UNIVERSITÉ DE NANTES

POLYTECH NANTES

Horasis Project

Conception Projet

Membres de l'équipe

Quentin LEVAVASSEUR

Nicolas GAILLARD

Anthony GRIFFON

Valentin BOUCHEVREAU

Client / Professeur tuteur

Mr. Marcus BARKOWSKY

Date de rendu du dossier de conception

Mardi 17 mai 2016

Département Informatique
Université de Nantes - Polytech Nantes

Avril 2016

Sommaire

I - Présentation des différents modules 3
I - Module des entrées/sorties 5
II - Module de la gestion de données internes et conversion 6
III - Module Interface Homme Machine 8
IV – Diagrammes de classes de Horasis 12
V - Prototypage de l'application 13
VI - Diagramme de Gantt 14

I - Présentation des différents modules

Notre application, Horasis, se découpe en trois modules distincts, le module Interaction Homme-Machine, le module entrées/sorties et le module des traitements effectués. Le module traitements effectués est composé d'une partie importante qui va interagir avec le module entrées/sorties comme on peut le voir sur ce diagramme.

Ces trois modules peuvent être implémentés parallèlement et ensuite intégrés les uns aux autres sans soucis.

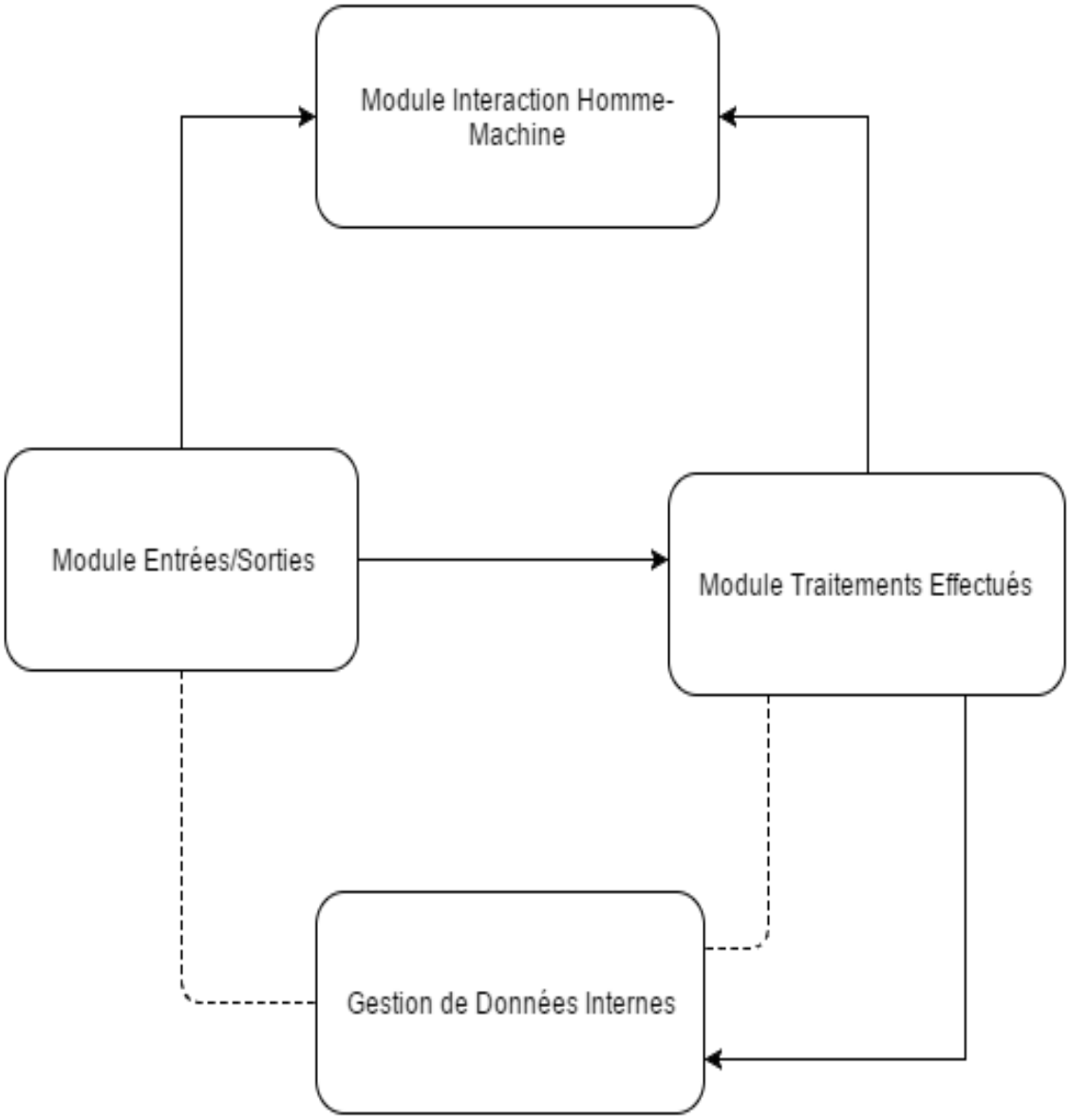


Diagramme des modules

Chaque flèche montre les interactions directes entre modules, tandis que les pointillés montrent comment les données transitent dans notre application.

