|  |
| --- |
| UNIVERSITÉ TOULOUSE III PAUL SABATIER |
| Dossier de programmation |
| A1110 : Informatisation du processus d’organisation d’un triathlon (IronTask) |
| **BRESOLIN Maxime**  **HOUTMANN Hadrien**  **POUSSARD Sébastien**  **ROMERO Diego**  **TROTTET Julien** |
| 8 juin 2018 |
| |  | | --- | |  | |
| **Tuteur/Client : M Thierry Beltran** |
| *Résumé : Ce document décrit les méthodes de travail utilisées dans le cadre de notre projet. Ce document est à destination du client.* |

**Identification :**

Type de document : Document de conception

Emetteur : STEELCORP

Version : 1.0

**Auteurs :**

|  |  |  |
| --- | --- | --- |
| **Nom et prénom** | **Fonction** | **Organisme** |
| POUSSARD Sébastien | Responsable organisationnel | STEELCORP |
| ROMERO Diego | Responsable technique | STEELCORP |
| BRESSOLIN Maxime | Responsable communication | STEELCORP |
| HOUTMANN Hadrien | Responsable technique | STEELCORP |

**Vérification :**

|  |  |  |
| --- | --- | --- |
| **Nom et prénom** | **Fonction** | **Organisme** |
| BELTRAN Thierry | Tuteur | Equipe enseignante |

**Approbation :**

|  |  |  |
| --- | --- | --- |
| **Nom et prénom** | **Fonction** | **Organisme** |
| TROTTET Julien | Chef de projet | STEELCORP |

**Listes des version et révisions :**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Objet** |
| 1.0 | 21 mai 2018 | Version initiale |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table des matières**

[1. Introduction 5](#_Toc515272922)

[2. Rappel de l’analyse fonctionnelle 5](#_Toc515272923)

[1.1 Recherche des fonctions de services 5](#_Toc515272924)

[2.2 Caractérisation des fonctions 6](#_Toc515272925)

[3. Processus organisationnel 7](#_Toc515272926)

[1.1 Méthode de travail 7](#_Toc515272927)

[3.2.1 Mise en place de la méthode agile 7](#_Toc515272928)

[3.2.2 Présentation de l’équipe Agile 8](#_Toc515272929)

[3.2.3 Utilisation de la technologie GIT 8](#_Toc515272930)

[3.2.4 Communication 9](#_Toc515272931)

[3.3 Planification du projet 9](#_Toc515272932)

[3.3.1 Planification prévisionnelle 9](#_Toc515272933)

[3.3.2 Planification réelle 9](#_Toc515272934)

[4. Conception du logiciel 10](#_Toc515272935)

[1.1 Conception de la base de données 10](#_Toc515272936)

[4.2 Cadre de travail 11](#_Toc515272937)

[4.2.1 Choix des technologies 11](#_Toc515272938)

[4.2.2 Modèle MVT (Modèle Vue Template) 11](#_Toc515272939)

[4.3 Détails du récit utilisateur, « Ajouter un triathlon » 13](#_Toc515272940)

[4.3.1 Les Liens (URL) 13](#_Toc515272941)

[4.3.2 Le Modèle 13](#_Toc515272942)

[4.3.3 La Vue 15](#_Toc515272943)

[4.3.4 Le Template 15](#_Toc515272944)

[4.3.5 Les formulaires 15](#_Toc515272945)

[4.4 Test de robustesse 15](#_Toc515272946)

[5. Bilan 15](#_Toc515272947)

[1.1 Bilan technique 15](#_Toc515272948)

[5.2 Bilan humain 16](#_Toc515272949)

[6. Conclusion 16](#_Toc515272950)

# Introduction

L’équipe projet a été restructurée et est maintenant composée de 5 étudiants à l’IUT informatique (Année Spéciale). Il a alors été décidé de fusionner les cahiers des charges des différents groupes afin de répondre au mieux à la demande.

Les premiers rendez-vous avec le client, nous ont permis d’affiner les fonctions logicielles importantes et d’épurer celles qui n’étaient pas voulues par Monsieur BELTRAN.

Le groupe est chargé du développement du logiciel IRONTASK, qui permettra de faciliter l’organisation de triathlon et d’organiser les tâches inhérentes aux l’événements.

Ce document de conception décrit la structure du logiciel et ses fonctionnalités, l’architecture (base de données) et la programmation. Nous détaillons également les méthodes de travail utilisées qui ont un impact direct avec le client.

# Rappel de l’analyse fonctionnelle

Nous avons redéfini notre équipe projet. Nous sommes maintenant 5 étudiants, il a été donc décider de refaire l’analyse fonctionnelle du projet afin que nous ayons tous les mêmes fonctions contraintes.

A terme, nous avons traduit cette analyse fonctionnelle en récit utilisateur. Ces récits utilisateurs sont utilisés pour la méthode agile que nous détaillons plus loin.

## Recherche des fonctions de services

Nous avons recherché les fonctions du logiciel de gestion de triathlon. Le diagramme “pieuvre” met en évidence les relations entre les différents éléments du milieu environnant et le produit.

