Université Côte d'Azur/DS4H
MIAGE – MASTER 1

Rapport final

Suivi de projet

par
Paul BUREL
Geoffrey LALIC
Axel MAÏSSA
Djade KHALDI
Rémy NGUYEN
Aryamaan SINGH KUNWAR

Monsieur Olivier OUDOT

Sommaire

Introduction

Objectifs du projet

L'objectif de ce projet est d'appliquer le plus fidèlement possible les méthodes, techniques et outils de suivi de projet telles qu'on pourrait le faire lors de la réalisation d'un projet en entreprise.

Ce suivi est appliqué à un projet de développement Java : la réalisation d'un scrabble sous forme de webservice.

Les étapes d'analyse, de conception et de réalisation se sont succédées afin d'effectuer quatre livraison sur une période de huit semaines. Une date de rendu a été définit, laquelle n'étant pas extensible.

Les six membres de l'équipe se sont divisés en trois binômes et se sont repartis des modules différents.

Chaque semaine l'avancement a été reporté sur un tableur afin de suivre le consommé de chaque tâche et de remarquer les dépassements éventuels.

Liste des membres de l'équipe avec leur rôle respectif

Dans l'ordre alphabétique,

Paul BUREL

Binôme de Ayramaan
S'occupe du développement de la classe Partie, accès sur les tests fonctionnels sur l'ensemble des classes et implémentation spring, tests des classes communes.

Geoffrey LALIC

Binôme de Rémy
Développement de la classe Anagramme, mise en place de Docker, test de la classe anagramme, commun, conception algo joueur. Ecriture du programme Python permettant de générer le suivi de projet.

Axel MAÏSSA

Binôme de Djade
Développement de la classe Joueur et de classes communes. Suivi de projet, rédaction du rapport final.

Djade KHALDI

Développement de la classe Joueur et de classes communes. Intégration de spring boot, test d'intégration de Joueur.

Rémy NGUYEN

Développement de la classe Anagramme, mise en place de Docker, test de la classe anagramme, commun, conception algo joueur. Suivi de projet.

Aryamaan SINGH KUNWAR

Développement de la classe Partie, classes communes, architecture et implémentation spring boot.

Contenu technique

Sujet du projet

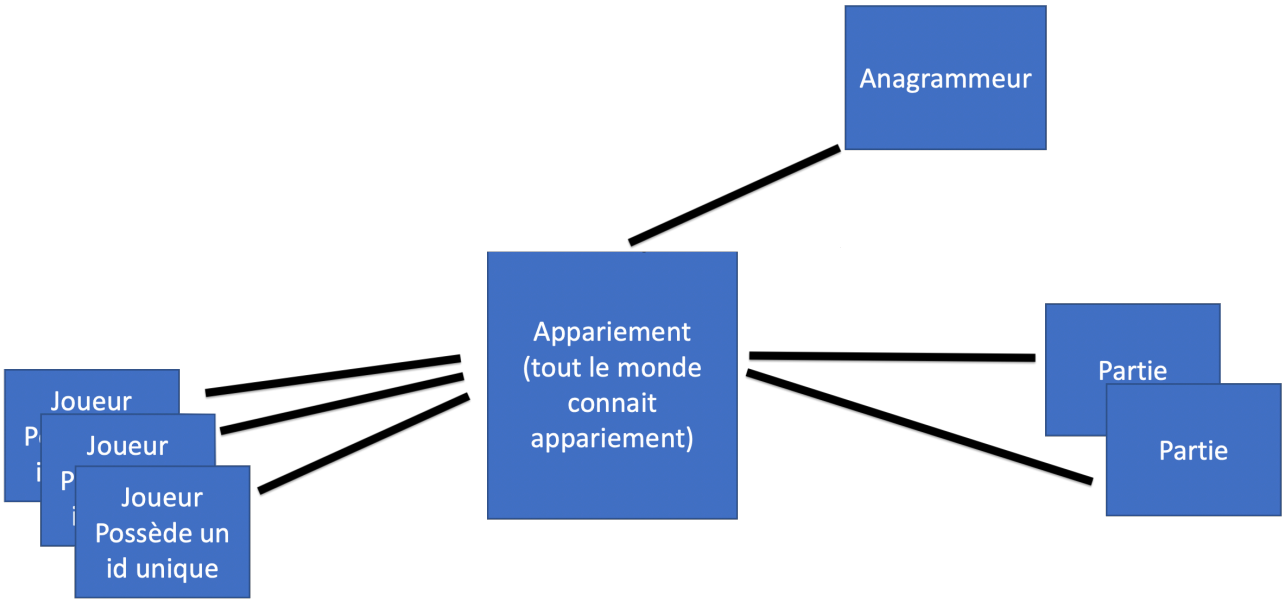
Le but du projet de développement est de créer un scrabble sous forme de webservice. Chaque service (un joueur, la partie, …) est un conteneur docker qui expose un webservice. Une partie est jouée sans humain.

