PROJET HFGL

Document d'architecture du logiciel

Version 2.0

Document d'architecture HFGL

1. Introduction

1.1 Objectifs

Ce document fournit une vue globale des besoins fonctionnelles et architecturale du système, de haut niveau, grâce à un certain nombre de vues qui décrivent les différents aspects du système. Ces vues montrent les décisions significatives prises concernant l'architecture et les fonctions du système qui influenceront la conception.

1.2 Portée

Le document d'architecture logiciel est destiné à l'architecte, aux concepteurs et aux superviseurs du projet. Toute modification sur le document d'architecture du logiciel sera répercutée sur les cas d'utilisation de l'application.

2. Cahier des charges

Description du problème

Il s'agit de créer un site Web qui permettrait à un étudiant de pouvoir s'inscrire très rapidement dans un cours et de suivre le cours et les examens liés à ce cours, à disposition de tous les étudiants.
Ainsi, seraient disponibles programmes, Questions des différents genres, diaporamas, transparents, comptes rendus,….
Le site doit permettre aussi de soumettre les examens et les comptes rendus.
Le site est structurée en plusieurs espaces suivant les thèmes abordés et les acteurs concernés dans chaque espace sera disponible les documents et informations liés au thème.
Chaque espace est administré par un ou plusieurs gestionnaires qui ont la responsabilité du contenu de cet espace.

3 types d'utilisateurs sont associés au site :

Administrateur :

Il gère l'inscription des participants, sachant que l'inscription est publique
Il nomme les gestionnaires et leur attribue un espace
Il crée ou supprime des espaces

Il est gestionnaire de tous les espaces du site

Du côté **étudiant** :

Créer son profil :
Choisir un nom ou pseudonyme qui n'est pas déjà utilisé.
Choisir son niveau d'études parmi ceux proposés dans l'application.
Consulter les scores des autres joueurs pour pouvoir se comparer à eux.
Passer et sauvegarder son évolution dans la plateforme. La sauvegarde ne sera effective que sur le poste où elle a été effectuée et le sauvegarde du score seras enregistrés le dernier bon score.
Charger une partie. Le chargement d'une partie se fera par transparence lorsque l'utilisateur sélectionnera son profil.

Du côté **enseignants** :

Gérer les questions disponibles dans le jeu (implique de donner la possibilité de gérer les questions) :
Ajout/suppression de questions.
Modification de questions.
Filtrer les questions par critère (niveau d'étude, thème, etc.).
Consulter les statistiques d'un utilisateur.

Le site doit être équipé d'un moteur de recherche afin de faciliter la navigation.

Au niveau de la page d'accueil, un utilisateur doit pouvoir s'inscrire ou s'authentifier afin d'obtenir les fonctionnalités correspondant à son statut et un espace spécifique de publication lui sera proposé et associés.

Compétences requises :

- Utilisation du PHP MVC afin de faciliter les taches entre les membres du groupe.
- Utilisation d'un SGBD
- Réalisation des formulaires Web (inscription …).

Priorités :

- Site d'apprentissage performant
- Une charte graphique homogéne.
- Outil simple de publication
- Interface claire et intuitive

3. Analyse des besoins

3.1 Contexte

Le projet HFGL sous le cadre du Génie Logiciel avancé mène à la réalisation d'une application avec une simulation graphique fonctionnelle d'une plateforme d'apprentissage. Autrement dit nous comptons programmer un site d'apprentissage permettant à chaque usager (étudiant) de rejoindre les cours proposé par un professeur aussi que de répondre aux questions / examens.

