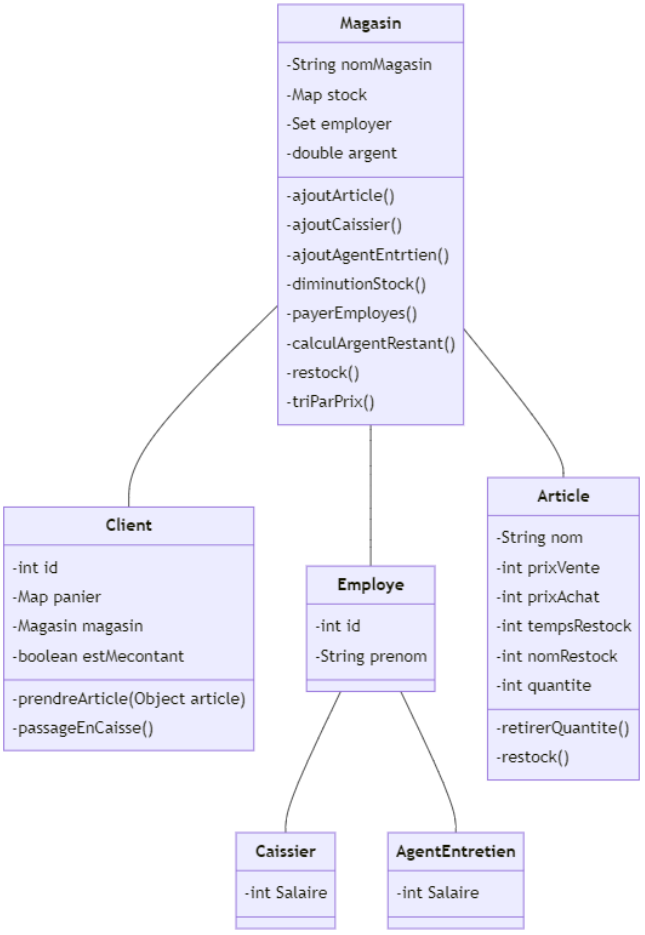
Documentation Projet Java

Etude de conception

L'idée principale est de simuler une pénurie dans un magasin telles que celles que nous avons pu vivre lors de la crise du COVID-19.
Pour ce faire, nous avons implémenté plusieurs classes: Magasin, Client, Article…



Le magasin fonctionne avec un système de “vagues”.
Comme montré dans le schéma ci-dessus, au lancement de l'application, 15 clients entrent dans le magasin par tour. Au bout de 18 tours, le début de la vague de client commence. La vague dure 2 tours, au cours desquelles environ 230 clients vont venir dans le magasin. Cette vague correspond aux dernières heures de la journée (de 18h à 20h) où il y a souvent le plus de monde.

A la fin de chaque vague, le programme nous envoie un récapitulatif de la journée et restock les articles qui sont prêts à être restockés.

```
------------------
Vague n°1
Argent restant : 10010.0
Nombre de clients : 245
Pourcentage de clients mécontents: 3%
pour continuer, écrivez C
```

Le magasin est doté d'une gestion de stock de ces articles et une gestion de restock d'articles en simulant des fournisseurs. Chaque article est restocké tous les X temps.

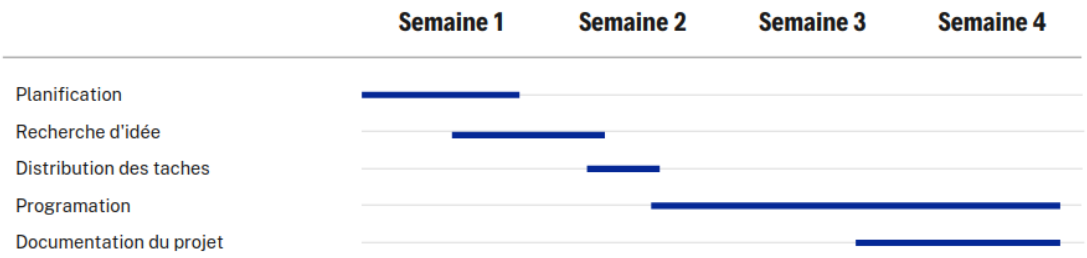
Nous avons aussi implémenté une note de satisfaction des clients, qui diminue en fonction du nombre de clients qui n'ont pas réussi à trouver les articles voulus dans les quantités voulues. Nous aurions aussi voulu implémenter un système d'ouverture et de fermeture des caisses, ce qui ajouterait une notion de perte de temps aux clients, s'il n'y a pas assez de caisses ouvertes pendant la vague par exemple, ce qui ferait encore diminuer la note de satisfaction.

