



Projet Base de Données

Groupe 07

BOUCHELOUCHE Marhez

CHOLVY Jordan

DIALLO Thierno

DUONG Khoi

MARTIN Yohan

Github: https://github.com/l3miage-cholvyj/Groupe7_ProjetBD

Mise en situation	2
Rapport intermédiaire et correction	2-9
Fonctionnement de l'application et état d'avancement	10-12
Réalisation des tests en interne	13
Gestion de projet	14
Conclusion	15

Mise en situation

Le 7 mars 2022, notre groupe de travail composé de BOUCHELOUCHE Marhez, CHOLVY Jordan, DIALLO Thierno, DUONG Khoi et MARTIN Yohan a reçu pour mission la réalisation et la mise en place d'une application de location de vélo pour une commune.

En effet, sur la base des modèles Vélib (Paris), Vélo'v (Lyon) ou VÉCUB (Bordeaux) la ville de Mahingan Falls, située entre les villes de Salem et Groville, souhaite mettre en place un service de location vélos en libre-service.

Ce service, nommé VéPick, sera géré par une entreprise locale qui se lance dans le domaine de la location de vélos. Notre groupe de travail est chargé du système de gestion de données de VéPick, basé sur une base de données relationnelle.

Ce projet a pour objectif de mettre en pratique l'ensemble des connaissances acquises dans l'UE Base de données - Système d'information.

Rapport intermédiaire et correction

Pour mener à bien ce projet, la première tâche consistait à mettre en place un modèle conceptuel représentant le squelette de notre application. Le but était de pouvoir représenter visuellement la problématique propre au sujet et ainsi pouvoir commencer le projet sur des fondations stables.

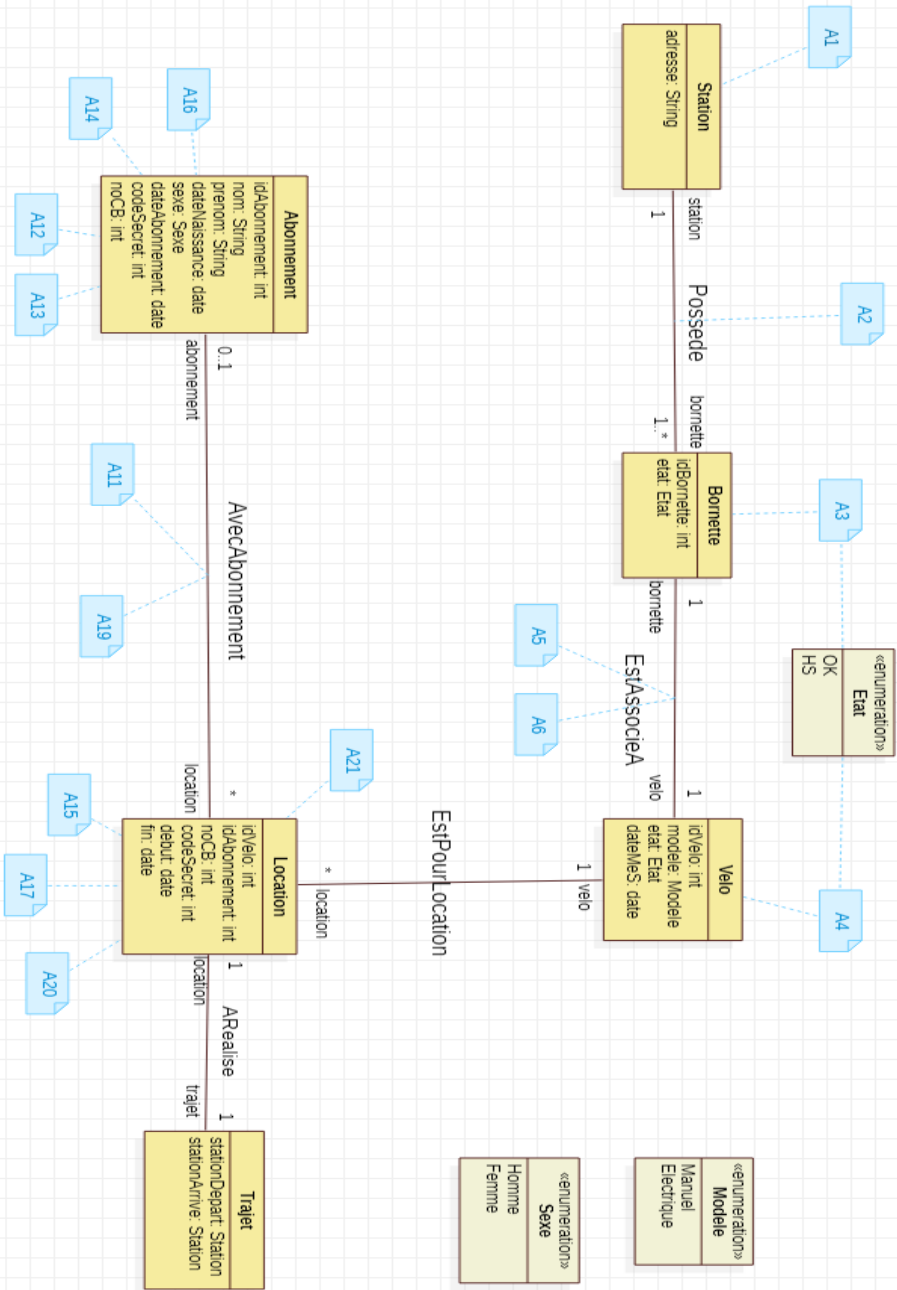
Nous devons mettre en place un modèle de données mais si plusieurs solutions nous semblaient pertinentes, alors, nous devons les proposer et émettre des critiques sur les problèmes que posaient chaque représentation.

