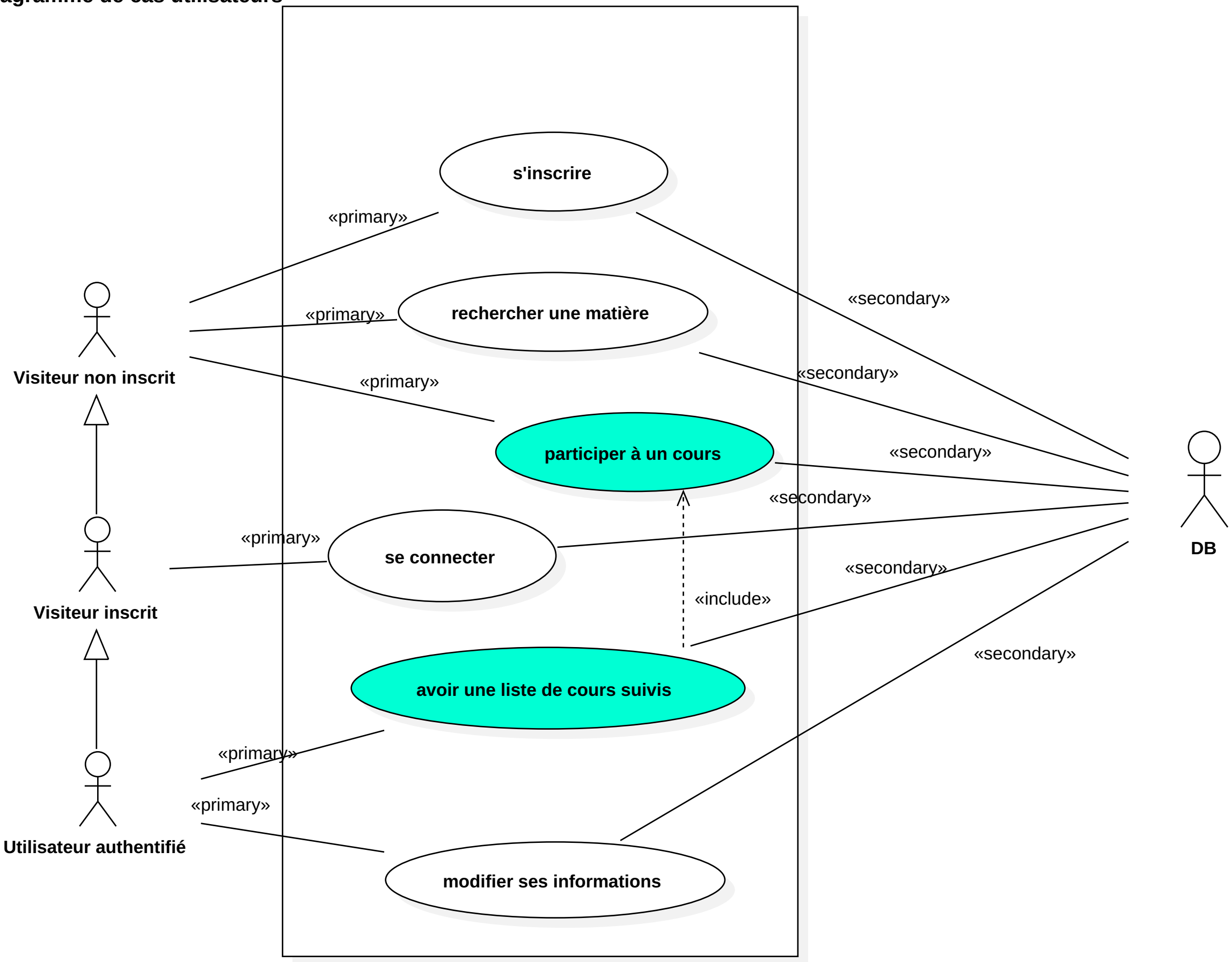
Celui-ci pourra se connecter via ses identifiants définis lors de son inscription dans l'application.