Nous avons créé des classes "Employé", "Caissier" et "Agent d'entretien" pour faire en sorte de gérer les salaires à chaque fin de journée et faire baisser l'argent du magasin. Nous pouvons donc imaginer dans un programme plus abouti, un système de recrutement et de licenciement des employés en fonction des besoins humains, de la satisfaction des clients et des charges liés aux employés.

Nous aurions aussi voulu lister les produits où le stock diminue de façon anormale et faire en sorte que les clients achètent plus les produits avec peu de stock par peur d'en manquer. Par exemple, si l'huile d'olive venait à manquer, il y aurait plus de personnes qui voudraient en acheter, et comme le produit ne serait pas forcément disponible, la note de satisfaction baisserait.

Nous avions aussi pensé à faire en sorte que tous les 6 jours (6 vagues), la quantité de réapprovisionnement et la fréquence de chaque article soit réévaluée et modifiée si cet article est peu demandé ou au contraire toujours en rupture, ce qui permettrait d'adapter le magasin aux ruptures et de constater l'impact de la modification.

Diagramme de Gantt :

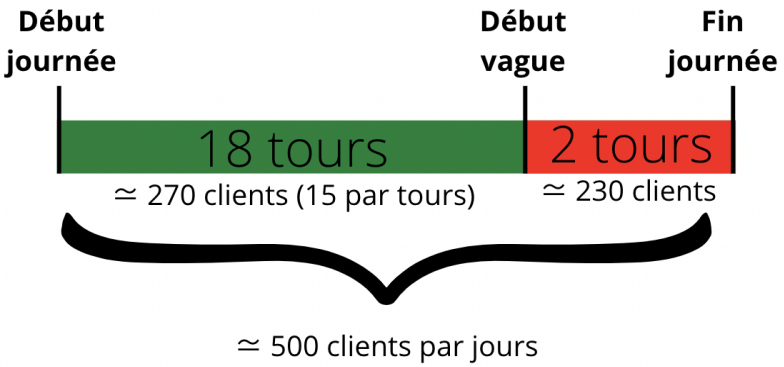


Clôture de travail

Synthèse du projet :

Actuellement notre simulation est composée :