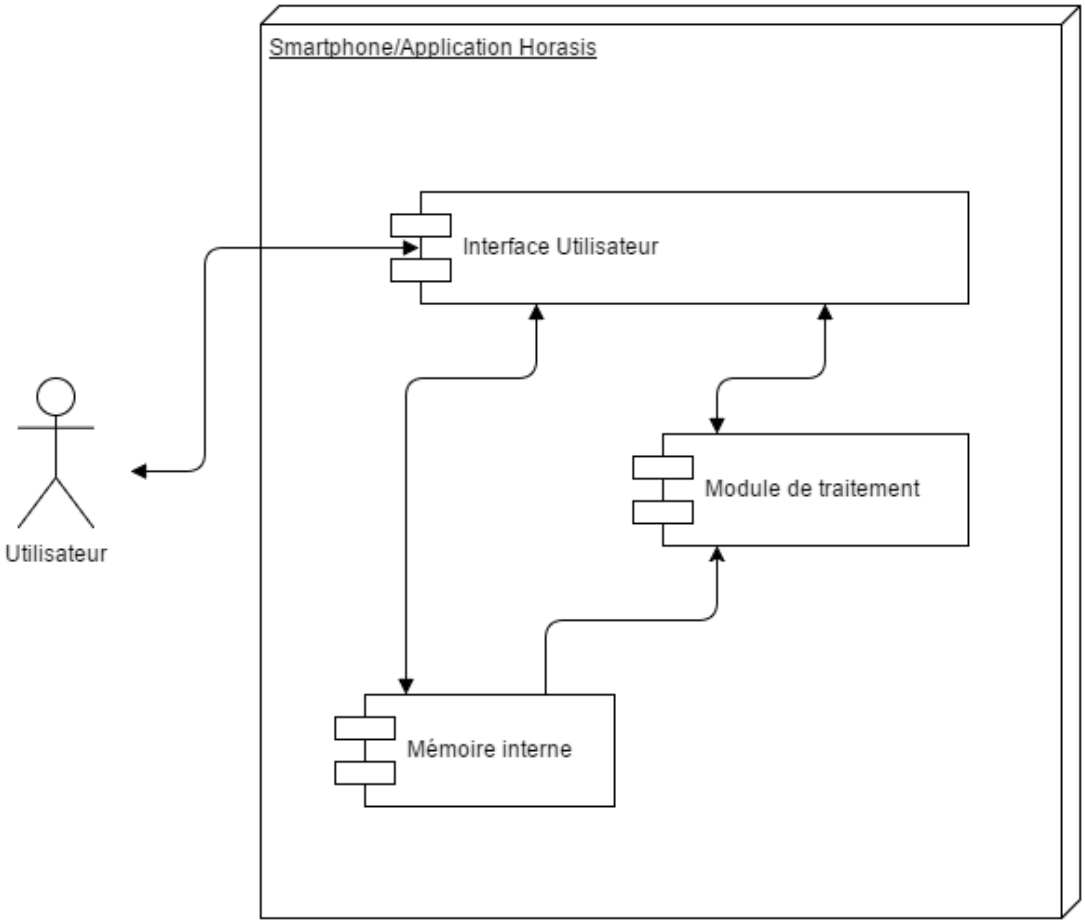


Diagramme de déploiement

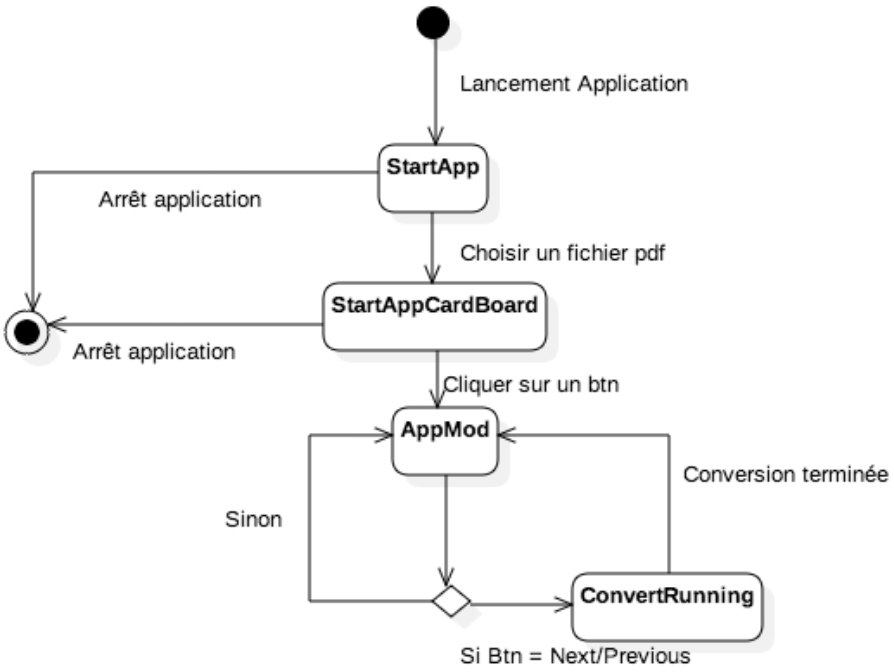


Diagramme d'états-transitions

I - Module des entrées/sorties

Horasis est un simple lecteur de document (au format PDF), pour ce faire, l'application ne nécessite pas d'un module pour gérer les sorties. En effet, notre application "ne produit rien", elle affiche seulement des documents via une paire de Google Cardboard.

L'application disposera de son propre file manager (gestionnaire de fichiers). Une fois le document sélectionné, l'application basculera automatiquement vers l'interface en réalité virtuelle. Il pourrait éventuellement filtrer tous les fichiers, afin de ne proposer que ceux au bon format (c'est à dire .pdf) mais aussi implémenter les solutions de stockage en ligne (Cloud Computing).

L'utilisateur interagira directement avec ce module. Il est également important de noter que l'usager devra choisir son fichier avant de disposer son téléphone dans la Cardboard et de le visionner en réalité virtuelle. Pour ce faire, l'affichage des différents fichiers se fera sous forme de menu déroulé, et l'usager aura alors à faire un simple clic pour ouvrir le document en VR.

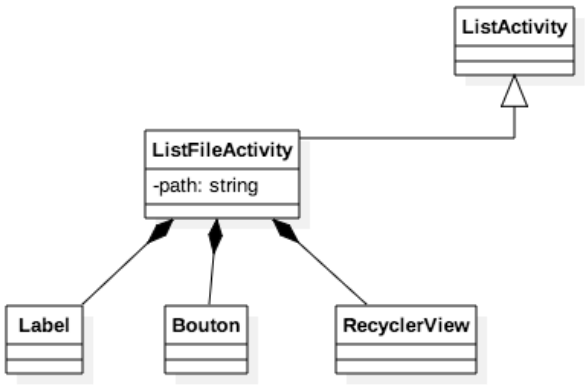


Diagramme de classes du module

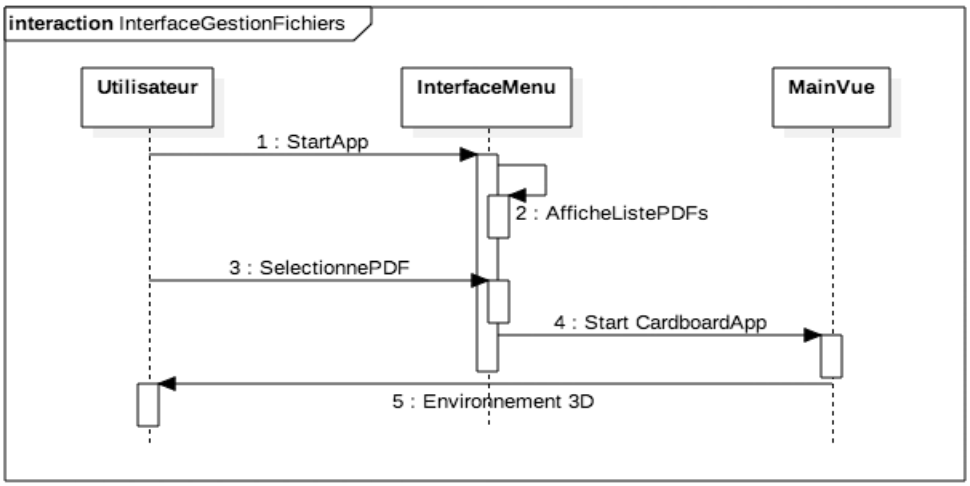


Diagramme de séquences du module

Estimation du temps :

- Temps : 6h/personne
- Nb de personnes : un binôme

II - Module de la gestion de données internes et conversion

Nous avons fait le choix de conception de rassembler ces deux modules pouvant être indépendant de notre projet en un unique module. Il s'agit ici de convertir la page courante en image. Une fois la conversion réalisée, il suffirait de disposer ces images sur des pages blanches modélisées en réalité virtuelle. Il faut aussi gérer la page courante, le nombre de pages, tourner les pages et gérer le pourcentage de zoom appliqué à la page.