- modélisation des composants de l'application ;
- définition de la présentation des composants du site avec une interface graphique ergonomique dédié aux étudiants / professeurs ;
- Ajouter, Supprimer, Soumettre les cours et les devoirs

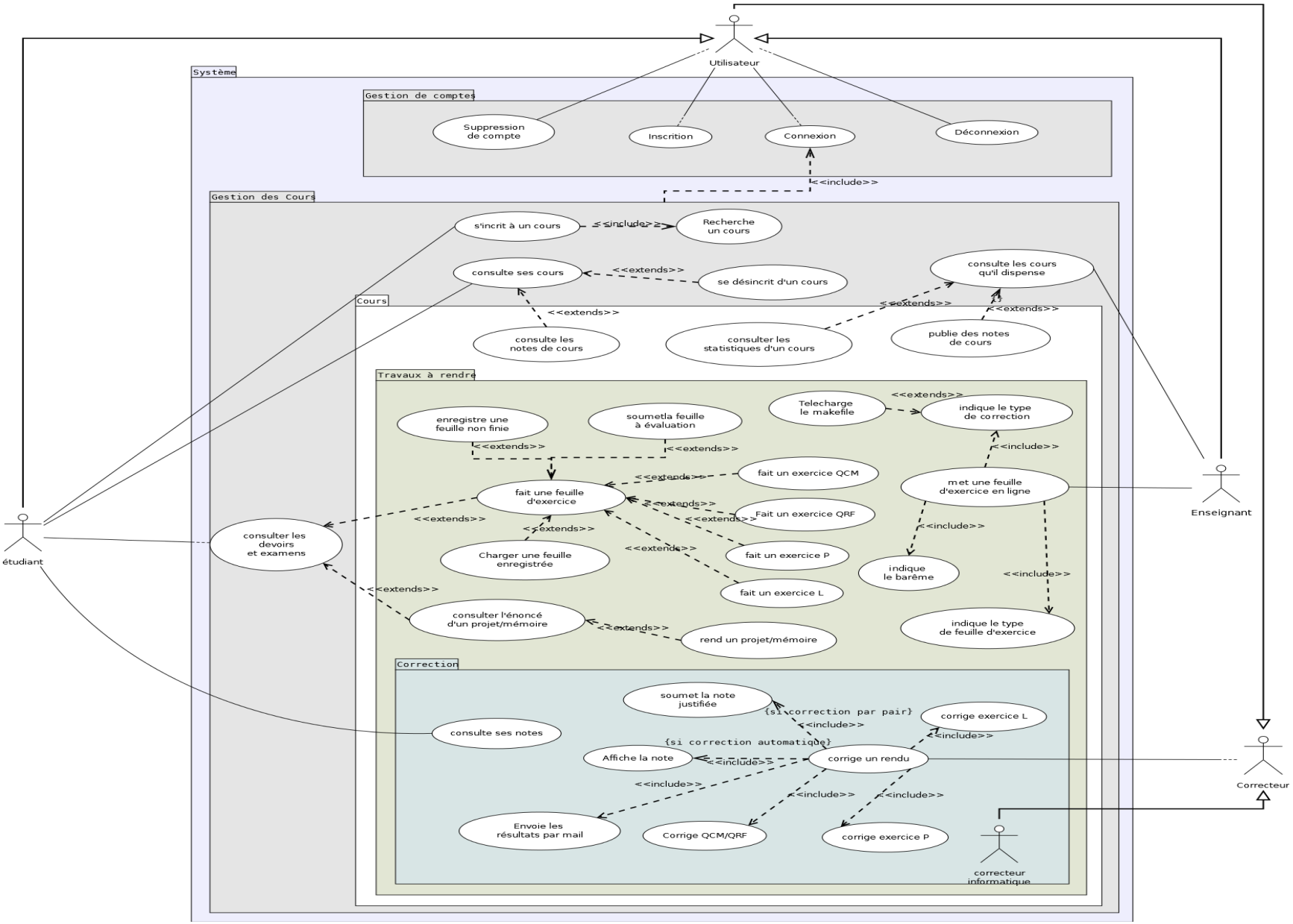
L'analyse de besoins permet d'identifier et de décrire précisément les besoins fonctionnels devant être satisfaits.
Dans ce document se trouvent :

- l'identification des besoins fonctionnels et non fonctionnels du système.
- l'identification des acteurs : identifie tous les acteurs susceptibles d'entrer en interaction avec le système.
- le diagramme de cas d'utilisation : permet de décrire le comportement prévu du système et de spécifier ses fonctionnalités.
- les prototypes du site : ce sont de brèves maquettes représentant globalement l'interface graphique du site.

