TimeTableEasy - Rapport

Release 1.0

GREATTEAMDEVELOPMENT

June 13, 2010

CONTENTS

CHAPTER

ONE

INTRODUCTION

Auteurs

- Guillaume Espanel
- Alexis Metaireau
- Nicolas Morel
- Julien Grande

Version 1.0

Pour réaliser EasyTimeTable, nous avons choisi d'utiliser des technologies qui nous tenaient à coeur, et le mot d'ordre était l'apprentissage. Nous avons souhaité avant tout profiter de cette opportunité pour découvrir de nouvelles approches à la réalisation de sites internet.

C'est finalement avec un plaisir non dissimulé que nous pouvons vous présenter la première mouture d'EasyTimeTable. Cette version n'est sans doute pas parfaite, et comme tout logiciel, se devrait d'être en continuelle évolution, grâce aux différents retours des utilisateurs.

Nous avons cependant mis en place une architecture solide,qui pourra accueillir les futures évolutions du logiciel, et qui comporte d'ores et déjà les fonctionnalités les plus importantes (Gestion de l'emploi du temps, des personnes et des universités/campus).

CHAPTER

TWO

PRÉSENTATION DU PROJET ET RÉPARTITION DES TÂCHES

2.1 Composition du groupe

Notre groupe de travail s'est composé de quatre personnes, avec chacune ses spécialités :

2.1.1 Alexis Metaireau

ID Booster: 79309

Alexis s'est occupé principalement de:

- la mise en place de la structure du projet
- la formation et le support autour de Django
- la modélisation
- développement de la partie planning (avec Guillaume)
- mise en place du design

Alexis à été développeur PHP par le passé, puis a découvert Python et développé plusieurs sites avec Django et Python. Il est passionné par l'architecture logicielle, les bonnes pratiques et le monde du logiciel libre.

2.1.2 Guillaume Espanel

ID Booster: 54322

Guillaume s'est occupé principalement, lors du projet, de:

- Software design
- Modélisation
- Développement de la partie planning

Lors de ses expériences précédentes, Guillaume a été amené à travailler dans les domaines de l'administration système sous FreeBSD et GNU/Linux et a développé, à ces occasions, plusieurs outils en Python.

2.1.3 Julien Grande

ID Booster: 74710

Julien s'est occupé principalement de:

- Modélisation
- Déploiement
- Documentations d'utilisation & d'administration
- Développements mineurs

Julien est à la base administrateur systèmes et réseaux; cependant, ses expériences passées et sa formation l'ont amené à développer avec différents langages (PHP, ASP, ...)et sur différentes plates-formes (UNIX, GNU/Linux, Windows).

Ce projet étant son premier contact avec le langage Python et le concept MVT, son activité principale a plutôt été la partie administration/déploiement/installation.

2.1.4 Nicolas Morel

ID Booster: 63208

Nicolas à travaillé sur les thématiques suivantes:

- Software design
- Modélisation
- Développement de l'interface d'administration (CRUD)
- Implémentation du menu
- Rédaction de documentation

Avant le projet Nicolas s'est principalement intéressé à la conception et au développement web assisté par des frameworks. Ceci est sa première expérience avec le framework Python Django et le modèle MVT.

CHAPTER

THREE

ORGANISATION ET DIFFICULTÉS RENCONTRÉES

3.1 Organisation

3.1.1 Attribution des rôles

Le seul rôle spécifique que nous ayons attribué était celui de chef de projet, qui est allé à Alexis en raison de son expérience sur des projets similaires.

Par ailleurs, nous avons tous effectué les tâches selon ce qui nous intéressait. Nicolas à pu par exemple travailler sur la mise en place d'un système extensible de gestion d'éléments (CRUD) alors que Guillaume s'occupait de la gestion du calendrier. Nous avons tous travaillé sur quasiment l'ensemble des fonctionnalités du logiciel, lorsque nous voyions des modifications à y apporter.

3.1.2 Répartition des tâches

Travaux centraux

Notre équipe s'est réunie très tôt pour décider du déroulement du projet. Lors de ces premières réunions, et après une première ébauche réalisée en commun du modèle de données, nous avons décidé de diviser le projet selon les 4 ressources que nous avons manipulé :

- Les évènements
- La pédagogie
- Les espaces
- Les classes

La gestion des évènements a été confiée principalement à Alexis et Guillaume, celle de la pédagogie et des espaces à Julien et Nicolas.

Ces assignations n'ont bien évidement empêché personne de se pencher sur une autre ressource que la sienne, et chaque membre a eu l'occasion de travailler avec chacun des trois autres sur chaque partie.