**FP1**

FC6

FC7,8

FC9

FC10

FC1,2,3,4,5

Figure – Diagramme Pieuvre IRONTASK

Voici la liste de la fonction principale et des fonctions contraintes en référence au diagramme pieuvre ci-dessus :

|  |  |
| --- | --- |
| Id. | Désignation |
| FP1 | **Permettre à l’organisateur de faciliter l’organisation d’un triathlon** |
| FC1 | Permettre à l’organisateur d’administrer les tâches d’un triathlon (ajout, suppression, …) |
| FC2 | Le produit doit être capable d’effectuer une recherche de tâches. |
| FC3 | Doit avoir un outil d’affichage des tâches dans le temps. |
| FC4 | Doit permettre de contacter les responsables de tâches. |
| FC5 | Alerter l’organisateur des tâches en retard. |
| FC6 | Répertorier et archiver les statistiques des participants sur le long terme (nb participants,…). |
| FC7 | Doit gérer les stocks des ressources matériels (Désignation, Quantité, Lieux, …) |
| FC8 | Doit stocker les devis et autres documents. |
| FC9 | S’adapter aux différents supports (PC, Smartphone, …). |
| FC10 | Respecter les normes de la FFTRI et CNIL. |

Tableau - Fonctions de services

## Caractérisation des fonctions

A présent, il est nécessaire de définir les fonctions de service (Tableau 2) avec un plus haut niveau de précision. Le tableau ci-dessous répertorie ces fonctions ainsi que le/les critères caractérisant ces fonctions en appliquant également un niveau et flexibilité pour les différents critères.

|  |  |  |  |
| --- | --- | --- | --- |
| Identifiant | Critères | Niveau | Flexibilité |
| FP1 | Gérer toutes les tâches  Planification | N tâches de bases  Calendrier année N | Ajout par FC1  -2 ans + 10 ans |
| FC1 | Ajouter/ Modifier /Supprimer une tâche | 1000 tâches | +- 100 |
| FC2 | Filtres de recherche | 5 filtres max |  |
| FC3 | Calendrier | -2ans/+5ans |  |
| FC4 | Contact par mail  Contact via messagerie interne |  |  |
| FC5 | Alerte automatique | X : paramètre tâche |  |
| FC6 | Statistique, nombres de participants  Durée d’archivage | 2 ans |  |
| FC7 | Gestion de stock (type, localité, quantité) | 2000 objets | +- 200 |
| FC8 | Stocker les devis | 512 Mo | <2048 Mo |
| FC9 | S’adapter à tous les écrans | Responsive design |  |
| FC10 | Normes FFTRI et CNIL | Normes FFTRI et CNIL | Normes FFTRI et CNIL |

Tableau - FONCTIONS DE SERVICES, CRITERES, NIVEAU ET FLEXIBILITE

# Processus organisationnel

## Méthode de travail

### Mise en place de la méthode agile

« Les méthodes agiles sont des groupes de pratiques de pilotage et de réalisation de projets. Elles ont pour origine le manifeste Agile, rédigé en 2001, qui consacre le terme « agile » pour référencer de multiples méthodes existantes.

Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles, impliquent au maximum le demandeur (client) et permettent une grande réactivité à ses demandes. Elles reposent sur un cycle de développement itératif, incrémental et adaptatif et doivent respecter quatre valeurs fondamentales déclinées en douze principes desquels découlent une base de pratiques, soit communes, soit complémentaires. » [[1]](#footnote-1)

Dans ce cadre nous avons utilisé le Framework SCRUM, ce Framework vise à régulièrement produire des livrables au client afin de pouvoir avoir un retour régulier. Ainsi le travail sur le projet à été découpé en différents sprints sanctionné par un livrable.

Pour l’organisation des Sprints nous avons utilisé Trello [[2]](#footnote-2). Chaque sprint étant d’une durée de 2 semaines, marqué par un test avec le client.

