

Projet Informatique

Guo Sylvain, Rachdi Mustapha, Hoarau Romain, Paterni Thomas

Mars 2022

Table des matières

1	Organisation	2
2	Conception logicielle	3
2.1	Modèle objet	3
2.1.1	Classes	3
2.1.2	Auxiliaires	3
2.2	Structure de données	3
3	Stratégie de résolution	4
3.1	Algorithmes principaux	4
3.1.1	L'écriture	4
3.1.2	Classe Entreprise	4
3.2	Indicateurs	4
4	Bilan	4
4.1	Résultats numériques	4
4.2	Limites et améliorations possibles	4

1 Organisation

En choisissant de travailler dans l'environnement Visual Studio Code, nous nous sommes naturellement tournés vers une des solutions proposées par Microsoft : Live Share. Cette extension est basée sur la technologie WebRTC, ce qui permet de communiquer des données en temps réel. Cependant, cette extension ne nous permettait pas de travailler sur un même fichier à n'importe quelle heure, car il fallait qu'un membre du groupe héberge la session.

A défaut de mettre en place un serveur hébergeur, nous avons préféré nous tourner vers d'autres méthodes telles que celle proposée par Git. Nous avons choisi *Github* comme hébergeur de code, puis nous avons décidé d'utiliser le logiciel *Gitkraken*, qui propose une interface graphique, facilitant ainsi la compréhension. Ce logiciel nous a permis de nous concentrer sur différentes parties afin de nous répartir au mieux les tâches. La présence de l'historique nous a permis de comparer et de visualiser les différences entre plusieurs versions d'un même fichier de manière simultanée par le biais d'un écran partagé (voir 1).

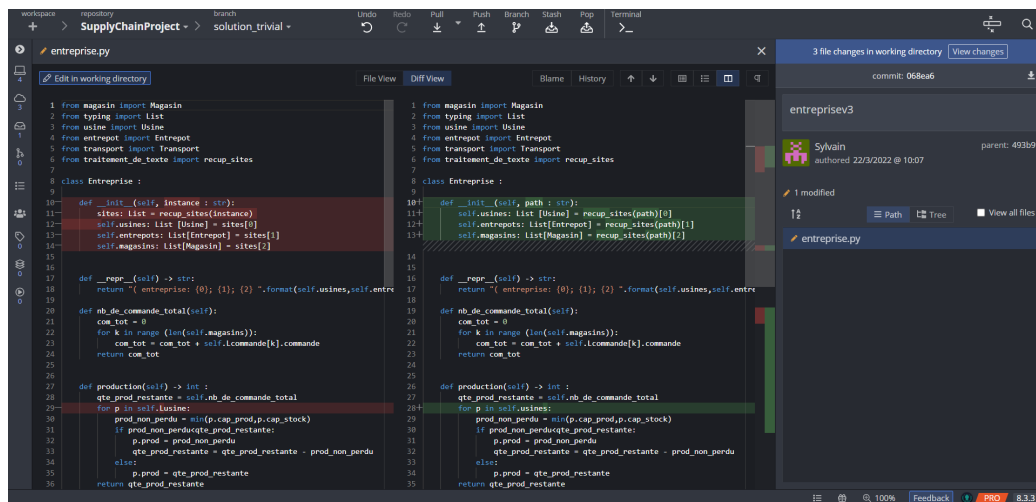


FIGURE 1 – Split view de Gitkraken

Le lien vers le dépôt du code est *disponible ici*.

2 Conception logicielle

2.1 Modèle objet

2.1.1 Classes

Nous avons fait le choix de diviser chaque entité en une classe. La classe *Site* recense les attributs communs des trois lieux (usines, entrepôts et magasins). Nous avons privilégié l'utilisation de l'héritage proposé par Python en utilisant l'initialisation avec *build*. Nous avons pu ainsi créer trois classes filles à partir de la classe mère *Site*.

- La classe *Usine*
- La classe *Entrepot*
- La classe *Magasin*

Par ailleurs, nous avons créé une classe *Transport* qui modélise le transport d'un site *i* vers un site *j*. Comme il ne s'agit pas d'un site, nous n'avons pas utilisé d'héritage pour la construire.

De même, nous avons construit une classe *Entreprise* qui comporte les méthodes principales de résolution du problème.

Enfin, la classe *Probleme* fournie joue le rôle du chef d'orchestre de toutes les autres. C'est elle qui sera appelée pour la résolution du problème.

2.1.2 Auxiliaires

- *traitement_de_texte* : Ce fichier comporte toutes les méthodes afin de réaliser tout le travail de lecture des différents fichiers *nom-*.txt*. Il est utilisé dans le programme principal.
- *generateur_de_demandes* comporte toutes les méthodes de génération de demandes.

2.2 Structure de données

Les classes *Usine*, *Entrepot* et *Magsin* sont définies comme filles de la classe *Site*. Même si peu de méthodes sont rajoutées dans les classes *Entrepot* et *Magsin* en particulier, l'utilisation de l'héritage permet toutefois d'assurer la clarté et la logique du code, notamment lors de différents appels de ces classes.

La classe *Site* comporte tous les attributs que peuvent contenir l'un des trois sites :

- La capacité de production
- Le coût de production
- La capacité de stockage

- Le coût de stockage
- Le stock initial
- Le stock final

Ainsi, pour les sites ne comprenant pas certains de ces attributs, il est nécessaire de définir ces-derniers comme égaux à 0. Cela permet d'assurer une cohérence notamment avec la lecture des fichiers (notamment *nom-site.txt*).

3 Stratégie de résolution

3.1 Algorithmes principaux

3.1.1 L'écriture

L'algorithme d'écriture se trouve dans la classe *Probleme*. Il se base sur le résultat de la méthode *sol()* de la classe *Entreprise*.

3.1.2 Classe Entreprise

Comme nous l'avons évoqué précédemment, la classe *Entreprise* comporte les méthodes principales. La méthode

- *production()* renvoie les quantités produites par chaque usine
- *trans()* planifie les transports
- *liste_com()* renvoie les commandes prévues pour chaque magasin
- *cout_prod_tot()* renvoie le coût total de production du jour
- *stock_sites()* rend compte de l'état des stocks de chaque site jour après jour
- *cout_total_stock()* renvoie le cout total du stock du jour
- *cout_trans()* renvoie le cout total de transport du jour

3.2 Indicateurs

4 Bilan

4.1 Résultats numériques

4.2 Limites et améliorations possibles