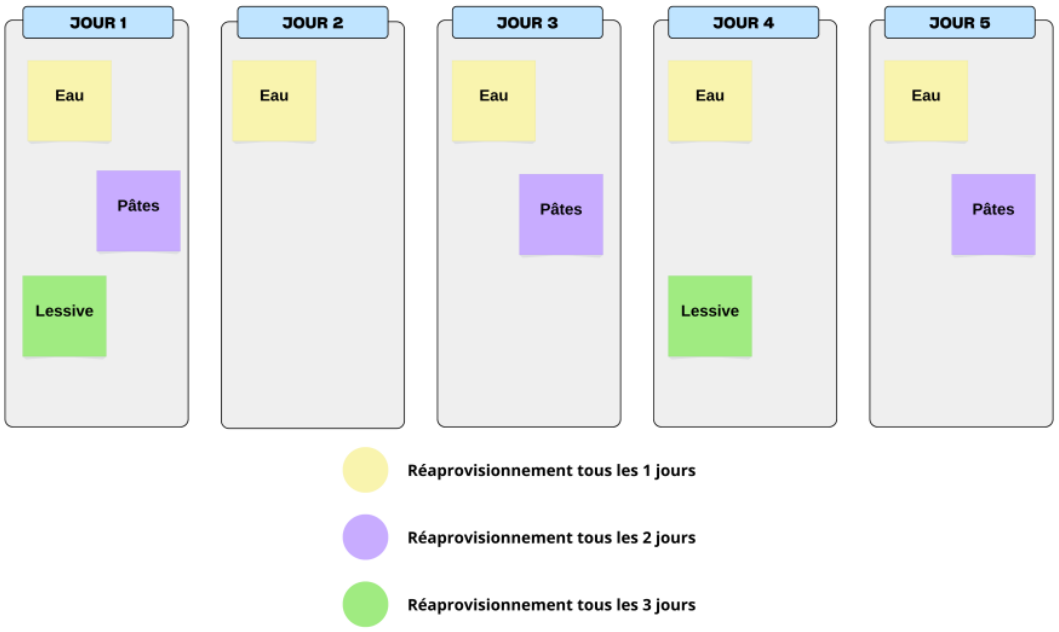
- D'un système de jours



1 jour = 500 personnes en 20 tours -> 15 personnes par tour et 230 les deux derniers tours.

- D’un système de stock et de restock

Pour chaque produit une quantité et une durée de restock lui sont attribuées.



- D’un système de satisfaction client

Pour chaque article non disponible, la note de satisfaction diminue.

- D’un système d'employés

Chaque employé a un salaire qui lui est versé à chaque début de vague.

Pour chaque fin de journée, nous aurons un petit récapitulatif qui est composé :
- Du numéro de la vague
- De l’argent restant dans le magasin
- Du nombre de clients qui sont passés dans le magasin
- D’une note de satisfaction (pourcentage de gens mécontents)

Exemple :

```
------------------
Argent restant avant ouverture : 8516.0
------------------
Vague n°1
Argent restant : 10010.0
Nombre de clients : 245
Pourcentage de clients mécontents: 3%
------------------
pour continuer, écrivez C
```

Bilan technique :

Afin de réaliser la simulation, nous avions besoin d'articles et de leur attribuer une quantité. Pour ce faire nous avons réalisé une classe "Article" qui possède plusieurs informations pratiques telles que son nom, son prix de vente, sa quantité... Ce choix permet de ne créer qu'un seul objet pour chaque article disponible dans un magasin et de stocker directement dans celui-ci toutes les informations nécessaires. La constitution de cette classe permettra également, dans le futur, d'implémenter d'autres fonctionnalités telles qu'une variation du prix des articles.

Il fallait ensuite un moyen de stocker les articles. Nous avons choisi de créer une classe "Magasin" dont l'attribut "stock" est une "HashMap" ce qui permet de lier une clé unique à une valeur. Ici la clé est le nom de l'article et sa valeur est l'article lui-même ce qui permet de n'avoir qu'un seul même nom d'article dans le stock et ne pas avoir de doublon (2 fois l'article "pâtes" par exemple). Cela facilite également la recherche dans le stock comparé à une structure de données de type "Set" dont on doit parcourir tous les ensembles de valeurs. Le réapprovisionnement est basé sur 2 attributs d'un article correspondant à une fréquence et une quantité. La fréquence est une intervalle de jours pendant laquelle l'article en question n'est pas

réapprovisionné et la quantité correspond à la quantité qui sera approvisionnée à chaque réapprovisionnement, ce qui permet de suivre l'évolution de la quantité de manière simple et efficace.

Pour la cohérence de la simulation, nous avons ajouté une classe "Client" qui permet de diminuer le stock d'un magasin de manière aléatoire. Nous générons 3 nombres grâce à un générateur de nombres pseudo-aléatoires: un correspondant au nombre d'articles qu'il va prendre, un autre correspondant à l'indice de l'article qu'il va prendre et un dernier afin de définir la quantité qu'il va prendre. Cela permet de vider le stock d'un magasin aléatoirement et ainsi impacter la quantité de ses articles qui peuvent se retrouver en rupture.

Nous avons également choisi d'ajouter une fonctionnalité de charges et bénéfices au magasin afin de voir en quoi une pénurie impacte les recettes d'un magasin. Des classes "Employé", "Caissier" et "Agent d'entretien" sont créées et permettent de simuler les charges du magasin. Le magasin doit payer à intervalles réguliers ses employés (charges) et reçoit le paiement de ses clients lors du passage en caisse. Les employés sont stockés dans une "HashSet" afin d'éviter les doublons et donc d'éviter des erreurs telles que payer 2 fois le salaire d'un employé. Cette structure nous oblige à la parcourir entièrement afin d'accéder à tous les employés, mais étant tous payés en même temps c'est une solution appropriée.