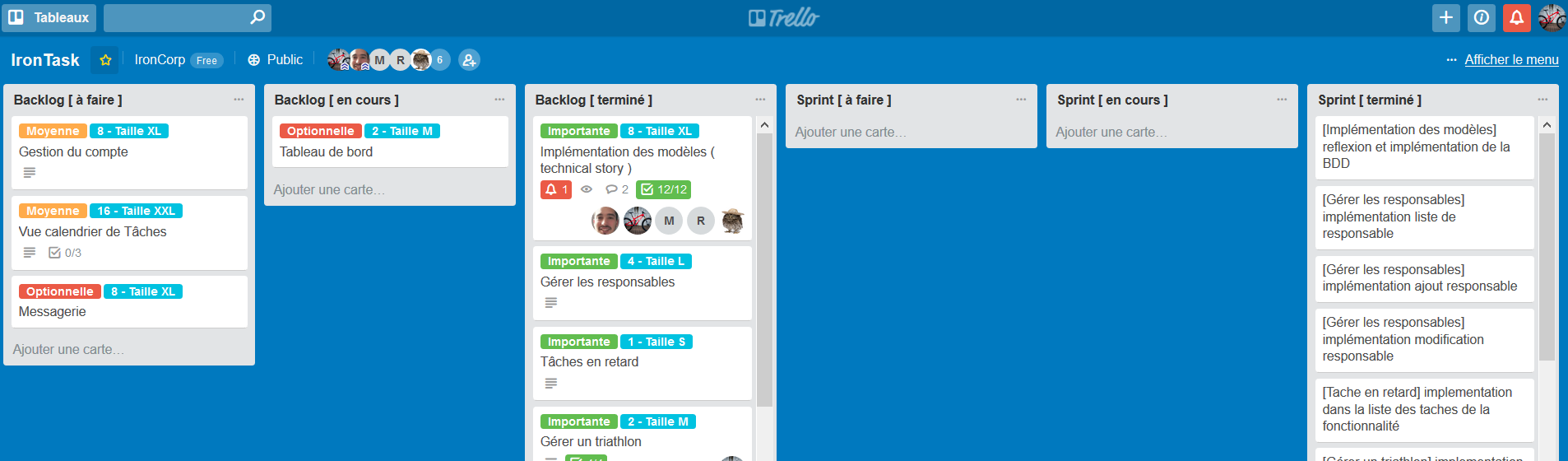


Figure - Organisation sous Trello

Lors du l’établissement du cahier des charges, nous avons définis les différentes fonctions que devrons respecter notre produit. A la mise en place de la méthode de travail, à savoir la méthode agile, nous avons définis, avec le client, le backlog du produit. Celui-ci est composé de plusieurs récits utilisateurs, que nous avons traduit en fonctions techniques à réaliser. La méthode agile veut que nous réalisions un récit utilisateur dans sa globalité. De cette manière, nous avons réussi à implémenter des actions complètement fonctionnelles que le client a pu tester lors de la fin d’un sprint.

Ce mode de fonctionnement est très avantageux pour le client car c’est une méthode itérative et nous avons un retour régulier du client. Nous, en tant que développeurs, sommes donc moins frustrés par les nouvelles modifications que le client exige.

Bien que notre équipe soit jeune, nous avons pu déterminer notre capacité de travail (ici 14). Ainsi pour un sprint de 15 jours, nous savons combien de récit utilisateur nous pouvons implanter. Un bilan technique détaillant les récits utilisateurs que nous avons réalisé ainsi que ceux restants est évoqué à la fin de ce document.

### Présentation de l’équipe Agile

Nous vous présentons notre équipe de développement sous le modèle agile. Pour rappel notre client est M Thierry Beltran.

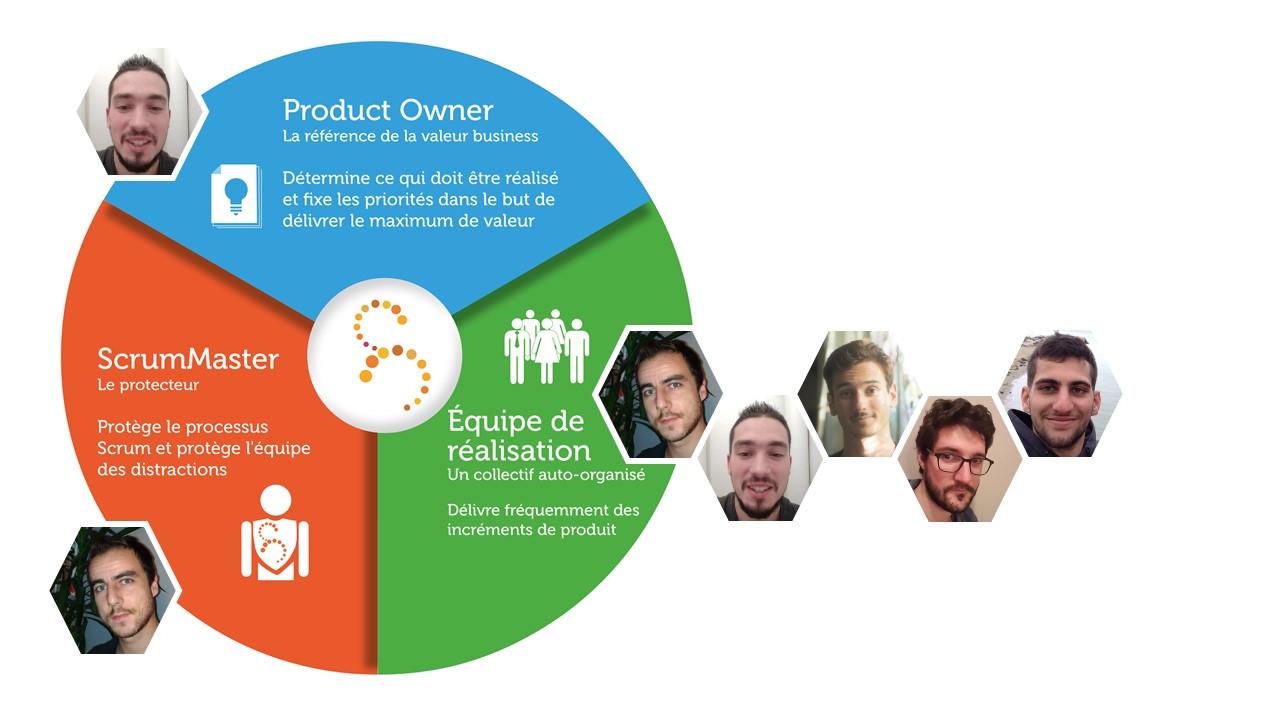


Figure - Présentation de l'équipe Agile

Au cours du projet, notre équipe a évolué de manière homogène grâce, en particulier, à Hadrien qui, ayant déjà développé sans le Framework Django , nous a monté progressivement en compétence lors de session « *extreme programming* »[[3]](#footnote-3) pour apprendre de nouvelles notions ensemble.