La partie distribue 7 lettres à chaque joueur, elle leur demande de jouer, les joueurs propose une liste des « Propositions » (Objet contenant une lettre, et la position à laquelle elle doit être posée) ou, s'il ne peut pas joueur, il défausse une lettre et partie lui en donne une nouvelle en échange.

La partie reçoit cette Proposition, vérifie qu'elle est valide (vrai mot, position autorisée, …) et l'inscrit sûr le plateau.

Vue statique

Une architecture générale a été donnée dans le sujet avec un conteneur par joueur, une classe appariement, un anagramme et un conteneur par partie.



Le déroulement est organisé par la classe partie. La classe Appariement est chargée de faire le lien entre un joueur qui souhaite jouer et une partie. L'anagrammeur est chargée de trouver des mots lorsqu'on lui donne un array de lettre.

Sont imposés des langages, outils et technologies, voici la liste :

- Java / Java Spring

- Architecture Web Service
- Versionnage avec Github
- Intégration continue sur Travis CI
- Test d’intégration
- Orchestration des conteneurs avec docker-compose

Planning

Liste des tâches

Analyse sujet (AS) :
Cette tâche consiste à cerner le sujet ainsi que la logique et les règles du jeu de Scrabble.

Conception jeu scrabble (CS) :
Le but de cette tâche et de concevoir le jeu, sous forme de code, donc distinguer les différentes partie à coder.

Heure de TD dédié au projet (TD) :
Heure de travaux dirigés par M. RENEVIER dans le cadre de l'UE Outils pour Ing. Logicielle et M. OUDOT pour l'UE Suivi de projet. De façon générale, l'équipe ne pointera pas sur cette tâche car pendant les heures de TD, l'équipe a travaillé sur d'autre tâche (développement, suivi de projet).

Réunion Team (R) :
Comptabilisation des réunions de l'équipe.

Project Tracking (PT):
Tâche qui regroupe l'activité de suivi de projet (inscription workload), le script python permettant de généré de KPI et la rédaction du rapport final.

MODULE JOUEUR (DJ)

Architecture de la classe Joueur (DJC):
Mise en place d'un squelette de Joueur.

Développement fonctions Joueur (DJF):
Développement des méthodes de joueur : algorithme permettant de choisir des lettres à poser ainsi que leur position.

Développement WebServices Joueur (DJWS):
Développement du webservice joueur afin qu'il puisse être utilisé par Partie.

MODULE PARTIE (DP)

Architecture de la classe Partie (DPC):
Mise en place d'un squelette de la classe Partie.

Développement fonctions Partie (DPF):
Développement des méthodes de Partie, gestion des tours, des points, de la pioche et du plateau.

Développement WebServices Partie (DPWS):
Développement du webservice Partie.

MODULE APPARIEMENT (DA)

Architecture de la classe Appariement (DAC):
Mise en place d'un squelette de la classe Appariement.