Il existe aujourd'hui de nombreuses librairies capables de faire cela. Cependant, il est nécessaire de vérifier que la conversion ne modifie en aucun cas le layout du document (ie. on ne doit pas altérer l'affichage du fichier). Une longue période de test des différentes librairies est donc indispensable afin d'obtenir le résultat le plus optimal (c'est à dire une conversion qui ne dénature pas le document, où l'on pourrait zoomer sur le document sans perdre en lisibilité etc.). Un seul traitement est donc nécessaire pour Horasis. Pour ce qui est du zoom, un seul chargement de l'image serait nécessaire pour chaque niveau de zoom, la solution étant donc de rapprocher la texture pour zoomer sur le pdf et non de faire un zoom de manière conventionnelle.

Librairie utilisée: Pdf-Renderer (Licence: GNU Lesser General Public License v2.1)
La licence utilisée nous permet d'utiliser la librairie dans le cadre de notre application et de la vendre.

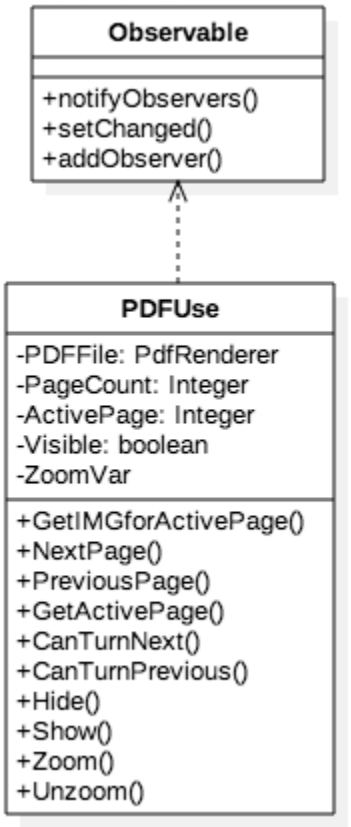


Diagramme de classes

Estimation du temps :