### Utilisation de la technologie GIT

Afin de développer à plusieurs sur le même projet il est nécessaire d’adopter une méthode d’organisation. Nous avons développé régulièrement en groupe le week-end car nous avons ressenti que nous progressions plus ensemble. Cependant en semaine l’équipe développait séparément. Dans ce cadre, il a été nécessaire d’adopter l’utilisation de GIT[[4]](#footnote-4) («git est un logiciel de gestion de versions décentralisé »[[5]](#footnote-5) ).

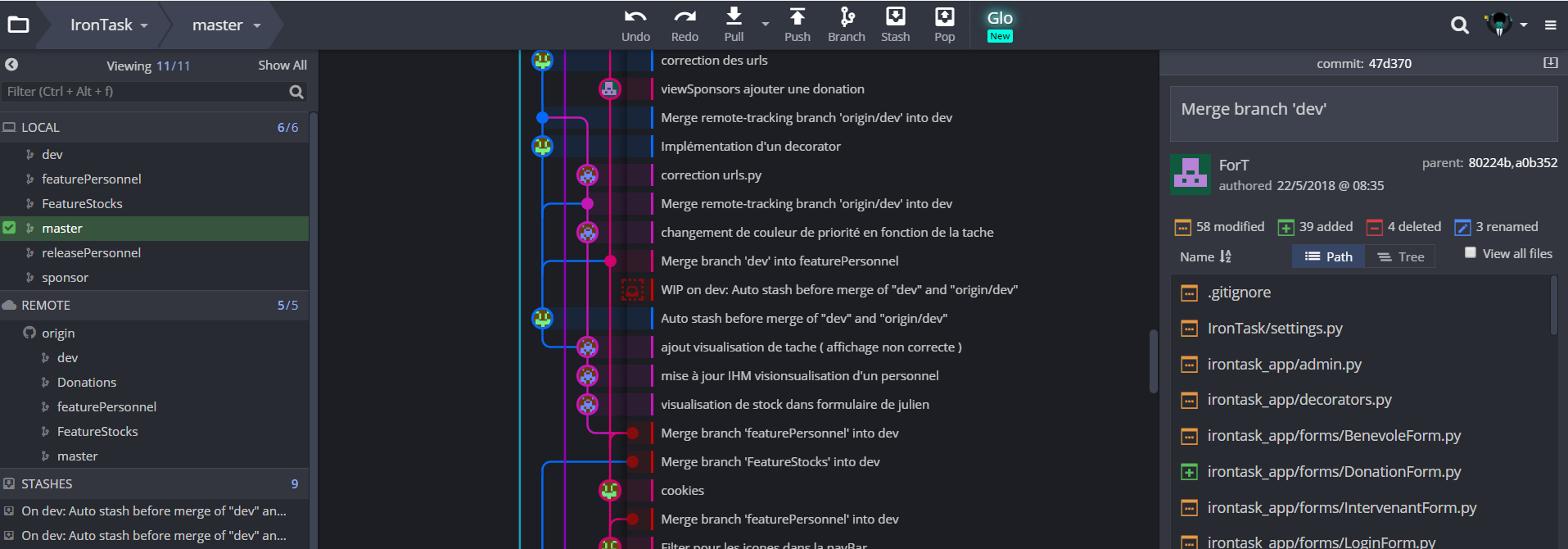


Figure - Organisation code sous Gitkraken

Nous avons poussé d’utilisation de Git au maximum en développant sur plusieurs branches. Nous avons défini 3 grand blocks (Tâche, Personnel et Stock) afin de découper notre projet. Ces blocks ont chacun deux branches (feature et release). Ainsi notre projet comporte les branches dev, master, featureTache, releaseTache, … Nous avons clairement identifié ces branches afin d’en avoir une définition commune pour éviter la confusion entre nous. Le tableau ci-dessous nous donne les procédures de « commit » que nous avons mis en place.

|  |  |  |
| --- | --- | --- |
| Type de branche | Qui valide ? | Fréquence |
| Dev | Tous | Très souvent (journalier) |
| Feature… | Tous | Souvent (hebdomadaire) |
| Release… | Scrum master et Product Owner | Rarement (un sprint, bimensuel) |
| Master | Scrum master et Product Owner | Très rarement (mensuel) |

Tableau - Procédure d'utilisation des branches

### Communication

En ce qui concerne la communication, nous avons utilisé le logiciel Slack et nous avons fait des réunions hebdomadaires afin de faire le parallèle avec la mêlée quotidienne du patron de travail Scrum de la méthode Agile.

Les nouvelles notions ont été apprise en programmant en groupe sur un seul ordinateur (extreme programming).

A la fin de chaque sprint, nous avons contacté M Beltran par mail afin que celui puisse tester les nouvelles fonctions implantées.

## Planification du projet

### Planification prévisionnelle

SEB + MAXIME

### Planification réelle

SEB + MAXIME

# Conception du logiciel

## Conception de la base de données

Figure : MCD de la base de donnée

Un Triathlon est caractérisé par une Type de triathlon afin de pouvoir réutiliser les différents types de triathlon (Ironman, long, short …).