Développement fonctions Appariement (DAF):
Développement des méthodes de la classe Appariement.

Développement WebServices Appariement (DAWS):
Développement du webservice Appariement.

MODULE ANAGRAMMEUR (DAN)

Architecture de la classe Anagrammeur (DANC):
Mise en place d'un squelette de la classe Anagrammeur.

Développement fonctions Anagrammeur (DANF):
Développement des méthodes de la classe Anagrammeur avec algorithme qui permet de choisir des mots en fonction d'une liste de caractères.

Développement WebServices Anagrammeur (DANWS):
Développement du webservice Anagrammeur.

MODULE COMMUN (DC)

Architecture des classes Commun (DCC):
Mise en place d'un squelette des classes communes (Plateau, Proposition, Case, …).

Développement fonctions Commun (DCF):
Développement des méthodes des classes communes. Plateau, Plateau factory, case, proposition.

Architecture Spring/WebService (SWS):
Architecture générale des Webservices avec spring.

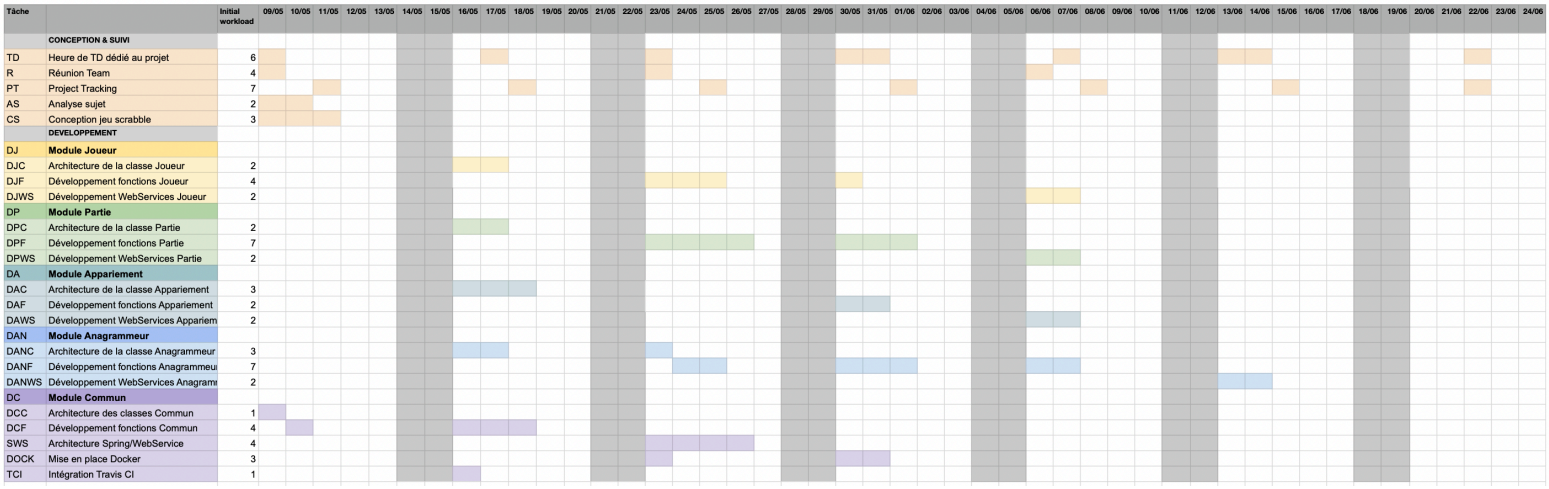
Mise en place Docker (DOCK):
Dockerisation des webservices.

Intégration Travis CI (TCI):
Intégration des pipelines Travis qui seront exécutées lors d'un commit + push.