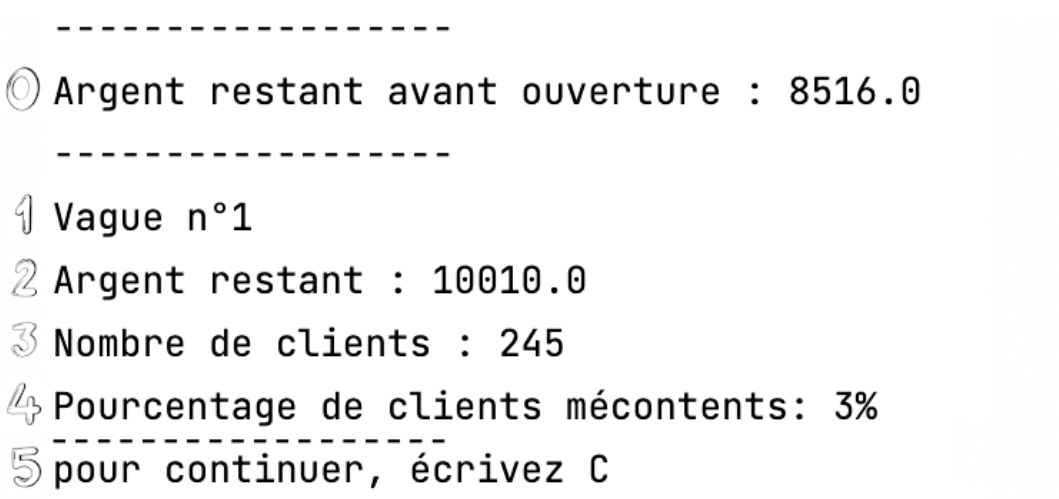
La fonctionnalité de "satisfaction du client" se base sur un "état de satisfaction". Cet état change si la quantité demandée par la client est supérieure à la quantité disponible en magasin, et nous calculons en chaque fin de journée le pourcentage de satisfaction basée sur l'ensemble des clients du jour. Cela permet de visualiser le changement de la satisfaction globale de jour en jour, les articles se faisant de plus en plus rares.

Problèmes rencontrés	Mesures d'amélioration
Nous avons rencontré un problème de temps. Nous avons été trop ambitieux.	Mieux répartir le travail à fournir en fonction des séances (ex : faire un Gantt plus détaillé)
Certains ont eu des incompréhensions sur l'utilisation des différentes structures de données: comment les parcourir etc.	Prendre un temps en groupe pour lire et comprendre comment fonctionne la structure.
Beaucoup d'indécisions sur les classes à produire, sur la méthode d'implémentation.	Plus s'écouter en groupe et prendre en compte l'avis de chacun, puis se positionner sur une façon de faire bien claire.

Manuel d’utilisation

(présent aussi dans le ReadMe)

Pour utiliser notre application, il suffit d'exécuter le code situé dans le fichier **src** du dépôt GitHub du projet.
À la fin de chaque vague (soit chaque fin de journée), l'utilisateur peut appuyer sur la touche "C” pour continuer et enchainer sur la vague suivante, ou appuyer sur une autre touche pour mettre fin à la simulation.



(0) Le montant total d’argent du magasin avant son ouverture (donc au début de la journée).
(1) Le numéro de la journée (et donc le numéro de la vague car il y a 1 vague/jour).
(2) Le montant total d’argent restant dans le magasin.
(3) Le nombre de clients ayant fait un passage dans le magasin dans la journée.
(4) Le pourcentage de clients mécontents (qui augmente lorsqu'un article est en rupture de stock ou, dans une future implémentation, dès qu'il y a une attente trop longue à la caisse).
(5) C’est l’instruction envoyée par le système afin de poursuivre la simulation.

Equipe :

Aurélien MATTERA
Ethan TESTA
Elisée LEYDIER
Lucas MURATET