Pour chaque triathlon peut intervenir des intervenants (médecins, animateurs …) étant donné que ces interventions peuvent être payantes nous avons choisis de stocker les Devis (sous forme binaire) et les prix des devis afin de l’afficher pour l’utilisateur.

A chaque triathlon sont associés également des sponsors qui vont faire des donations.

Chaque triathlon appartient également à une catégorie qui qui donne l’intervalle des âges ainsi que les sexes autorisés à participer.

Les taches seront associées à un triathlon et auront à chaque fois un responsable (qui est un bénévole) mais aussi une liste de bénévoles qui y participent.

Enfin nous enregistrons également le matériel disponible par l’association, ce matériel peut être dispatché dans différents triathlons via l’allocation notons qu’un bénévole gère ce matériel alloué, il en est le responsable.

La base de données utilise le format MySQL.

HAD

## Cadre de travail

### Choix des technologies

HADRIEN + DIEGO

Blbla apli lourde vs appli client server

#### Front-End

#### Back-End

### Modèle MVT (Modèle Vue Template)

Comme nous l’avons décidé, nous utilisons l'architecture Django qui diffère légèrement de l'architecture **MVC** (Modèle Vue Contrôleur) classique. Ce Framework gère automatiquement a partie Contrôleur. Nous nous occupons de l’architecture **MVT** : **M**odèle-**V**ue-**T**emplate.

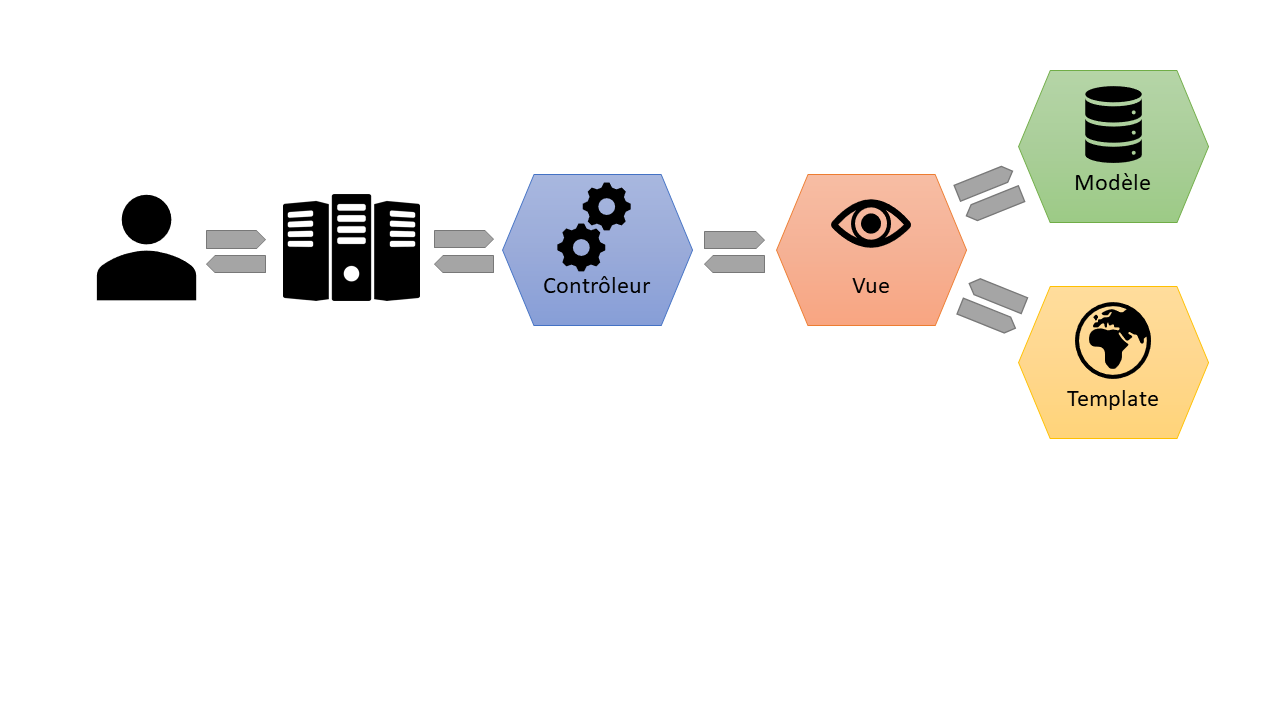


Figure - Modèle MVT (Modèle Vue Template)