ID	Description	Initial Workload	Personnes assignées
TD	Heure de TD dédié au projet	6	Tous
R	Réunion Team	4	Tous
PT	Project Tracking	7	Tous
AS	Analyse sujet	2	Tous
CS	Conception jeu scrabble	3	Tous
DJ	**Module Joueur**		
DJC	Architecture de la classe Joueur	2	Axel, Djade
DJF	Développement fonctions Joueur	4	Axel, Djade
DJWS	Développement WebServices Joueur	2	Axel, Djade
DP	**Module Partie**		
DPC	Architecture de la classe Partie	2	Aryamaan, Paul
DPF	Développement fonctions Partie	7	Aryamaan, Paul
DPWS	Développement WebServices Partie	2	Aryamaan, Paul
DA	**Module Appariement**		
DAC	Architecture de la classe Appariement	3	Tous
DAF	Développement fonctions Appariement	2	Tous
DAWS	Développement WebServices Appariement	2	Tous
DAN	**Module Anagrammeur**		
DANC	Architecture de la classe Anagrammeur	3	Geoffrey, Remy

ID	Description	Initial Workload	Personnes assignées
DANF	Développement fonctions Anagrammeur	7	Geoffrey, Remy
DANWS	Développement WebServices Anagrammeur	2	Geoffrey, Remy
DC	**Module Commun**		
DCC	Architecture des classes Commun	1	Tous
DCF	Développement fonctions Commun	4	Tous
SWS	Architecture Spring/WebService	4	Tous
DOCK	Mise en place Docker	3	Tous
TCI	Intégration Travis CI	1	Tous

Diagramme de Gantt



Suivi de projet

---

3 KPI

ESTIMATION INTIALE VS ACTUEL

Afin de voir si nous avions bien estimé notre initial workload, nous avons crée un indicateur simple qui fait la différence entre l'estimation initiale et actuel. Si ce chiffre est positif, c'est que nous avons surestimé la tâche (ou l'ensemble des tâches), s'il est négatif, c'est que nous l'avons sous-estimé. Enfin, s'il est nul, c'est que l'estimation était parfaite.

Formule : initial workload - actual workload

JOURS DE TRAVAIL RESTANT

Afin de vérifier si nous ne sommes pas en retard et que nous pourrons finaliser le projet dans le temps voulu, nous proposons cet indicateur. Il suffit de comparer ce chiffre à la somme des workload restant pour vérifier qu'on ait pas un problème : si ce chiffre est supérieur à la somme des workload, on finira certainement à temps, si elle est inférieur, c'est qu'il ne nous reste pas suffisamment de temps pour terminer le projet et qu'il va donc falloir re-priorisé et abandonner des tâches.

Formule : Nb jour de travail avant livraison finale **x** Nb membre dans l'équipe
ou : Nb jour de travail dans une semaine **x** Nb semaine avant livraison finale **x** Nb membre dans l'équipe

RATIO JOUR MANQUANT / RESTANT

Pour être certain de respecter la date de rendu, nous avons utiliser un ratio entre les jours de travails manquants (car au début du projet, nous sommes partis avec 73 jour/homme d'initial workload pour 63 jour/homme de travail disponible) et ceux restants. Notre but étant de faire tendre ce ratio vers un point d'équilibre à zéro en fin de projet (jours manquants = jours restants) et surtout pas un ratio supérieur à 1 (jours manquants > jours restant).

Formule : Nombre de jour de travail manquant (restant - remaining) / Nombre de jour de travail restant (détails de la formule dans le paragraphe précédent)

Programme Python

Dans le fichier Project.py, nous avons développé les différentes classes qui composent le tracking excel.

```python
class Project:
    def __init__(self, name, duration):
        self.name = name
        self.duration = duration
        self.tasks = []
        self.ressources = []
        self.tracking = []
```

