Alexandre Pominville (1392978)
Julien Gélineau Poirier (0766919)
Émile Tremblay (1647637)
Éric Marquis (9171475)
Pleshko Oleksandr Stepanovitch (1508300)

29 août 2022

Cahier de charges : Sprint 1

remis dans le cadre du cours

420-B34-RO – Développement des applications informatiques

Collège de Rosemont

LEA 3.N – Programmation orientée objet et technologies WEB

Explication du projet et fonctionnalités souhaitées

Dans cette section, je vais détailler l'interprétation d'équipe quant à la définition des besoins énoncés par le client. Ce dernier nous a demandé de réaliser un logiciel qui permettra une validation informatisée des feuilles de temps des employés afin de s'assurer que celles-ci respectent bien les règles en place dans l'entreprise. Actuellement, chaque semaine, tous les employés doivent remplir une feuille de temps donc en faire la validation est une tâche de longue haleine.

Afin de sauver du temps, l'entreprise nous a demandé d'établir un logiciel permettant de générer des messages d'erreurs si une feuille de temps ne respecte pas les règles en place dans l'entreprise. À l'heure actuelle, l'entreprise ne fait pas mention d'une interface graphique puisqu'elle souhaite que la logique applicative du *backend* soit développée de façon prioritaire. Voici, une liste des exigences fonctionnelles que ce prototype d'application devra implémenter :

EF1 : Le patron doit être capable de fournir une feuille de temps en entrée au logiciel.

On considère que la feuille de temps sera toujours structurée de la même façon. Celle-ci sera représentée par un fichier .json qui devra contenir les informations suivantes :


```
[
    {
        "id_employe_projet": 1,
        "numero_employe": 5000,
        "numero_projet": 350,
        "id_feuille_temps": 1,
        "temps_travaille": 45,
        "date_travail": "2022-01-01"
    },
    {
        "id_employe_projet": 2,
        "numero_employe": 5000,
        "numero_projet": 400,
        "id_feuille_temps": 1,
        "temps_travaille": 45,
        "date_travail": "2022-08-25"
    },
    {
        "id_employe_projet": 3,
        "numero_employe": 5000,
        "numero_projet": 350,
        "id_feuille_temps": 1,
        "temps_travaille": 45,
        "date_travail": "2022-07-26"
```

Cette structure a été choisie afin de faciliter éventuellement une migration vers une base de donnée dans des étapes (Nous avons inclus le diagramme de classe UML ainsi que notre diagramme de données en Annexe).

id_employe_projet: Il s'agit de l'identifiant de notre objet EmployeProjet, cet objet représente un temps travaillé par un employé sur un projet particulier à une date quelconque.
numero_employe: Il s'agit de l'identifiant de notre employé. Les employés ayant un numéro d'employé inférieur à 1000 sont des employés de l'administration sinon ils sont des employés réguliers.
numero_projet: Il s'agit de l'identifiant de notre projet. Les projets ayant un numéro de projet supérieur à 900 sont des codes de télétravail.
id_feuille_temps: Chaque feuille de temps hebdomadaire remplie par un employé aura un identifiant unique et éventuellement pourrait contenir la semaine de l'année auquel cette feuille de temps correspond.
temps_travaille: Correspond au temps travaillé en minutes par l'employé sur le projet durant cette journée.
date_travail: Il s'agit de la date correspondante à l'entrée EmployeProjet faite par l'employé

EF2 : Le logiciel doit tester si les données entrées dans la feuille de temps respectent chacune des 7 règles de l'entreprise. Ce dernier doit fournir les messages appropriés en fonction de si les règles sont respectées ou non.

Les fichiers de résultat contiennent un seul objet Résultat contenant le numéro d'employé ainsi qu'un tableau d'objet Règle

Résultat: Contient le numéro de l'employé et un tableau de règle
Règle: Chaque règle est définie par un *id_règle* numérique identifiant le numéro de la règle, un booléen *est_respectée* indiquant si la règle a été respectée dans la feuille de temps fournie au logiciel et un message approprié détaillant pourquoi celle-ci est respectée ou non.

Voici un exemple d'un fichier résultat :

```
{
  "numero_employe": 5000,
  "regles": [
    {
      "id_regle": 1,
      "est_respectee": true,
      "message": "Il s'agit d'un employé normal qui n'est pas assujetti a cette règle."
    },
    {
      "id_regle": 2,
      "est_respectee": false,
      "message": "L'employé régulier n'a pas travaillé au moins 38 heures au bureau cette semaine."
    },
    {
      "id_regle": 3,
      "est_respectee": true,
      "message": "L'employé n'a pas travaillé plus de 43 heures au bureau cette semaine."
    },
```


Voici les règles définies par l'entreprise:

Règle #1 : Les employés de l'administration doivent travailler au moins 36 heures au bureau par semaine.