- Temps : 26h/personne
- Nb de personnes : un binôme

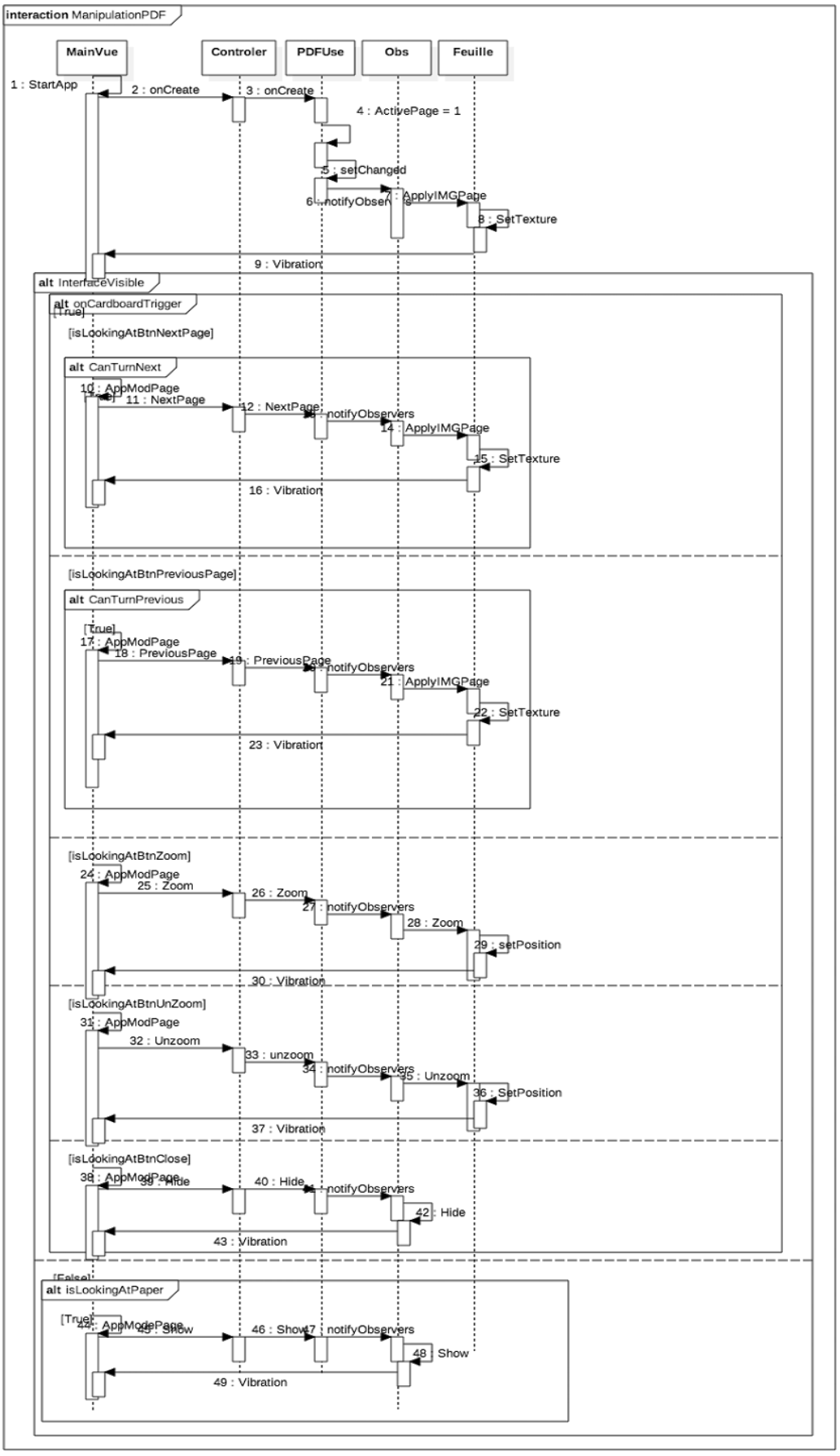


Diagramme de séquences

III - Module Interface Homme Machine

Pour représenter le module IHM, nous utilisons le patron d'architecture modèle-vue-controleur (ou MVC) afin de répondre aux besoins de la partie interactive de l'application.

Le modèle est la partie qui gère la manipulation de données. Nous avons élaboré une classe nommée *PDFUse*, qui possède l'ensemble des ressources nécessaires au bon traitement des données. Le détail de celle-ci est visible dans le schéma MVC plus bas dans le document.

La vue, quant à elle, retourne une représentation des données provenant du modèle. Ici, elle correspond tout simplement à notre environnement virtuel créé à partir du SDK Google Cardboard et la librairie graphique OpenGL.

Concrètement, la vue (ou l'environnement 3D) reconstituera un bureau en réalité virtuelle (pas seulement le meuble mais l'ensemble de la pièce). En effet, il est important de modéliser un espace vivant afin que l'usager ne soit pas mal à l'aise ; on parle alors de confort environnemental. Un espace trop restreint, même en réalité virtuelle, peut entraîner une certaine claustrophobie, tout comme un espace trop vaste qui lui engendrait une agoraphobie. Enfin, l'échelle du mobilier se devra d'être semblable à la réalité afin de ne pas avoir l'impression d'être intimidé, faible et petit. Sur ce bureau, sera présent une feuille (correspondant à une page d'un document) et des boutons afin de naviguer dans le document, ou bien zoomer.