3.2 Besoins fonctionnels

Cette section présente les cas d'utilisation les plus importants pour l'architecture du logiciel (cf. Modèle des cas d'utilisation 1.0). Les cas d'utilisation que nous avons retenus sont les suivants :
Le système doit permettre de :

- Créer un nouveau compte utilisateur.
- Connecter, Déconnecter, Supprimer un utilisateur
- Afficher une liste de cours correspondant aux critères définis par l'étudiant
- Inscrire, Désinscrire un étudiant à un cours
- Télécharger ou mettre en ligne les notes de cours d'un cours et mettre le barème
- Télécharger l'énoncé d'un projet ou d'un mémoire.
- Charger, enregistrée ou finir une feuille d'exercice
- Accéder à tous les devoirs, exercices et examens que l'étudiant doit faire
- Indique le type de feuille d'exercice et de correction
- Faire les exercices des différents types et sa soumission


Utilisateur
Système
Gestion de comptes
Suppression de compte
Inscrition
Connexion
Déconnexion
<<include>>
Gestion des Cours
s'incrit à un cours
<<include>>
Recherche un cours
consulte ses cours
<<extends>>
se désincrit d'un cours
consulte les cours qu'il dispense
Cours
<<extends>>
consulte les notes de cours
consulter les statistiques d'un cours
publie des notes de cours
Travaux à rendre
Telecharge le makefile
<<extends>>
indique le type de correction
enregistre une feuille non finie
soumetla feuille à évaluation
<<include>>
fait un exercice QCM
fait une feuille d'exercice
met une feuille d'exercice en ligne
Fait un exercice QRF
Enseignant
étudiant
consulter les devoirs et examens
Charger une feuille enregistrée
fait un exercice P
indique le barême
fait un exercice L
consulter l'énoncé d'un projet/mémoire
rend un projet/mémoire
indique le type de feuille d'exercice
Correction
soumet la note justifiée
{si correction par pair}
corrige exercice L
consulte ses notes
{si correction automatique}
Affiche la note
corrige un rendu
Correcteur
Envoie les résultats par mail
Corrige QCM/QRF
corrige exercice P
correcteur informatique

3.3 Besoins non fonctionnels

Cette section décrit les qualités et les contraintes importantes pour l'architecture du logiciel :

- **Performance :**

Temps de réponse : Le logiciel devra accomplir les fonctions demandées en un minimum de temps. Ainsi que d'avertir le client au cas où un problème de connexion sur le site ou lors de la soumission d'un projet ou examen.

- **Rapidité :**

L'application doit optimiser les traitements pour avoir un temps de génération de schéma raisonnable.

- **Efficacité:**

L'application doit être fonctionnelle indépendamment de toutes circonstances pouvant entourer l'utilisateur.

- **Maintenabilité et scalabilité:**

Le code de l'application doit être lisible et compréhensible afin d'assurer son état évolutif et extensible par rapport aux besoins du marché.

- **Ergonomie et souplesse:**

L'application doit offrir une interface conviviale et ergonomique exploitable par l'utilisateur en envisageant toutes les interactions possibles à l'écran du support tenu.

3.4 Identification des acteurs

Le site étant dédié à une manifestation des étudiants et d'enseignant, il est nécessaire qu'il soit géré par un **administrateur**. Celui-ci aura la charge de gérer les inscriptions des participants sur le site. Il devra ensuite nommer les **étudiants** et les **enseignants** et leur attribuer un espace associé à chacun d'eux. Enfin, il aura la possibilité de créer ou supprimer des espaces.

Un **étudiant** est une personne qui s'est inscrit sur le site pour participer au cours créé par les **enseignants**.

3.5 Prototype d'interface

Ces prototypes ont été effectués dans le but de visualiser l'aspect global qu'aura le site web à la fin du développement.

Page d'accueil du site:


www.hfgl.fr/connexion
Bienvenue sur HFGL
Have fun, Good learning
S'inscrire
Nom d'utilisateur
Mot de passe
Connexion

4. Analyse architecture

Cette section décrit la structure statique de la vue logique de l'architecture, en montrant les composants, leurs interconnexions et les interfaces offertes par ces composants.