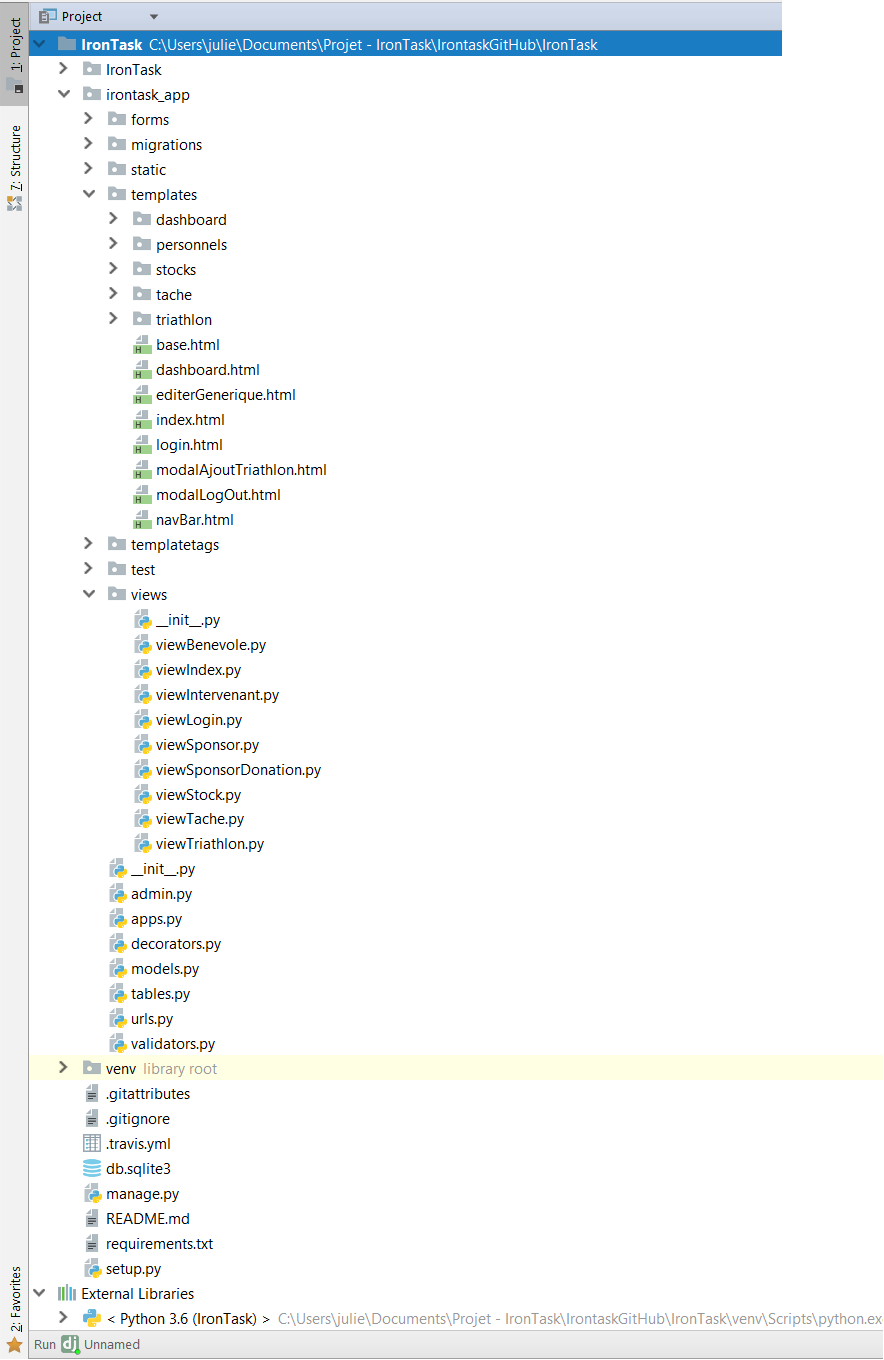
« Tout d'abord, **le modèle** *représente une information* enregistrée quelque part, le plus souvent dans une base de données. Il permet d'accéder à l'information, de la modifier, d'en ajouter une nouvelle, de vérifier que celle-ci correspond bien aux critères (on parle d'intégrité de l'information), de la mettre à jour, etc. Il s'agit d'une interface supplémentaire entre votre code et la base de données, mais qui simplifie grandement les choses, comme nous le verrons par la suite.

Ensuite **la vue** qui est, comme son nom l'indique, la *visualisation de l'information*. C'est la seule chose que l'utilisateur peut voir. Non seulement elle sert à présenter une donnée, mais elle permet aussi de *recueillir une éventuelle action* de l'utilisateur (un clic sur un lien, ou la soumission d'un formulaire par exemple). Typiquement, un exemple de vue est une page web, ni plus, ni moins.

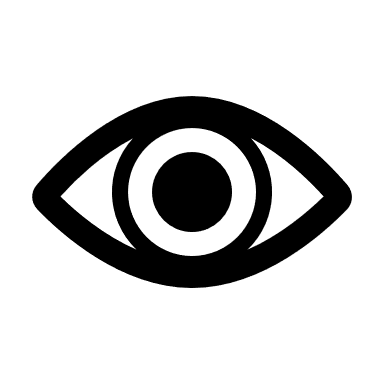
Enfin, le **Template** (voir figure suivante). Un Template est un fichier HTML, aussi appelé en français « gabarit ». Il sera récupéré par la vue et envoyé au visiteur ; cependant, avant d'être envoyé, il sera analysé et exécuté par le Framework, comme s'il s'agissait d'un fichier avec du code. Django fournit un moteur de Template qui permet, dans le code HTML, d'afficher des variables, d'utiliser des structures conditionnelles (if/else) ou encore des boucles (for), etc.