Travaux périphériques

En plus des tâches directement relatives au métier, la réalisation de l'application a nécessité d'autres travaux relatifs à l'infrastructure et à la création de l'environnement de développement.

Ces tâches ont notamment été :

- La création d'un dépôt git [1] (sur github)
- La mise en place d'un squelette d'application
- La rédaction de la présente documentation

3.2 Difficultés

3.2.1 Design

La réalisation d'une telle application a nécessité une importante réflexion lors de sa conception.

L'ensemble de l'application reposant sur le modèle, toute l'équipe a participé à son écriture, et nous n'avons dû y apporter que des modifications mineures au cours du projet.

3.2.2 Javascript et Fullcalendar

Pour l'affichage du calendrier et des évènements, nous avons choisi d'utiliser la bibliothèque Fullcalender [2] dont il nous a été nécessaire d'appréhender le fonctionnement.

Nous nous sommes également heurtés à notre méconnaissance du Javascript, que nous avons surmonté, non sans difficultés (dates, par exemple).

3.2.3 Division du groupe

Il a été de plus en plus difficile pour notre équipe de dégager du temps en commun pour des réunions d'avancement. Nous avons cependant réussi à pallier ce problème grâce à Mumble [3] et au fait que nous étions suffisamment avancés pour permettre un développement relativement indépendant des différentes parties de l'application.

[1] Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, le créateur du noyau Linux, et distribué sous la GNU GPL version 2. : http://git-scm.om

[2] Fullcalendar est une bibliothèque Javascript dédiée à l'affichage de calendriers : http://arshaw.com/fullcalendar/

[3] "Mumble is an open source, low-latency, high quality voice chat software primarily intended for use while gaming." : http://mumble.sourceforge.net/

CHAPTER

FOUR

PRÉSENTATION FONCTIONNELLE

4.1 Objectifs :

Dans le cadre de la gestion d'une ou plusieurs universités, il est nécessaire de mettre en place un système permettant la gestion des personnes, des évènements rattachés à ces personnes, des lieux et des éléments pédagogiques (cours, évaluations etc).

Il est également important de gérer les rôles de ces membres, afin de permettre une délégation efficace de la direction de chaque campus.

Pour des raisons évidentes, il est important que ce logiciel soit facile d'accès afin que les membres de chaque université soient informés et aient un accès au informations qui les concernent.

4.2 Résultats attendus :

Pour répondre à ces objectifs, TIMETABLEASY, un site internet accessible depuis tous les navigateurs permet à un nombre illimité d'utilisateurs de visualiser et d'éditer en fonction de leur rôles les évènements qui les concernent sur leur calendrier.

TIMETABLEEASY permet également la gestion avancée des personnes, des lieux ainsi que des évènements rattachés à la gestion pédagogique.

4.3 Besoins / Liste des Fonctionnalités :

- Interface administrateur
- Gestion des utilisateurs du logiciel
- Gestion des cursus
- Gestion des campus
- Gestion des classes
- Gestion des étudiants inscrits
- Gestion et vue du planning de l'université
- Gestion et vue du planning d'un campus
- Gestion et vue du planning d'une période d'étude d'un cursus
- Gestion et vue du planning d'une classe

- Gestion et vue du planning d'un utilisateur
- Gestion des informations liés à un événement
- Affichage des informations liés à un évènement
- Affichage des plannings
- Prise en charge du format iCal
- Gestion éventuelle de salles pour chaque événement
- Accès au site internet depuis un client mobile

CHAPTER

FIVE

ANALYSE ET MODÉLISATION

Voici l'analyse que nous avons effectuée grâce au cahier des charges fourni par le client.

Les schémas ci-dessous utilisent la notation UML (Unified Modeling Langage), qui est une manière de se représenter les problématiques logicielles.