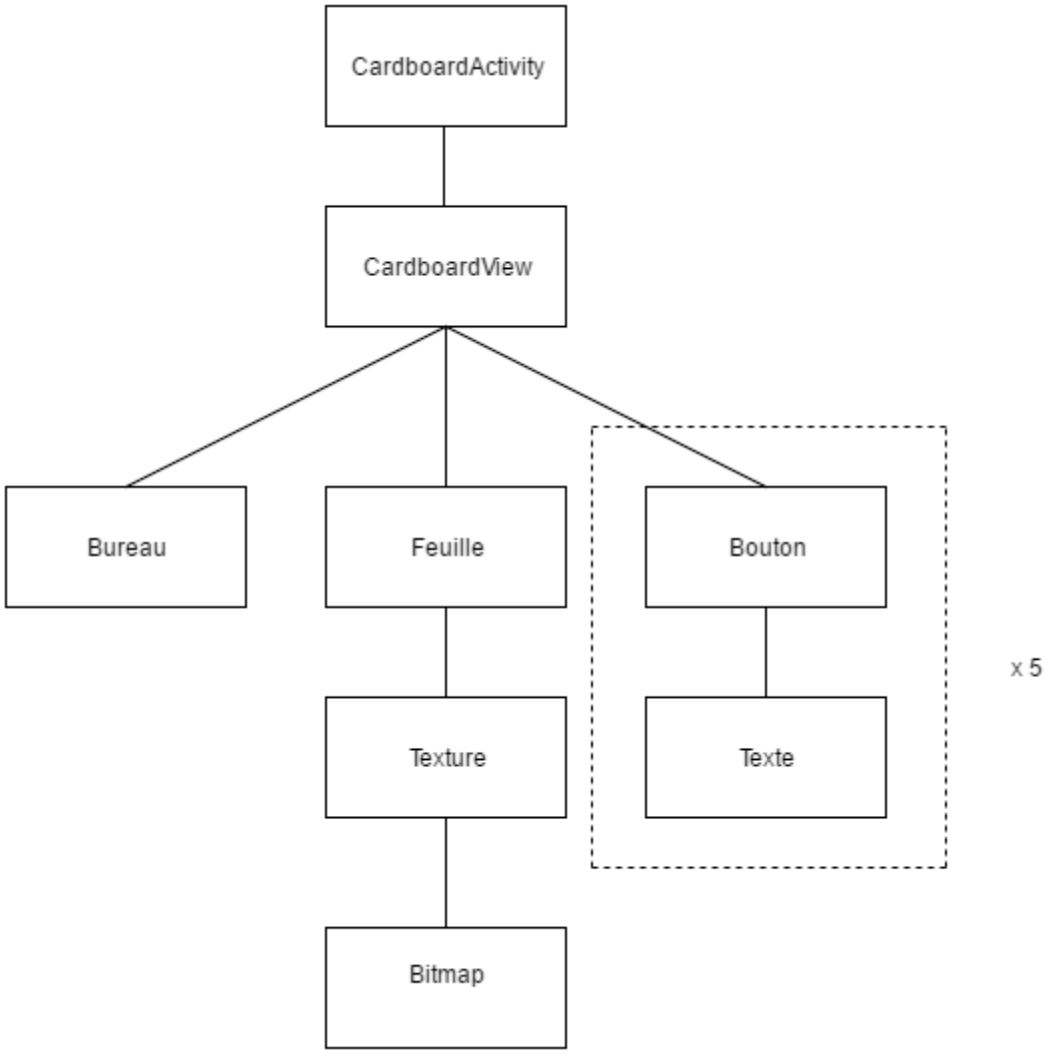


Schéma de la vue

Enfin, le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de la vue et enclenche les actions à effectuer.

Un de nos choix de conception est d'intégrer le contrôleur à la vue. Il regroupe l'ensemble des actions enclenché par l'appuie sur un bouton virtuel (qui modifieront le modèle) mais aussi des observateurs qui ont le rôle de modifier la vue dès que notre modèle est remanié. Il joue donc un rôle de tour opérateur, essentiel dans ce patron d'architecture.