**Le contrôleur** *prend en charge tous les événements de l'utilisateur* (accès à une page, soumission d'un formulaire, etc.). Il se charge, en fonction de la requête de l'utilisateur, de récupérer les données voulues dans les modèles. Après un éventuel traitement sur ces données, il transmet ces données à la vue, afin qu'elle s'occupe de les afficher. Lors de l'appel d'une page, c'est le contrôleur qui est chargé en premier, afin de savoir ce qu'il est nécessaire d'afficher. »[[6]](#footnote-6)

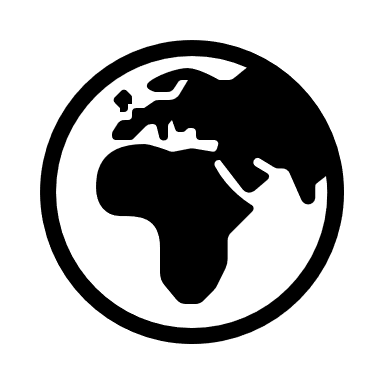
Nous avons développé sous le logiciel Pycharm. PyCharm est un environnement de développement intégré (EDI) utilisé pour programmer en Python. Il est développé par l'entreprise tchèque JetBrains.



Vue



Template



Modèle



Figure - Structure MVT sous Pycharm

## Détails du récit utilisateur, « Ajouter un triathlon »

Afin de bien expliquer notre projet d’un point de vue technique, nous nous focaliser sur un seul récit utilisateur, à savoir : « En tant qu’organisateur, je souhaite ajouter une tâche à un triathlon donné ». Ce récit sera détaillé dans chaque partie du modèle MVT comme présenté ci-dessus.

### Les Liens (URL)

MAXIME

### Le Modèle

Django est basé sur un ORM (Mapping objet-relationnel)[[7]](#footnote-7), qui correspond à une couche d’abstraction entre l’objet produit par l’ORM et la base de données relationnel.

L’ORM permet ainsi de manipuler la base de données à travers des objets, pour ce faire il est nécessaire de définir dans un fichier models.py les différentes tables de la base de données qui seront accessible à travers des objets dans les autres parties de Django.

Un modèle est donc une manière de décrire une table, pour ce faire il est nécessaire d’importer dans models.py différentes classes :

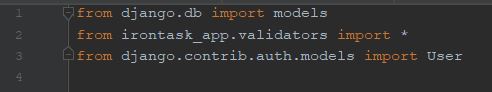


Figure : import de models.py

* Models de django.db est nécessaire pour pouvoir utiliser l’ORM et définir la base de données dans ce fichier.
* VALIDATORS ???
* User de django.contrib.auth.models permet de gérer de manière automatique des compte utilisateur capable de se connecter à l’application. En l’état actuel seul le compte administrateur est capable de se connecter à l’application

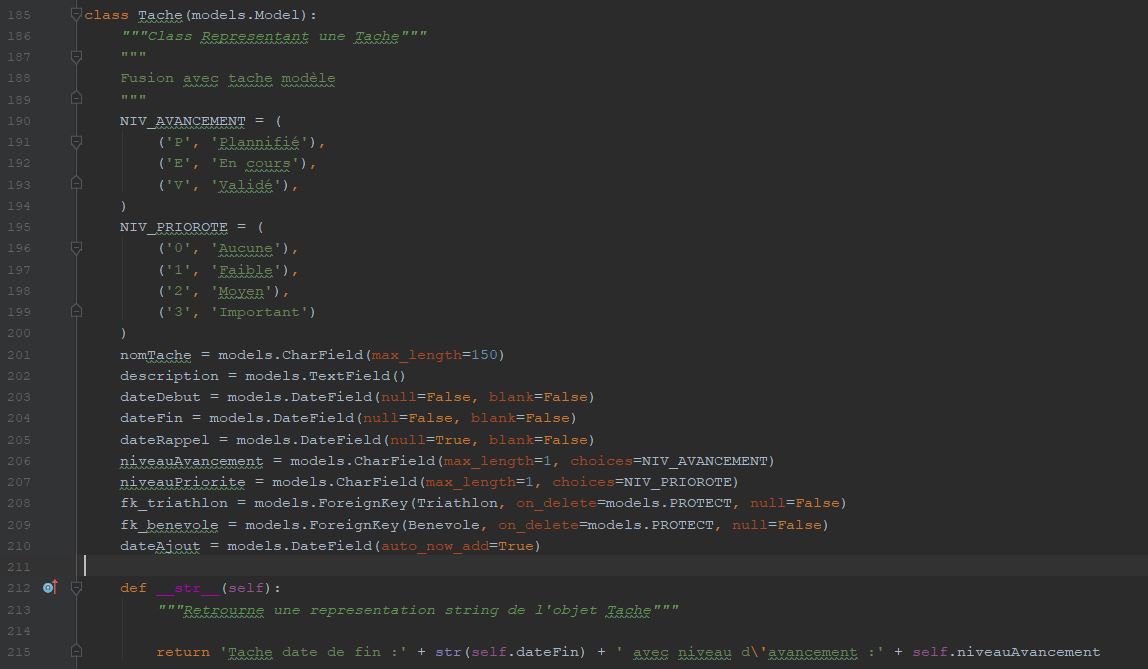
Nous pouvons constater que la table de Tache est définie comme une classe.

Figure : DEFINITION DE LA TABLE TACHE

Le niveau d’avancement et le niveau de priorité sont définis comme des type de données pour faciliter leur utilisation (une tache « planifié » est représentée en base de données par « P ») cela permet d’une part d’optimiser la base de données mais aussi de restreindre les états possibles de la variable.