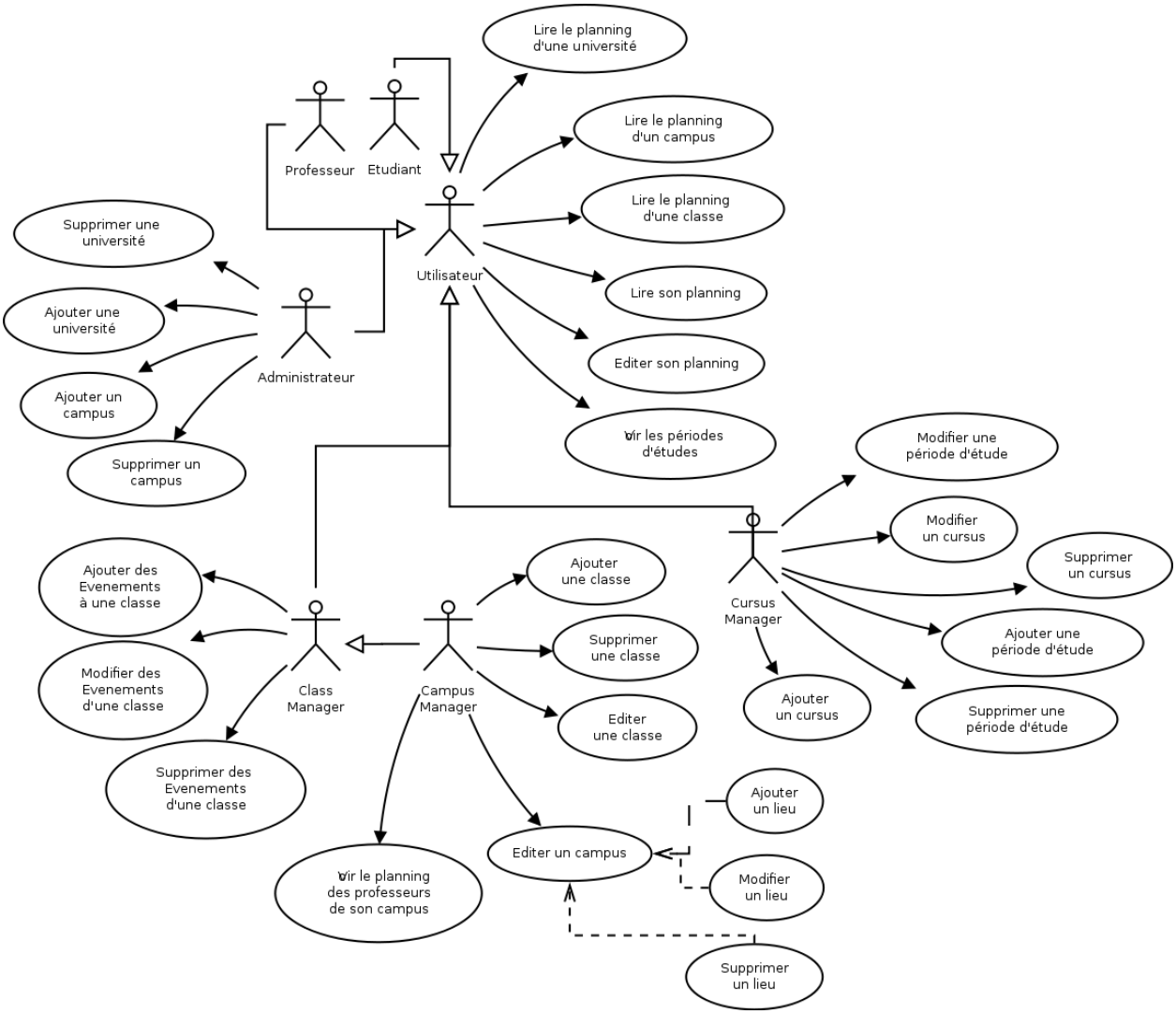
5.1 Analyse et compréhension du besoin métier

Pour commencer, nous avons construit une mind map, après lecture de l'ensemble des documentations fournies par le client. L'idée ici était de mettre à plat notre compréhension des besoins utilisateurs, pour ensuite travailler sur la modélisation du besoin. Vous pouvez trouver le résultat sur la page suivante.

5.2 Différents cas d'utilisation

Voici les différents cas d'utilisations que nous avons recensés, et qui représentent la base de notre projet.

Au vu du nombre de cas d'utilisations, le diagramme est un peu surchargé, mais nous avons fait notre mieux pour qu'il puisse rester lisible.