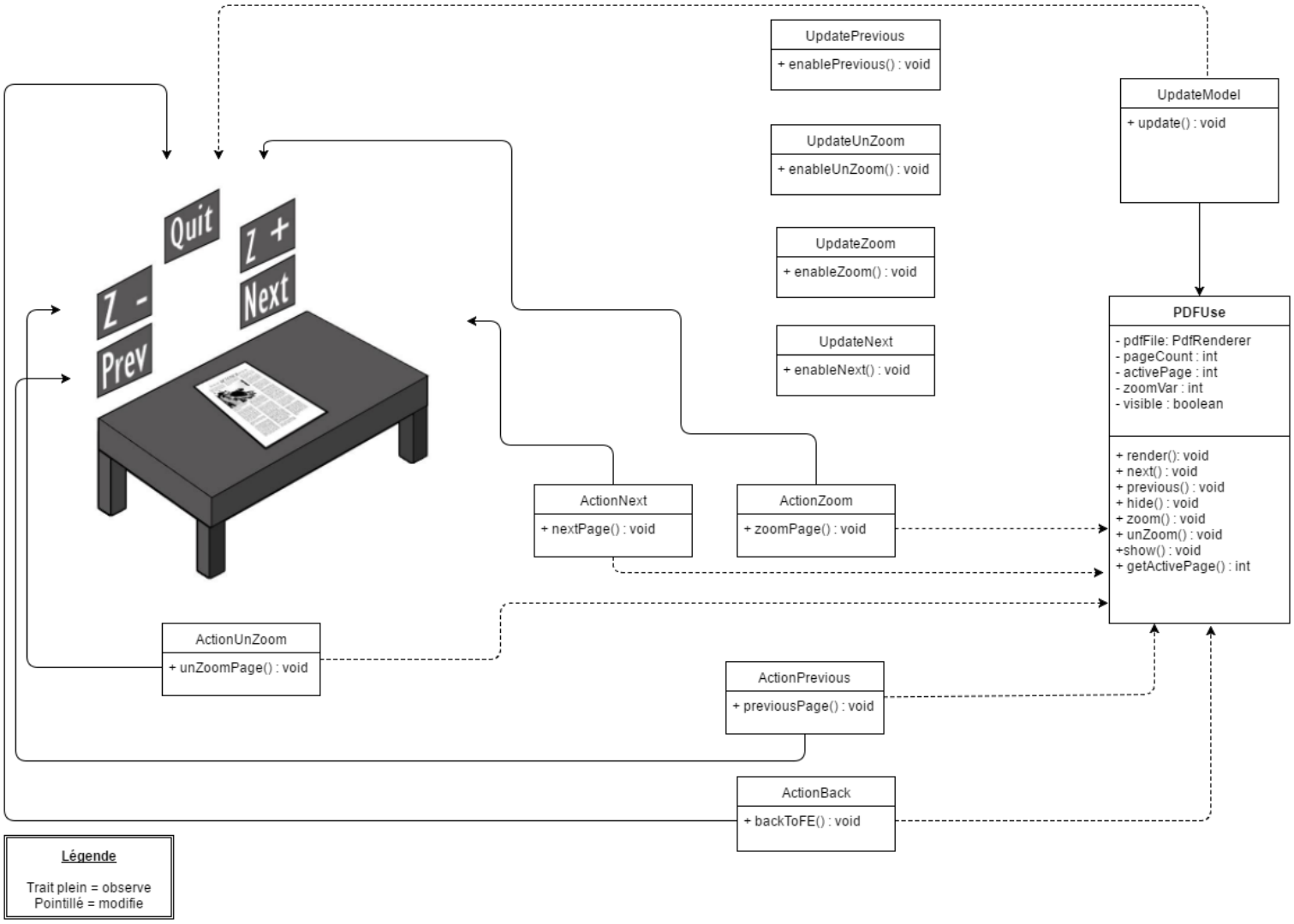


Schéma MVC

Pour des raisons de clarté, nous n'avons pas relié les updates des boutons. Chaque update observe le modèle (trait plein) et