La classe Project contient : ‘name’ qui est le nom du projet, ‘duration’ qui correspond à la durée en mois du projet, ‘tasks’ qui est la liste des tâches, ‘ressources’ qui représente la liste des ressources utilisées, et ‘tracking’ qui est une liste des tracks.

```python
def add_ressource(self, person):
    self.ressources.append(person)

def add_task(self, task):
    self.tasks.append(task)

def add_progression(self, date, who, task, workload, comment):
    self.tracking.append(Track(date=date, who=who, task=task, workload=workload, comment=comment))
    task.realized += workload
    task.remaining = task.actual - task.realized
    task.progress = task.realized / task.actual * 100
```

La méthode add_ressource, permet d’ajouter une ressource à la liste ‘ressources’ du projet.
La méthode add_task, permet d’ajouter une tâche à la liste ‘task’ du projet.
La méthode add_progress, permet de créer et d’ajouter une track à la liste ‘tracking’ du projet. Elle permet également de mettre à jour les différents attributs d’une tâche comme ‘task.realized’ , ‘task.remaining’, ’task.progress’.

```python
def sum_workload_per_ressource(self):
    sum = {}
    for ressource in self.ressources:
        workload = 0
        for track in self.tracking:
            if ressource == track.who:
                workload += track.workload
        print("\n")
        sum[ressource] = workload
    return sum

def sum_workload_per_task(self):
    sum = {}
    for task in self.tasks:
        workload = 0
        for track in self.tracking:
            if task == track.what:
                workload += track.workload
        sum[task] = workload
    return sum
```

La méthode sum_workload_per_ressource, retourne un dictionnaire contenant la somme de workload par ressource.
La méthode sum_workload_per_task, retourne un dictionnaire contenant la somme de workload par tâche.

```
def print_summary(self):
    res = ""
    res += repr(self) + '\nResources in this project :\n'
    res += "{:15}{:15}".format("id", "name") + "\n"
    for ressource in self.ressources:
        res += "{:15}{:15}".format(ressource.id, ressource.name) + "\n"
    res += '\n\n'
    res += "All tasks\n"
    res += "{:<15}{:<40}{:>15}{:>15}{:>15}{:>15}".format("id", "desc", "load", "actual", "realized",
                                                        "remaining", "progress") + "\n"
    for task in self.tasks:
        res += repr(task) + "\n"
    res += "\nSum of workload per resource\n"
    works = self.sum_workload_per_ressource()
    res += "{:40}{:40}".format("Resource:", "Workload:") + "\n"
    for key in works.keys():
        res += "{:40}{:40}\n".format(key.id, str("%.2f" % works[key]))
    res += "\n\nSum of workload per task\n"
    work_task = self.sum_workload_per_task()
    res += "{:40}{:40}".format("Task:", "Workload:") + "\n"
    for key in work_task.keys():
        res += "{:40}{:40}".format(key.id, str("%.2f" % work_task[key])) + "\n"
        # res += key.id + " -> " + str(work_task[key]) + " j/h; \n "
    res += "\n"

    res += "\nTracking: \n"
    res += "{:<15}{:<15}{:<15}{:<15}{:15}".format("Date", "Who", "Task", "Workload", "Comment") + "\n"
    for track in self.tracking:
        res += repr(track) + '\n'

    print(res)
```

La méthode print_summary, permet d'afficher tous les attributs du projet en formatant chaque valeur pour les visualiser facilement. On y retrouve les ressources, tâches, tracks et les méthodes sum_workload_per_ressource et sum_workload_per_task pour le tableau croisé dynamique.

```
class Ressource:
    def __init__(self, id, name):
        self.id = id
        self.name = name
```

La classe Ressource contient : 'id' qui est l'id de la ressource et 'name' qui est le nom de la ressource.

```python
class Task:
    def __init__(self, id, desc, load, actual):
        self.id = id
        self.desc = desc
        self.load = load
        self.actual = actual
        self.realized = 0
        self.remaining = 0
        self.progress = 0

    def __repr__(self) -> str:
        return "{:<15}{:<40}{:>15}{:>15}{:>15}{:>15}{:>15}".format(self.id, self.desc, self.load, self.actual,
                                                                   "%.2f" % self.realized, "%.2f" % self.remaining,
                                                                   "%.2f" % self.progress + "%")
```