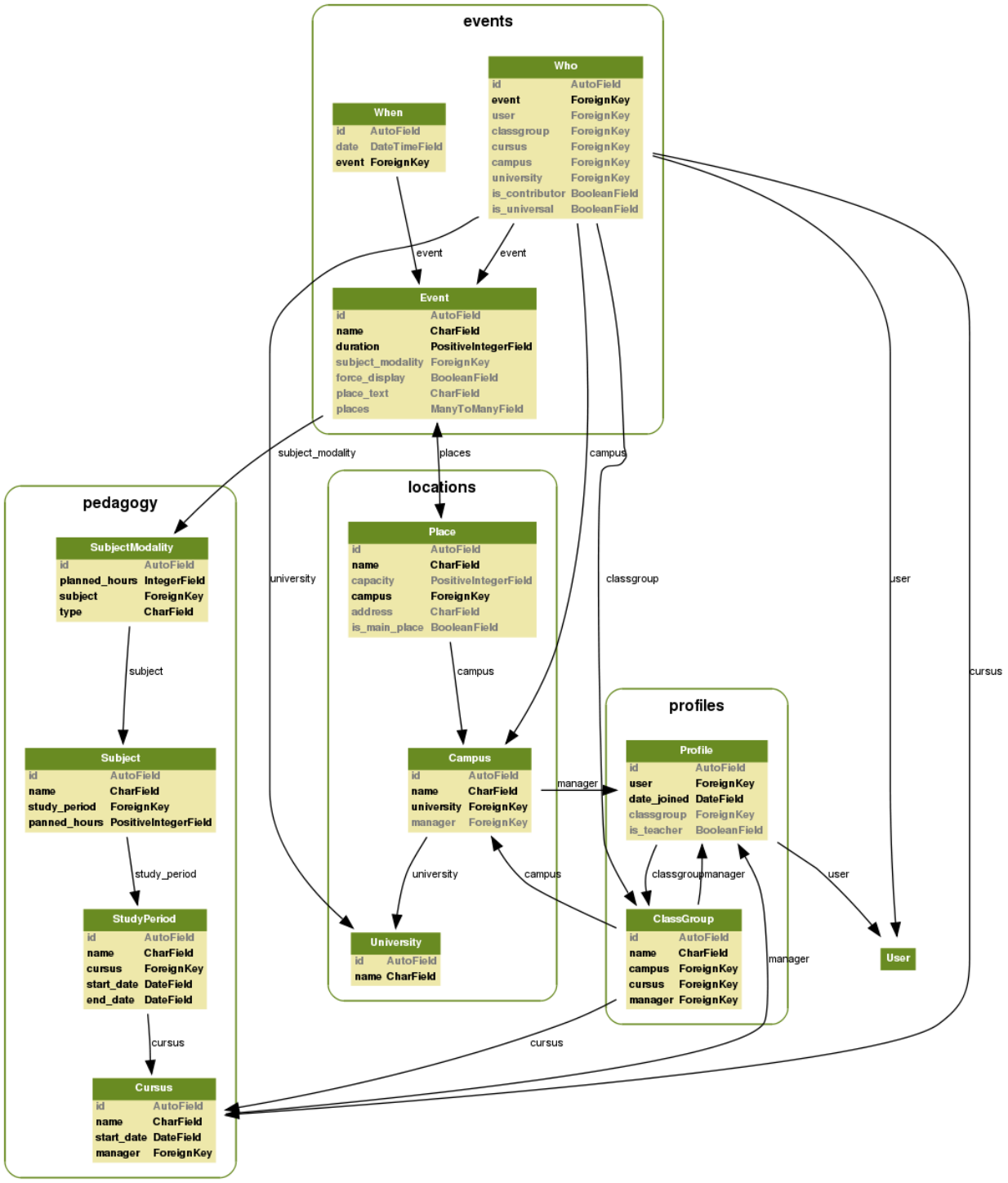
5.3 Diagrammes de classe, base de données

Nous utilisons Django, et celui ci propose des outils qui permettent de réfléchir au niveau du développeur et non pas des bases de données. En d'autres termes, il est possible de se concentrer sur la modélisation de nos objets, et le stockage de ceux-ci dans une base de données (relationnelle ou non) est automatiquement géré par le framework.

Il est bien évidemment possible de choisir une base de donnée particulière en fonction des besoins, mais il ne semblait pas y en avoir ici, donc c'est un point que nous laissons à l'appréciation du client. Les solutions techniques existent et sont très facilement exploitables.

Nous n'avons donc pas réalisé de diagramme de modélisation des données, mais un diagramme des relations qui existent entre nos différentes classes qui font partie du modèle de données.

Les deux images ci dessous représentent de manière exhaustive ou non les relations entre les différentes données.



5.4 Architecture logicielle sous-jacente

Comme vous pouvez le remarquer, nous avons choisi de séparer le projet selon quatre types de ressources :

- les évènements

- la pédagogie
- les espaces
- les classes et profils

Avant d'expliquer plus en détail l'architecture que nous avons mis en place, il est nécessaire d'expliquer comment fonctionne Django.

5.4.1 Comment Django fonctionne-t-il ?

Un des avantages de s'appuyer sur un framework *full stack* est qu'il définit de manière très claire une architecture logicielle qui fonctionne, et qui à fait ses preuves.

Cela n'empêche aucunement de faire par soi même une architecture logicielle, mais permet de ne pas s'en soucier si cela n'est pas nécessaire. Pour présenter un peu les concepts que met en avant Django, voici une rapide description, qui permettra de mieux expliquer par la suite l'architecture logicielle que nous avons mise en place.