→ *utilisateur authentifié*

L'utilisateur authentifié accédera à sa liste de cours suivis. Il pourra reprendre là où il en était. Par ailleurs, il aura accès à ses informations personnelles afin de les consulter ou de les modifier.

Vous trouverez page suivante le diagramme de cas utilisateurs.

Diagramme de cas utilisateurs



Architecture

Présentation générale

L'architecture du projet a été pensée sur un modèle **MVVM** (**M**odel **V**iew **V**iew **M**odel). Dans un premier temps, nous développons le **Model** qui s'interfacera avec le reste de l'architecture **View Model** développé dans un second temps et non présenté ici autrement que par le package **View**. Nous terminerons par le développement du **View** en `UWP` déployé pour Windows Phone et en `Xamarin` pour terminaux mobiles Android et Apple.

L'architecture est structurée autour de 4 pôles :

1. La gestion des cours qui repose sur un **pattern Composite** permet de définir comme un tout un ensemble de cours. Ce tout définit une progression de plusieurs cours de différents formats. Plusieurs progressions peuvent exister dans l'application. Ce `pattern composite` nous sera utile pour représenter l'arborescence des cours. 2 types de contenus seront développés dans cette version : du contenu textuel et des évaluations.

2. La gestion des utilisateurs s'axe autour d'une classe abstraite principale ***Utilisateurs*** dont héritent **Etudiant** et **Prof**. Cette dernière classe ne sera développée que dans une version future. Cette classe abstraite principale nous permet de définir des données et des méthodes génériques à tous les utilisateurs.

3. La persistance des données sera assurée par l'interface ***IDataManager*** qui sera implémentée par les classes qui assureront le stockage en fonction du type de données. 3 types de données sont possibles : (1) un *stub* (classe **Stub**) qui fournira un lot de données permettant d'effectuer les tests (package **Test**) ; (2) un stockage local grâce à la classe **Local** ; (3) un **DBManager** est envisagé en XML.

4. Afin de sécuriser l'accès au **Model** par le **View Model** et de simplifier et centraliser les interactions entre modèle et vue, un ensemble de méthodes ainsi que l'accès au modèle est regroupé dans une façade (**Pattern facade**).

Dépendances au sein du projet

Voir diagramme de packages ci-après pour identifier leur situation.

Couche Présentation

Cette partie sera développée plus tard dans l'année (`View` et `View Model`).

Couche Application

Le package `Core` de l'application regroupe les 3 pôles décrits précédemment : gestion des cours, gestion des utilisateurs, façade y compris ***IDataManager***. Seules les classes spécifiques aux types de données resteront en dehors de `Core`.

La `façade` (**FacadeApplication**) va interagir avec les classes spécifiques du projet et avec le `View Model` développé en P2. Le reste du package `Core` n'aura pas d'interactions directes en dehors du package.

Le package `Core` est une bibliothèque de classes.