modifie la vue (pointillés) en grisant le bouton correspondant en fonctions des attributs de *PDFUse*.

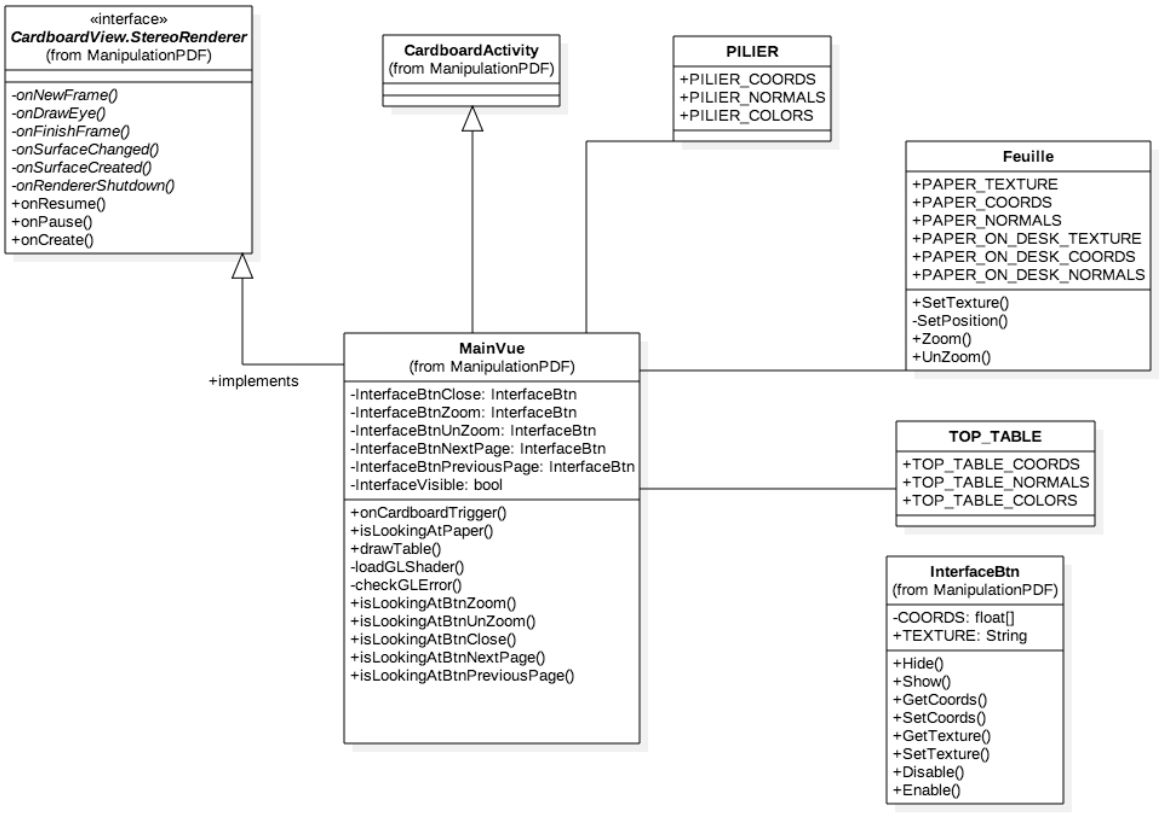
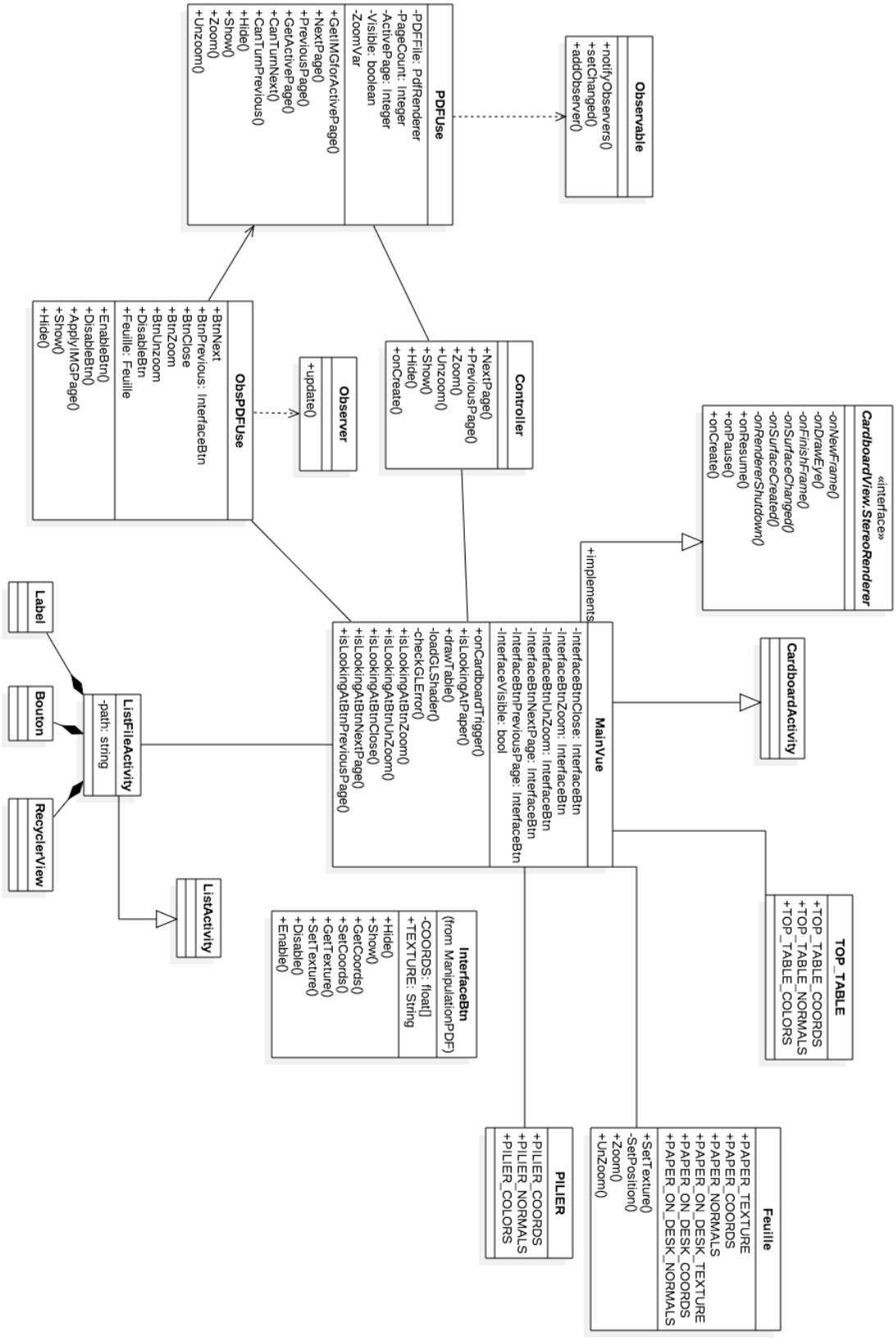