Chaque colonne de la base de donnée est définit comme une variable via une fonction de type models.xxxxx :

CharField pour les Char.

TextField pour un Texte.

DateField pour une date.

ForeignKey pour une clé étrangère.

D’autres types de fonctions ont été utilisés qui ne seront pas développés ici.

Remarque : la création de clé primaire est automatique par défaut (elle peut toutefois si besoin est être redéfinis.

Différents paramètres peuvent être donnés pour les fonctions :

* max\_length définit la taille.
* null autorise les valeur null ou non.
* blank autorise un contenu vide ou non.
* choices permet de restreindre les valeur aux types précédemment définit (NIV\_AVANCEMENT, NIV\_PRIORITE).
* Pour les clés étrangères il est nécessaire de préciser en paramètre la table qui est indexé.
* On\_delete permet de paramétrer le comportement de la base si la clé référence par la clé étrangère vient à être supprimée, models.protect permet de protéger de la suppression en cascade.
* Auto\_now\_add permet pour une date un ajout automatique du contenue de la date à aujourd’hui.

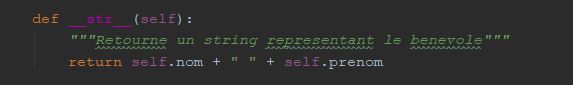
Def \_\_str\_\_ (self) est une fonction qui permet de renvoyer un String pour l’objet créer représentant la table, c’est l’équivalent du toString pour java. Cette fonction est par défaut implémenté dans Django est peut-être surchargé au besoin, exemple figure suivante :

Figure : surcharge de \_\_str\_\_ pour la table bénévoles

Ainsi l’objet « bénévole » qui permet de travailler sur la table bénévole de la base de données pourra à travers l’appel de la méthode \_\_str\_\_ remonté comme un String définit par le nom et le prénom du bénévole.

### La Vue

MAXIME

### Le Template

DIEGO

### Les formulaires

JULIEN

## Test de robustesse

HAD

# Bilan

## Bilan technique

JULIEN

Application des choses vue en cours, méthode agile + test + TDD (essayer de mettre en place mais pas assez mature avec le langage) ……

+ ce qu’il reste à faire…

## Bilan humain

JULIEN

Différence de fonctionnement, mise en place terrain d’entente. Réunion personnelle point négatif et positif sans langue de bois

# Conclusion

Dans, ce rapport nous mettons l’accent sur l’organisation. En effet, après l’établissement d’un cahier des charges et d’un dossier de conception se basant sur les principes UML, nous avons aborder la programmation en utilisant les méthodes agiles que nous avons vu en cours de notre formation. Nous avons pu noter que c’est une méthode très pratique et très orienté client. De ce fait, nous avons été plutôt efficace aux nouvelles attentes du client. Celui-ci a pu tester notre application avant le livrable final. Pour notre part, nous avons été surpris par la facilité à nous organiser et à travailler ensemble efficacement. En définissant la taille de nos fonctions à implémenter, nous avons pu mesurer la taille de notre projet et donc nous avons gardé un rythme soutenu et régulier tout au long, afin de fournir un livrable correct dans le temps qui nous était imparti.

D’autre part, nous avons trouvé que la technologie GIT apporte une réelle plus-value d’un point de vue personnelle. Cela, c’est évidement traduit par une facilité de travailler en groupe et notre projet a pu avancer à grand pas. Nous sommes conscients du travail des uns et des autres, et les nouvelles notions ont pu être acquises de manière homogène au sein du groupe.

En ce qui concerne la programmation, même si, nous n’avons pas forcément utilisant un langage vu en cours (ici Python), nous avons essayé d’utiliser au maximum les méthodes de travail académiques comme les tests et l’utilisation du langage UML. En effet, nous avons trouvé que le patron de travail Django utilise la programmation objet et se rapproche fortement du modèle MVC vue en cours d’année.

Enfin, humainement, nous avons, malgré certaines difficultés, réussi à travailler en groupe et progresser ensemble. Nous sommes montés en compétence par l’utilisation de la méthode agile, la technologie Git et par l’acquisition du patron de travail Django. Nous avions volontairement fait une conception plutôt ambitieuse (voir MCD) afin d’y trouver un chalenge et de nous forcer à utiliser des méthodes de travail très efficace. Le bilan est donc très positif pour nous. En espérant que le client soit également satisfait.

# 

1. https://fr.wikipedia.org/wiki/M%C3%A9thode\_agile [↑](#footnote-ref-1)
2. https://trello.com/b/op3sFIJV [↑](#footnote-ref-2)
3. En informatique et plus particulièrement en génie logiciel, extreme programming (XP) est une méthode agile plus particulièrement orientée sur l'aspect réalisation d'une application, sans pour autant négliger l'aspect gestion de projet. [↑](#footnote-ref-3)
4. https://github.com/SteelCorp [↑](#footnote-ref-4)
5. https://fr.wikipedia.org/wiki/Git [↑](#footnote-ref-5)
6. Source : https://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-django/le-fonctionnement-de-django [↑](#footnote-ref-6)
7. https://fr.wikipedia.org/wiki/Mapping\_objet-relationnel [↑](#footnote-ref-7)