La classe Task contient : ‘id’ qui est l’id de la tâche, ‘desc’ qui est le description de la tâche, ‘load’, ‘actual’, ‘remaining‘ et ‘progress’ sont les différents attributs du table excel.
La méthode __repr__ permet de formater l’objet Task.

```python
class Track:
    def __init__(self, date, who, task, workload, comment):
        self.date = date
        self.who = who
        self.what = task
        self.workload = workload
        self.comment = comment

    def __repr__(self):
        return "{:<15}{:<15}{:<15}{:<15}{:>15}".format(self.date, self.who.id, self.what.id, self.workload,
                                                       self.comment)
```

La classe Track contient : ‘date’ qui est date du track, ‘who’ qui représente la ressource issue du track, ‘what’ qui est la tâche en cours d’accomplissement, ‘workload’ qui est le nombre de jours/ homme utilisé pour effectuer la tâche et ‘comment’ qui est le commentaire lié au track.

Pour ce qui est du fichier main.py.

```python
prj = Project(name="Spring", duration=2)
# Project resources / team members
LG = Ressource(id='LG', name="LALIC Geoffrey")
NR = Ressource(id='NR', name="NGUYEN Rémy")
KD = Ressource(id='KD', name="KHALDI Djade")
MA = Ressource(id='MA', name="MAISSA Axel")
SKA = Ressource(id='SKA', name="SINGH Kunwar Aryamaan")
BP = Ressource(id='BP', name="BUREL Paul")
team = [BP, KD, LG, MA, NR, SKA]
for person in team:
    prj.add_ressource(person)
```

Nous avons créé un objet projet et ajouté ses ressources. En appelant la méthode add_ressource.

```python
# Project tasks
TD = Task(id='TD', desc='Heure de TD dédié au projet ', load=6, actual=6)
R = Task(id='R', desc='Réunion Team', load=4, actual=1)
PT = Task(id='PT', desc='Project Tracking', load=7, actual=4)
DJ = Task(id='DJ', desc='Module Joueur', load=0, actual=0)
DJC = Task(id='DJC', desc='Architecture de la classe Joueur', load=2, actual=1)
DJF = Task(id='DJF', desc='Développement fonctions', load=4, actual=5)
DJWS = Task(id='DJWS', desc='Développement WebServices Joueur', load=2, actual=2)
```

Nous avons initialisé les toutes les tâches qui composent le projet.

```
prj.add_task(TD)
prj.add_task(R)
prj.add_task(PT)
prj.add_task(DJ)
prj.add_task(DJC)
prj.add_task(DJF)
prj.add_task(DJWS)
```

Nous les avons ajoutées au projet. En appelant la méhode add_task.

```
prj.add_progression(date="09/05/2022", who=BP, task=TD, workload=0.2, comment="Mise en place architecture github, création pom.xml")
prj.add_progression(date="09/05/2022", who=KD, task=TD, workload=0.2, comment="Mise en place architecture github, création pom.xml")
prj.add_progression(date="09/05/2022", who=LG, task=TD, workload=0.2, comment="Mise en place architecture github, création pom.xml")
prj.add_progression(date="09/05/2022", who=MA, task=TD, workload=0.2, comment="Mise en place architecture github, création pom.xml")
prj.add_progression(date="09/05/2022", who=NR, task=TD, workload=0.2, comment="Mise en place architecture github, création pom.xml")
prj.add_progression(date="09/05/2022", who=SKA, task=TD, workload=0.2, comment="Mise en place architecture github, création pom.xml")
prj.add_progression(date="09/05/2022", who=MA, task=TCI, workload=0.2, comment="Mise en place de travis")
prj.add_progression(date="09/05/2022", who=BP, task=AS, workload=0.5, comment="Analyse du sujet en groupe")
prj.add_progression(date="09/05/2022", who=KD, task=AS, workload=0.5, comment="Analyse du sujet en groupe")
```