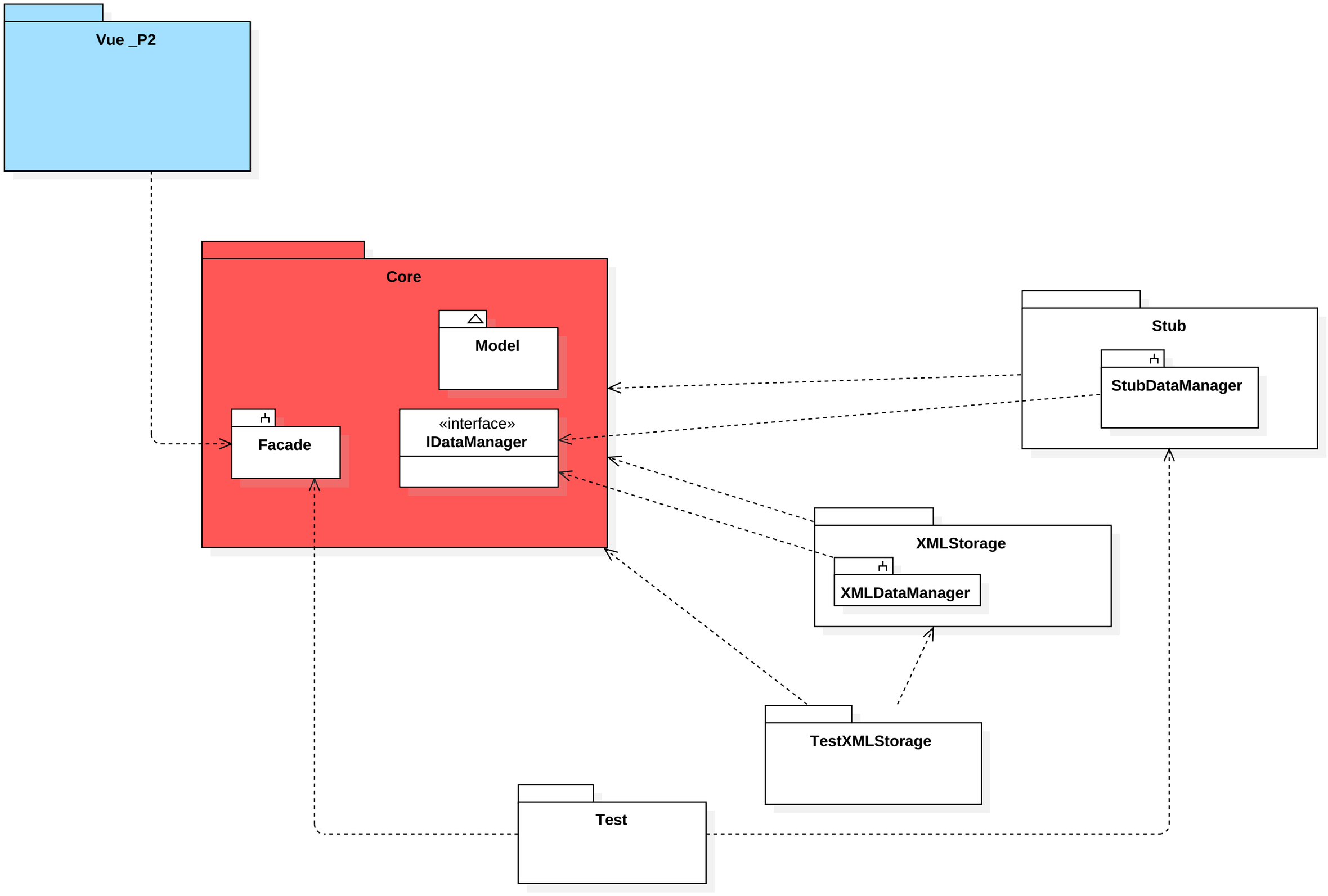
Le package `Test` sera l'exécutable en mode console qui testera le fonctionnement correct du projet réalisé en P1.

Couche Persistance

Le package `Stub` fournira un ensemble de données test (Utilisateurs, Cours) qui seront traitées par le **DataManager** du `Core`. `Stub` est une bibliothèque de classes.

`XMLStorage` fournira les méthodes de stockage des données au format XML.

Diagramme de packages



Classes du modèle

Nous avons transposé la représentation de notre contexte et de notre modèle selon le diagramme de classes ci-après.

On retrouve le pôle « gestion des cours » en haut à droite qui décrit un pattern composite autour d'un composant « Cours » dont les classes SousCours (composite), Texte (feuille) et Evaluation (feuille) héritent. Seuls Texte et Evaluation contiendront effectivement une séance de cours.

Au centre à gauche, le pôle « gestion des utilisateurs » avec la classe abstraite Utilisateurs dont hérite Etudiant. La classe Etudiant aura un ensemble de cours stocké au travers d'une collection « progression ».

A gauche en vert, se trouve la façade, pôle central pour les futurs développements représenté par une classe unique qui interfacera le modèle avec les vues qui seront développées dans le futur.

Enfin le dernier pôle, celui de la persistance, se trouve dans l'interface IDataManager dont hérite les classes de gestion de données StubDataManager et Local.

En gris foncé, se trouvent les classes qui ne feront pas l'objet de développement durant la première phase.

Diagramme de classes