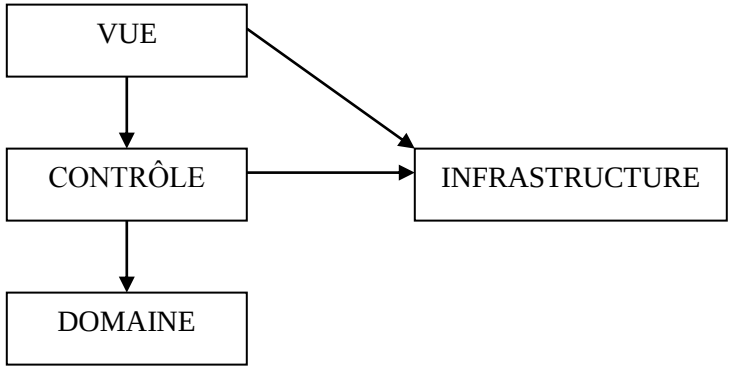
Il est primordiale à la conception de tout système informatique de choisir le modèle d'architecture qui lui sera adéquat pouvant assurer un bon fonctionnement, des meilleurs performances ainsi que la réutilisation et l'interconnexion fiable de ce système avec d'autres.

C'est à cet effet que nous optons pour le modèle MVC qui sera également très pratique pour gérer l'interaction entre les différents composants de notre application.

Nous décrivons cette architecture dans la section suivante.

4.1 Vue des couches

Le diagramme suivant représente les couches du logiciel. Nous expliquerons ensuite le rôle de chaque couche.



4.1.1 La couche Vue

Le rôle de cette couche est de fournir une vue à l'utilisateur pour qu'il puisse manipuler le logiciel. Elle va interagir avec la couche contrôle et va lui transmettre les données saisies et l'appeler en fonction des évènements déclenchés par l'utilisateur.
La couche vue ne gère que les aspects graphiques de l'application, c'est le niveau le plus haut dans l'architecture du logiciel et c'est la seule couche qui va interagir avec l'utilisateur :

Exemple : Vue « S'inscrire à un cours ».

4.1.2 La couche Contrôle

Le rôle de cette couche est de piloter les cas d'utilisation du logiciel. Chaque élément de la couche contrôle (contrôleur) est responsable d'un cas d'utilisation et c'est lui qui gère son fonctionnement. La couche contrôle agit sur 2 niveaux de l'architecture :

- Avec la couche Interface, afin de gérer l'Interface Homme-Machine
- Avec la couche Model, afin d'effectuer les traitements sur les objets métier de l'application, c'est-à-dire au niveau interne.

Exemple : Contrôleur « ControlleurProf ».

4.1.3 La couche Model

Le rôle de cette couche est de gérer les composants élémentaires de l'application, c'est-à-dire les objets Métier (ou entités), qui contient la logique métier.

5. Autres vues

5.1 Vue données

Cette section décrit les données persistantes du logiciel :

- La base de données en SQL.
- Génération du site Web projet (HTML/CSS/PHP MVC).

6. Concepts du domaine

Cette section décrit les concepts spécifiques du domaine et leurs relations.