Model - View - Template

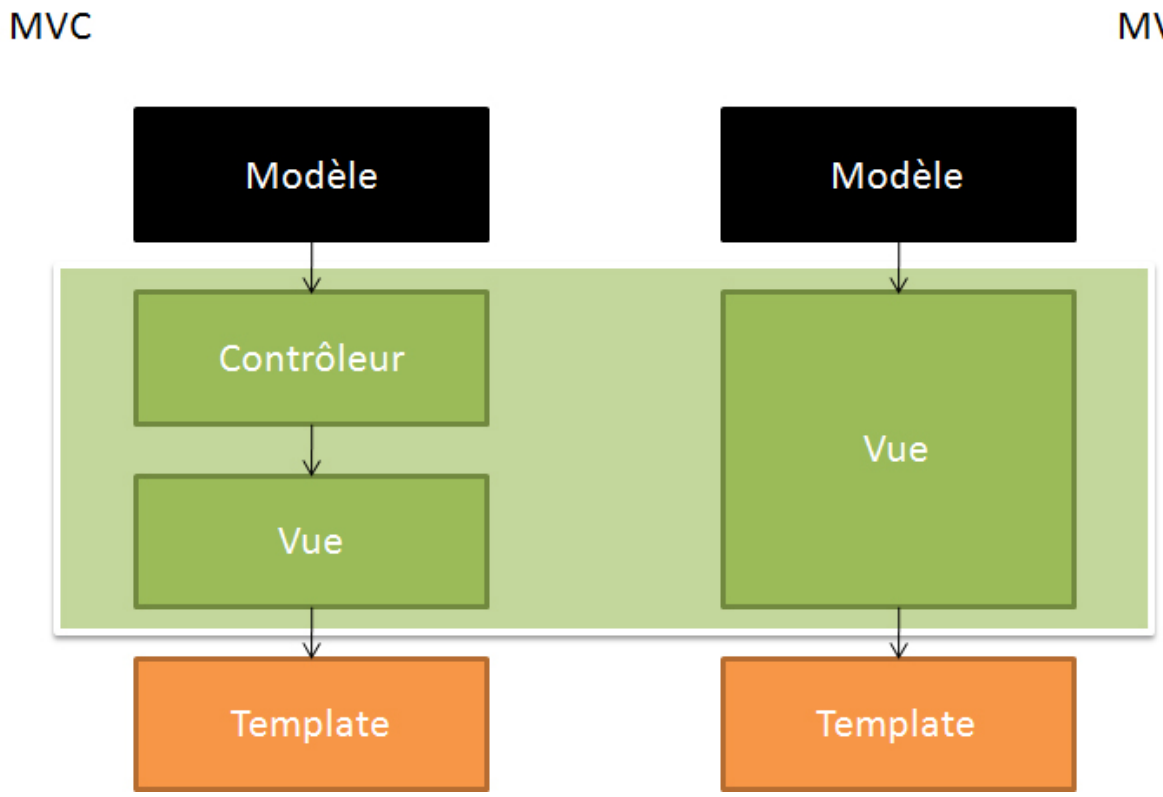
Le motif MVT, pour Model, Vue, Template, est très proche du modèle mieux connu : *MVC* (pour Modèle, Vue, Contrôleur). Alors que le motif *MVC* définit les couches comme suit:

- Modèle : Accès à la persistance des données. Ce sont les modèles qui se chargent de la relation base de donnée avec le monde objet (ORM), et qui idéalement (cela diffère selon les versions), s'occupent de la validation des données.
- Vue : La vue représente la partie IHM (Interface Homme Machine) de l'application. Dans un site web, c'est la vue qui s'occupe de transformer les données métier en HTML, par exemple. Une vue peut faire appel à des templates pour s'afficher.
- Contrôleur : C'est le composant qui se charge de faire la liaison entre le modèle et la vue. Il effectue les transformations nécessaires et contient peu de logique.

Le motif, *MVT* garde la même définition du modèle, mais il est nécessaire de comprendre les différences de notions de vues et de templates:

- Les Vues remplacent ici complètement les contrôleurs (MVC), et prennent également une partie des responsabilité des vues MVC. Ce sont elles qui passent les informations aux templates.
- Les templates sont des modèles d'affichage pour du HTML, qui est quasiment toujours utilisé dans le motif *MVC*.

Pour illustrer, voici un schéma qui permet de mieux saisir les différences:



Séparation des applications

Django encourage la séparation des applications, afin d'isoler les différentes logiques du projet. Nous avons donc choisi de séparer notre application en plusieurs applications.