Diagramme de classes du module

Estimation du temps :

- Temps : 40h/personne
- Nombre de personnes : 2 personnes

IV – Diagrammes de classes de Horasis



V - Prototypage de l'application

Prototype n°1 : File Manager + Affichage table :

- Il s'agit du tout premier prototype qui nous a permis de découvrir l'affichage 3D ainsi que les bases du File Manager sous Android. Ce prototype permet donc de parcourir l'arborescence des fichiers puis de basculer en VR afin d'afficher une table en 3D (constituée d'un socle et de 4 pieds).
- Ce prototype est déjà disponible.

Prototype n°2 : Conversion d'un PDF :

- Ce prototype se doit de convertir au moins une page d'un document en un fichier image.
- Ce prototype est déjà disponible.

Prototype n°2.5 : Affichage d'un PDF sur une application Android (2D)

- Ce prototype nous permet d'afficher un document PDF sur une application Android, nous permettant ainsi de nous familiariser avec la manipulation de PDF. C'est un prototype secondaire.
- Il ne prend pas en compte la Cardboard et doit se contenter de transformer au moins une page du PDF en une image à afficher sur l'écran.
- Ce prototype sera disponible le 19 mai.

Prototype n°3 : Affichage d'une image sur une texture :

- Ce prototype se rapproche beaucoup plus de l'application que les trois précédents étant donné qu'il affiche correctement une image sur le bureau virtuel sans pour autant intégrer les fonctionnalités de zoom et de changement de page.
- Ce prototype sera disponible le 24 mai.

VI - Diagramme de Gantt

- Définir un ordre d'implémentation
- Rajouter les tâches de la consigne