Une fois cette tâche réalisée, il fallait faire le schéma relationnel de ce modèle.


Voici à quoi ressemblait notre rapport intermédiaire:

VéPick

=====
(A1) Chaque station est identifiée par son adresse. (A2) Elle possède plusieurs emplacements, appelés "bornettes", chacune pouvant accueillir un vélo. (A3) Chaque bornette d'une station est identifiée par un numéro et son état (OK, HS). (A4) Chaque vélo est décrit par son numéro identifiant, son modèle et la date de sa mise en service et son état (OK, HS). (A5) Achaque instant, un vélo peut être en station associée à une bornette, loué par un client, ou en maintenance. (A6) L'association d'un vélo à une bornette est automatique grâce à la puce RFID présente dans chaque vélo. (A11) Les clients peuvent être ou non abonnés au service VéPick. (A12) Les abonnés ont renseigné leur nom, prénom, date de naissance, sexe, adresse, numéro de CB, ainsi qu'un code secret leur permettant de s'identifier. (A13) Plusieurs abonnés différents peuvent avoir le même numéro de CB. (A15) Un abonné dispose de 30% de réduction sur le prix de location. (A16) Chaque abonné peut simplement louer un vélo en le prenant et le ramenant dans un parc après s'être identifié sur la bornette à l'aide de son code secret. (A19) Il est possible pour un client de louer plusieurs vélos. (A20) Pour chaque location par un non-abonné, un code secret est attribué au locataire, qu'il devra mémoriser pour s'identifier quand il rendra son vélo. (A21) Un client peut éventuellement louer plusieurs vélos à la fois.



Note: Les constraints de A1 - A21 sont assez bien faits. Les constraints de A22 jusqu'à A32 sont à continuer