La structure est la suivante:

```
events
pedagogy
locations
profiles
utils
```

Il sera par la suite possible d'ajouter facilement d'autres applications, qui auront pour rôle de faire d'autres choses que la logique liée à la gestion des agendas, par exemple pour ajouter des fonctionnalités CMS à l'application finale, d'ailleurs, Django propose des applications pour ce genre d'utilisations.

Managers

Les managers sont des objets qui s'occupent de construire les queryset, pour interagir avec l'ORM de Django. Dans l'exemple ci dessous, nous utilisons le manager par défaut de Django, fourni avec les modèles, sous le nom "objects":

```
>>> MyModel.objects.all()
[<MyModel "Model1">, <MyModel "Model2">]
```

Ici, nous avons renvoyé l'ensemble des objets MyModel. Il est possible d'étendre le manager que Django fournit par défaut, et de spécifier ses propres méthodes pour notre manager. Par exemple pour récupérer les emplois du temps relatifs à un utilisateur, on peut imaginer avoir une méthode *for_user*:

```
>>> Event.objects.for_user(user)
[<Event "Event #2">, <Event "Event #34">]
```

Les managers sont définis dans le module python *managers.py*. Si besoin, il sera possible par la suite de le découper en plusieurs fichiers.

Templates

Les templates servent sur-couche à du HTML, pour permettre une intégration facile avec Django. Ils sont situés selon le schéma suivant: *templates/appname/viewname/actionname*.html

Chacune des applications comporte ce qui lui est spécifique: les modèles, les formulaires, et les vues.

CHAPTER

SIX

CHOIX TECHNIQUES

Pour la réalisation de ce projet, nous nous appuyons sur la stabilité et les possibilités offertes par les logiciels libres.

"TimeTableEasy" étant particulièrement spécifique, afin d'accroître notre vitesse de développement, nous avons choisi de nous appuyer sur des composants déjà existants, ainsi que sur des frameworks.

Après une analyse du besoin et la modélisation des données, nous avons jugé que l'utilisation du langage Python nous permettrait d'aller à l'essentiel, et de gagner en efficacité. Associé au framework web Django, nous avons pu nous appuyer sur l'expérience accumulée de milliers de développeurs et d'utilisateurs à travers le monde, mais aussi de l'appui d'une communauté extrêmement active.

Afin de bien pouvoir comprendre nos choix techniques, nous allons vous présenter nos choix par rapport à la liste des fonctionnalités que nous proposons dans TIMETABLEEASY:

- Dans le cas de la gestion des utilisateurs nous avons utilisé django.contrib.auth (module du framework Django s'occupant de l'authentification) associé à un système personnalisé d'ACL (Access Control List).
- Pour l'interface d'administration permettant la gestion des cursus, des campus, des classes et des étudiants nous avons créé une application Django réutilisable, factorisant les fonctionnalités de création, modification, liste et suppression (CRUD) d'éléments.
- Pour la gestion des lieux et des campus nous avons utilisé l'API Google maps afin d'en offrir une représentation géographique.
- Pour les cursus, les périodes d'étude et les matières nous avons utilisé la bibliothèque graphique Javascript "HighCharts", permettant ainsi de mettre en valeur les informations qui nous semblaient pertinentes.
- Pour les gestion du planning, nous avons choisi de nous appuyer sur une extension au framework javascript jQuery nommée FullCalendar.
- Pour l'export au format iCal, nous utilisons une bibliothèque Python, vobject.

Il est possible d'héberger notre solution sur des machines dédiées, sur des Systèmes d'exploitations tels que Microsoft Windows, des Unixs (nous utilisons FreeBSD pour notre serveur de test) et GNU/Linux.

Nous préconisons l'utilisation de solutions BSD ou Linux pour le gain en coût que cela implique, et pour le cout de maintenance à venir. Il est également possible d'héberger notre solution sur le Cloud de Google ou sur Amazon EC2, si l'on souhaite pouvoir prévoir les futures montées en charge des serveurs.

Nous avons également utilisé, pour faciliter notre travail collaboratif, le système de gestionnaire de versions git.

6.1 GIT

Git est un gestionnaire de versions décentralisé, dont le principal avantage est de bien savoir "merger". Cela signifie qu'il est possible de travailler en même temps à plusieurs sur le même fichier, sans que cela pose énormément de

problèmes.

6.2 Python

Python est un langage simple, concis et précis, qui permet d'aller à l'essentiel.

C'est un langage de script qui est notamment utilisé pour des sites à fortes charges, tels que Google et YouTube par exemple. Un des gros avantages de python est qu'il est facile à prendre en main, et dispose d'une communauté très active, proposant très souvent des bibliothèques, en accès libre et gratuit, via le Python Package Index (PyPI).

Python est un language orienté objet qui favorise la programmation impérative structurée.