Règle #2 : Les employés réguliers doivent travailler au moins 38 heures au bureau par semaine.

Règle #3 : Aucun employé n'a le droit de passer plus de 43 heures au bureau.

Règle #4 : Les employés de l'administration ne peuvent pas faire plus de 10 heures de télétravail par semaine.

Règle #5 : Les employés réguliers peuvent faire autant de télétravail qu'ils le souhaitent.

Règle #6 : Les employés réguliers doivent faire un minimum de 6 heures au bureau pour tout les jours ouvrables.

Règle #7 : Les employés de l'administration doivent faire un minimum de 4 heures au bureau pour tout les jours ouvrables.

Explication du fonctionnement du logiciel

Pour utiliser notre application vous devez vous rendre dans le dossier suivant : release/Sprint1 qui se trouve à la racine de notre dossier de projet à l'aide d'une invite de commande. À l'intérieur se trouvera un fichier *Timesheet.jar* qui pourra être exécutée avec cette commande spécifiant deux arguments soit: la feuille de temps à valider et le fichier qui contiendra le résultat. Pour ce faire, vous devez bien sûr avoir installé préalablement la JDK 11.

Voici un exemple:

java -jar Timesheet.jar feuille_temps1.json resultat1.json

Ainsi, feuille_temps1.json est la feuille de temps à valider et resultat1.json le fichier de résultat. Il est à noter que la feuille de temps doit se trouver dans le même dossier que *Timesheet.jar*. Le fichier de resultat sera toujours produit dans le même dossier que le .jar.

Division du travail en équipe

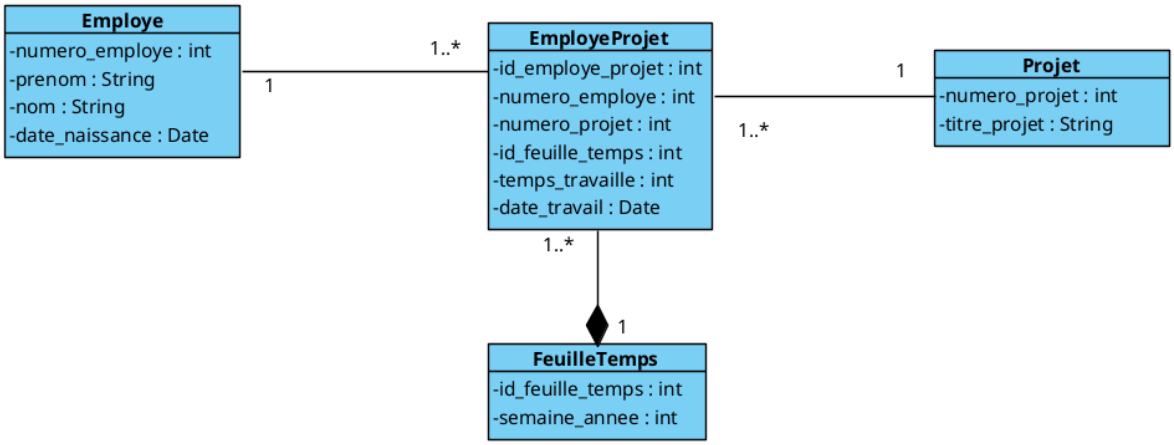
Afin de s'assurer de la cohésion du démarrage de notre projet nous avons réalisé la plus grande partie de ce premier sprint en mode eXtreme programming. Nous voulions être sûr de bien modéliser les parties maîtresses de notre logiciel afin de faciliter son extensibilité pour les mises à jour futures. Nous avons divisé le projet entre tout les membres de l'équipe Scrum pour la réalisation des méthodes devant tester chacune des règles puisque celles-ci se prêtaient bien à cet exercice mettant à profit l'asynchronisme. Éric a pris en charge de réaliser les feuilles de temps qui seront utilisées pour tester le fonctionnement de chacun des méthodes validant les règles de l'entreprises.

Poker Scrum

Nous n'avons pas réalisé de Poker Scrum à cette étape étant donné le faible temps requis pour la réalisation de chacun des fonctionalités, nous étions d'avis que nous allions être capable de réaliser le code en un après-midi. Puisque la charge de travail était supplémentaire pour la deuxième étape et donc plus facilement découpable nous avons décider de le réaliser à cette étape du projet. Nous sommes toutefois conscient qu'il faut réaliser cette estimation collective des temps de réalisation de chaque tacĥe à chaque début de sprint.

Annexe

Diagramme de classe UML



Modèle de données