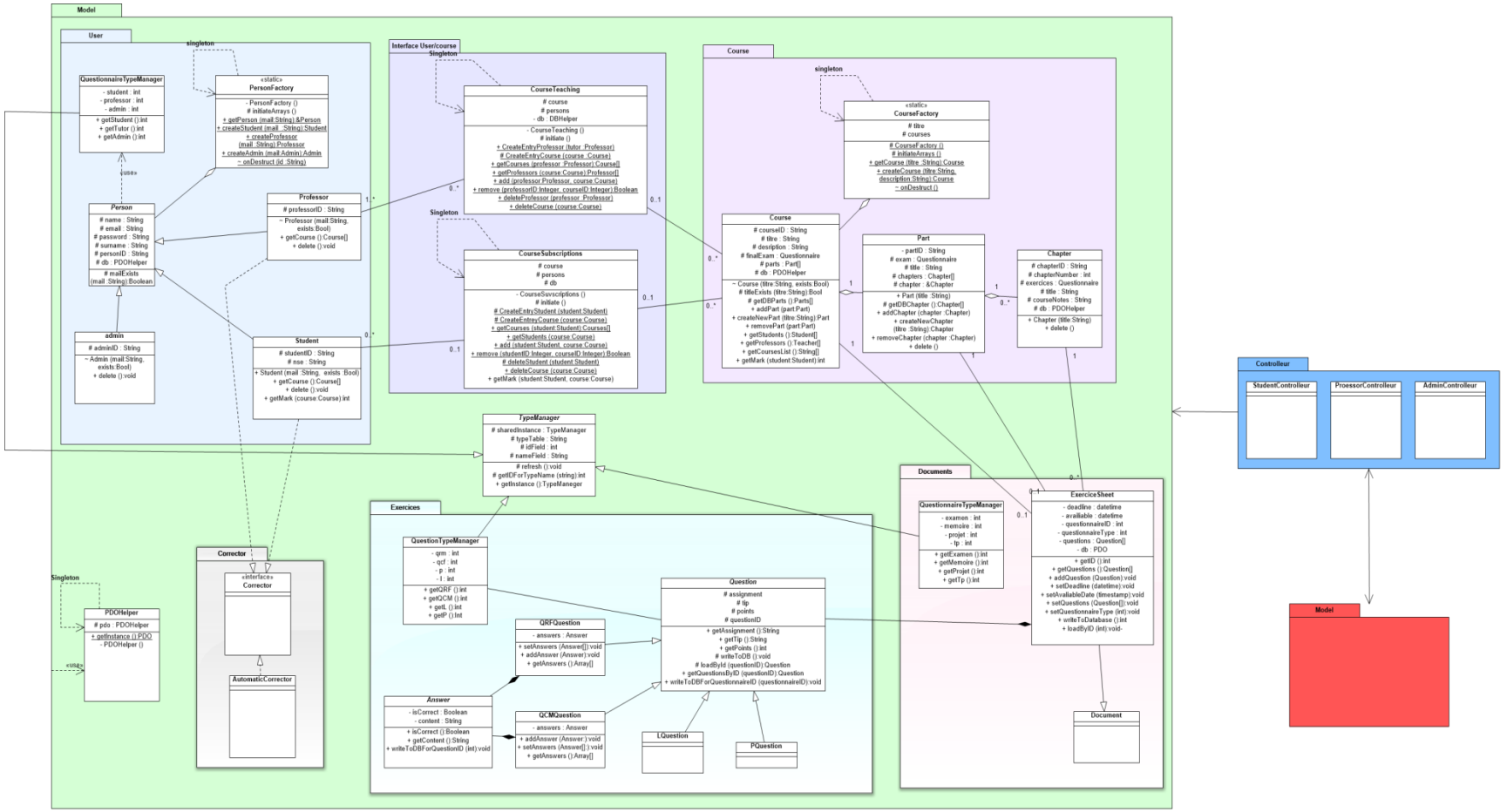


Figure 1 : diagramme de classe des éléments du domaine

7. Qualités de l'architecture

- **Avantages :**

L'utilisation d'une architecture en couche de type Interface/Contrôle/Model permet de décomposer de façon claire la réalisation d'un cas d'utilisation. Ce système permet de déléguer à chaque couche une responsabilité lors de l'utilisation d'une fonctionnalité du logiciel. Chacune d'elles communiquera avec une ou plusieurs couches sous-jacentes, ainsi que la séparation des données de la vue et du contrôleur (ce qui permet une conception claire et efficace de l'application), et de plus un des avantages c'est d'avoir une indépendance des données, de l'affichage et des actions (ce qui donne plus de souplesse pour la maintenabilité et l'évolutivité du système) , et enfin, Un gain de temps de maintenance et d'évolution de l'application.

- **Inconvénients :**

Les développeurs peuvent avoir des problèmes concernant la localisation de certaines structures de données, les accès entre composants de couches différentes…