Python est un language qui permet de travailler plus efficacement, et d'intégrer des solutions de manière rapide. Quiconque peut apprendre à utiliser Python et ressentir des gains de productivité immédiats et de réduire ses coûts de maintenance.

Un autre avantage lié à l'utilisation de Python est qu'il fonctionne sur la plupart des plateformes informatiques, de Windows à Unix, en passant par Linux, MacOS ou même le JRE de Java ou le CLR de .NET.

6.3 Django

Il existe plusieurs frameworks Web écrits en python, et qui profitent des atouts de ce langage, nous avons avons choisi de nous baser sur Django, pour plusieurs raisons:

- Django est un framework *full stack*, ce qui signifie que l'ensemble des composants qui le composent interagissent de manière efficace.
- Au niveau des membres de l'équipe que nous avons choisi de constituer, 3 membres sur quatre avaient une expérience de Python, et souhaitaient utiliser cette opportunité pour découvrir Django et Python plus en profondeur
- De plus, il s'agit d'un outil qui à dores et déjà fait ses preuves, puisque de nombreux sites à forte charge l'utilisent.

Django est connu pour être un outil qui facilite la productivité des développeurs, et qui apporte énormément de solutions aux problèmes rencontrés de manière récurrente lors de la mise en place de sites web.

De plus, Django à son propre écosystème, et de nouvelles applications (au sens de modules que l'on peut ajouter ou enlever) voient le jour ou sont mises à jour toutes les heures.

6.4 HTML

Le HTML est sûrement la manière incontournable de présenter du contenu dans des pages web. Alors que nous aurions pu utiliser des technologies qui ajoutent une sur-couche à HTML (Flash, Silverlight), nous avons choisi d'utiliser le plein potentiel de HTML. Cela permet entres autres d'avoir un contenu accessible facilement, et de manière sémantique. Il nous est ainsi possible de changer la représentation de ces données uniquement en modifiant les feuilles de styles qui s'occupent du design de l'application.

6.5 Javascript / jQuery

Pour ce qui est de l'affichage des données, nous avons choisi d'utiliser le couple désormais connu HTML + Javascript. jQuery est un framework javascript qui permet de simplifier l'utilisation de javascript. jQuery propose également

une API qui permet de factoriser certaines utilisations courantes en javascript, par exemple pour ce qui concerne les requêtes dites “Ajax”, ou la manipulation du DOM.

Outre le fait que Javascript soit plus facile d’accès grâce à jQuery, celui ci est grandement enrichi par le fait qu’il s’agisse d’un logiciel libre, bénéficiant ainsi d’une communauté grandissante et active, qui propose chaque jour de nouveaux plugins, pour tous les usages que l’on peut imaginer, un peu à la manière de Django.

6.6 Blueprint CSS

CSS est un langage qui sert à décrire la présentation des documents HTML et XML. Un des principal défaut de CSS est relatif aux logiciels que l’on utilise pour naviguer sur internet. Chacun à ses spécificités, et cela rends difficile de créer des feuilles de style qui sont compatibles avec l’ensemble de ces navigateurs.

Blueprint CSS propose entres autres de résoudre ce problème, et apporte également un système de représentation des pages web en grilles. Il devient alors possible de s’abstraire des problématiques bas niveau et de travailler directement avec une représentation en grille.

6.7 JSON

JSON signifie “Javascript Object Notation”, et il s’agit d’un format de données textuel, qui est implémenté dans énormément de langages, et notamment Python et javascript sont capable de transformer des objets JSON en objets javascript ou python, et inversement.

6.8 FullCalendar

FullCalendar est un plugin jQuery qui permet d’afficher de manière simple des évènements au sein d’un calendrier. Il possède plusieurs vues (mois, semaine et jour), et permet la communication avec le format JSON.

6.9 Les APIs Google Maps

Afin de représenter les adresses dans notre logiciel, nous nous appuyons sur la très simple API Google Maps, qui nous permet d’afficher des images avec les adresses que nous souhaitons.

CHAPTER

SEVEN

COÛTS DE LA SOLUTION

Pour étudier le coût de notre application, nous avons pris en compte à la fois l'investissement humain et l'investissement logiciel (coût des licences etc.)