Nous avons des Tracks en fonction de l'avancement des tâches. En appelant la méthode add_progression.

```
prj.print_summary()
```

Enfin nous appelons la méthode print_summary.
Voici l'affichage du print_summary :

```
Project: name: Spring, duration: 2
Resources in this project :
id          name
BP          BUREL Paul
KD          KHALDI Djade
LG          LALIC Geoffrey
MA          MAISSA Axel
NR          NGUYEN Rémy
SKA         SINGH Kunwar Aryamaan
```

```
Sum of workload per resource
Resource:                   Workload:
BP                          6.00
KD                          6.45
LG                          6.10
MA                          7.05
NR                          6.50
SKA                         6.30
```

```
Sum of workload per task
Task:                       Workload:
TD                          5.00
R                           0.60
PT                          1.90
DJ                          0.00
DJC                         0.60
DJF                         3.45
DJWS                        1.05
DP                          0.00
DPC                         0.60
DPF                         3.00
DPWS                        0.75
DA                          0.00
DAC                         0.20
```

```
All tasks
id          desc                                load        actual      realized    remaining   progress
TD          Heure de TD dédié au projet            6            6          5.00        1.00      83.33%
R           Réunion Team                           4            1          0.60        0.40      60.00%
PT          Project Tracking                       7            4          1.90        2.10      47.50%
DJ          Module Joueur                          0            0          0.00        0.00       0.00%
DJC         Architecture de la classe Joueur       2            1          0.60        0.40      60.00%
DJF         Développement fonctions                4            5          3.45        1.55      69.00%
DJWS        Développement WebServices Joueur       2            2          1.05        0.95      52.50%
DP          Module Partie                          0            0          0.00        0.00       0.00%
```

```
Tracking:
Date        Who         Task        Workload    Comment
09/05/2022  BP          TD          0.2         Mise en place architecture github, création pom.xml
09/05/2022  KD          TD          0.2         Mise en place architecture github, création pom.xml
09/05/2022  LG          TD          0.2         Mise en place architecture github, création pom.xml
09/05/2022  MA          TD          0.2         Mise en place architecture github, création pom.xml
09/05/2022  NR          TD          0.2         Mise en place architecture github, création pom.xml
09/05/2022  SKA         TD          0.2         Mise en place architecture github, création pom.xml
09/05/2022  MA          TCI         0.2         Mise en place de travis
09/05/2022  BP          AS          0.5         Analyse du sujet en groupe
09/05/2022  KD          AS          0.5         Analyse du sujet en groupe
```

Pour lancer le main depuis un terminal, il faut installer python et lancer la commande :
« python3 .\Tracking\pythonTracking\main.py », depuis le dossier root.

Bilan

Bilan Global sur le Suivi de Projet

Nous sommes satisfaits de l'évolution de notre suivi de projet qui été assez défaitiste au début, avec un initial workload supérieur au nombre de jour de travail disponibles, qui a été corrigé au fils des semaines. La tâche a tout de même été assez lourde et stressante, surtout lorsque nos estimations montrées que nous n'allions pas terminer le projet de développement. Au final, le projet de développement ainsi que celui de suivi ont été terminés.

Apprentissage, Problème et Améliorations

Nous avons découvert que notre estimation initiale était plutôt mauvaise, certaines tâches que nous avions estimées à 6 jour/hommes ont en fait été terminées avec 2 jour/hommes. Cette différence peut être constatées en comparant les colonnes « Initial workload » et « Actual ». L'initial workload a été surestimé d'environ 1,6 fois (75 / 45).

Egalement, sur l'estimation et le tracking des tâches, nous nous sommes rendus compte que l'utilisation de l'unité jour/homme n'était pas optimale et qu'il aurait été préférable d'utiliser des heure/homme puisque nos tâches sont plutôt bien découpées et que la majorité de nos séances de travailles se résumes à 2h (soit 0,28 jour/homme).