7.1 Investissement Humain :

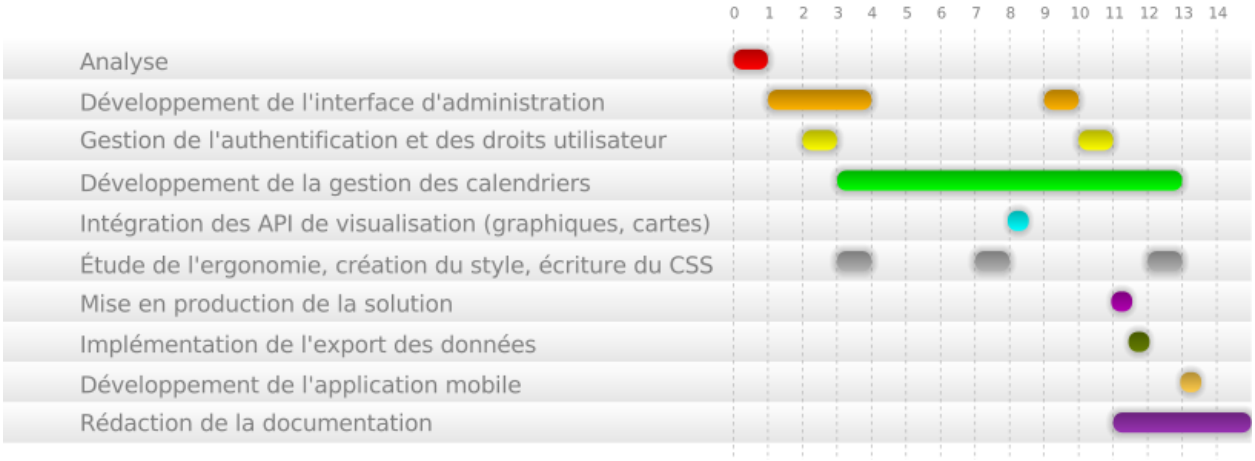
Coût en jour homme:

Coût (en jours homme)	Taches
4	Analyse(modélisation, mockups)
4	Développement de l'interface d'administration
2	Authentification, gestion des droits utilisateur
1/2	Affichage des graphiques, géolocalisation
4	Ergonomie du logiciel
20	Gestion du calendrier
5	Phase de tests, correction de bugs
1/2	Mise en production de la solution
4	Rédaction de la documentation
1/2	Application mobile
1/2	Export iCal

Total: **45 jours-homme**

L'équipe étant constitué de quatre personnes, nous nous sommes réparti le travail sur un total de 15 jours.

Vous pouvez vous référer au diagramme de GANTT pour plus d'informations



Pour évaluer les coûts, nous évaluons à 100 euros/jour le coût d'un développeur en entreprise.

Le développement de TIMETABLEEASY à donc eu un coût de "main d'oeuvre", de 40 (jours) * 100 (€) = 4300€.

7.2 Investissement Logiciel :

Nous nous sommes appuyé uniquement sur des outils open source pour la * réalisation de TIMETABLEEASY, ce qui fixe le coût des licences à 0.

Par contre, il est nécessaire de prendre en compte le coût des serveurs utilisés pour déployer la solution.

7.3 Étude de la concurrence et fixation d'un prix pour TIMETABLEEASY:

Le premier concurrent à notre logiciel est hyperplanning. Bien implanté dans le monde de l'éducation française ils fixent les tarifs via un forfait annuel ou en vendant la licence pour l'utilisation du logiciel.

Trois types d'offre sont disponibles :

- Monoposte avec un seul poste possédant la capacité d'administrer le système de planning à 1140 euros HT par an ou 3406 euros HT pour une licence "à vie".
- Réseau local à 2376 euros HT à l'année, 7114 euros HT à vie
- Réseau local et internet à 3406 euros à l'année et 10204 euros HT à vie.

Pour pouvoir mettre en ligne les plannings, le client devra s'offrir un hébergement non compris dans l'offre elle même.

Au niveau des fonctionnalités, hyperplanning possède une fonction pour comptabiliser l'absence et la présence des élèves que nous ne possédons pas.

En revanche nous possédons une capacité à gérer plusieurs universités, la géolocalisation des différentes universités et autres lieux, la représentation graphique de nos informations (hors calendrier), et un plus large panel de types d'utilisateurs.

Étant donnée le faible demande, hyperplanning et ses concurrents répondant déjà à une bonne part du marché, et nous devons nous aligner sur les prix du marché.

À la lumière de ces informations et d'après les coûts induits par le développement de l'application, nous en fixerons le prix le de licence d'utilisation à 2000 euros hors-taxes, ce qui nous permettrait de rentabiliser rapidement notre investissement.

Chaque client sera libre de demander une personnalisation, ou l'ajout de modules complémentaires à un prix forfaitaire permettant au différents développeurs de maintenir et d'enrichir l'application.

L'hébergement n'est pas inclus dans le prix du logiciel. Nous proposons des forfaits qui permettent d'essayer le logiciel sur des courtes durées, pour ensuite s'engager.