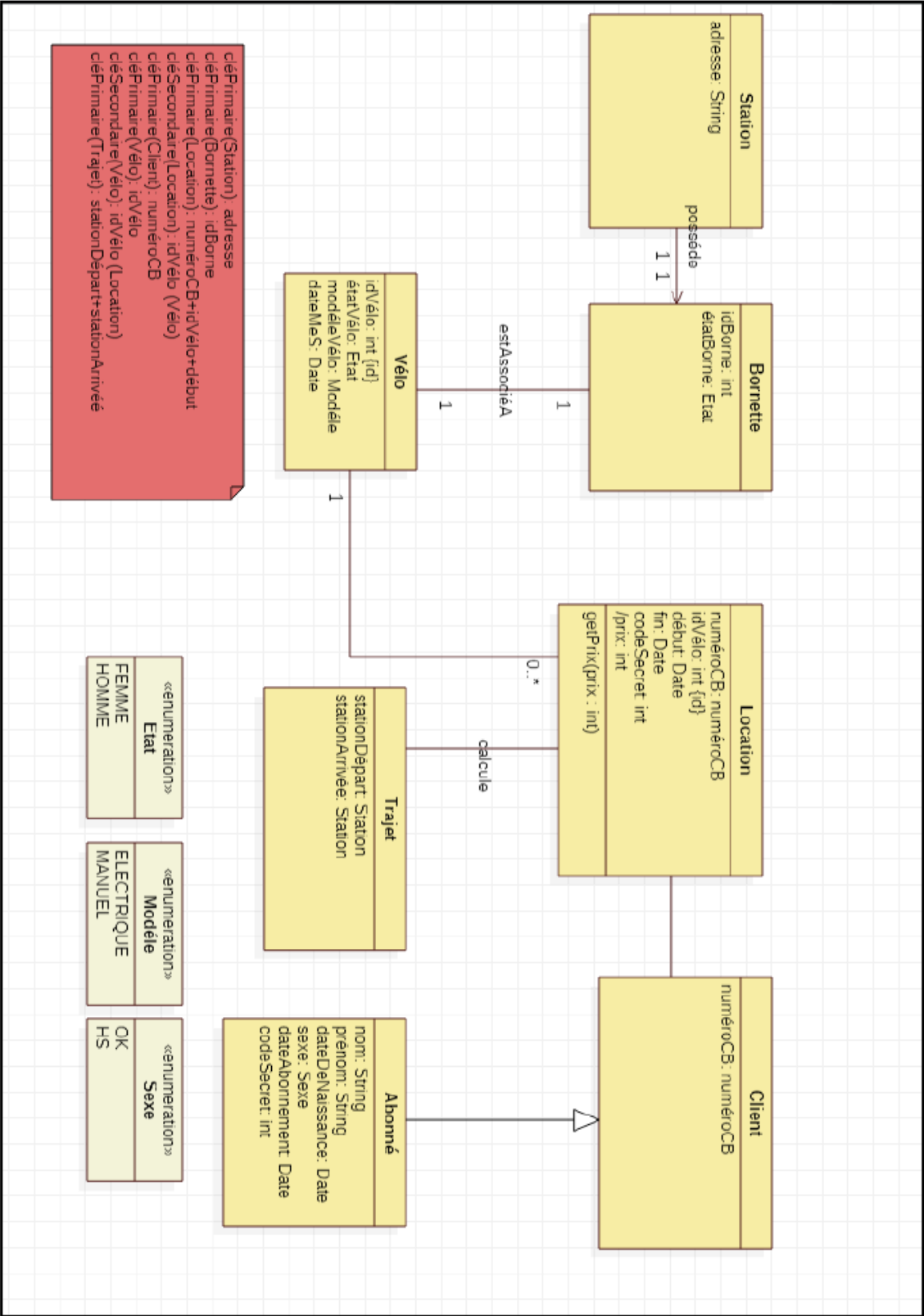
Cette première version est une mise au propre de notre première modélisation de la solution. Nous y avons réfléchi pendant la première séance de TP et pour tester notre modèle, nous y avons intégré des scénarios pour vérifier sa capacité à répondre aux besoins de VéPick.

Cette première version ne possède pas la classe Client, c'est un choix qui, au début, nous paraissait cohérent puisque les informations du client peuvent être intégrées à la classe Location.

L'un des problèmes de cette représentation est dans la non représentation du Client. Il n'est pas forcément intuitif d'imaginer une représentation d'un achat sans y intégrer visuellement un acheteur.

Ayant pris en compte cette problématique, nous avons alors proposé un deuxième modèle qui, cette fois-ci, sépare les table Location et Client en deux tables.

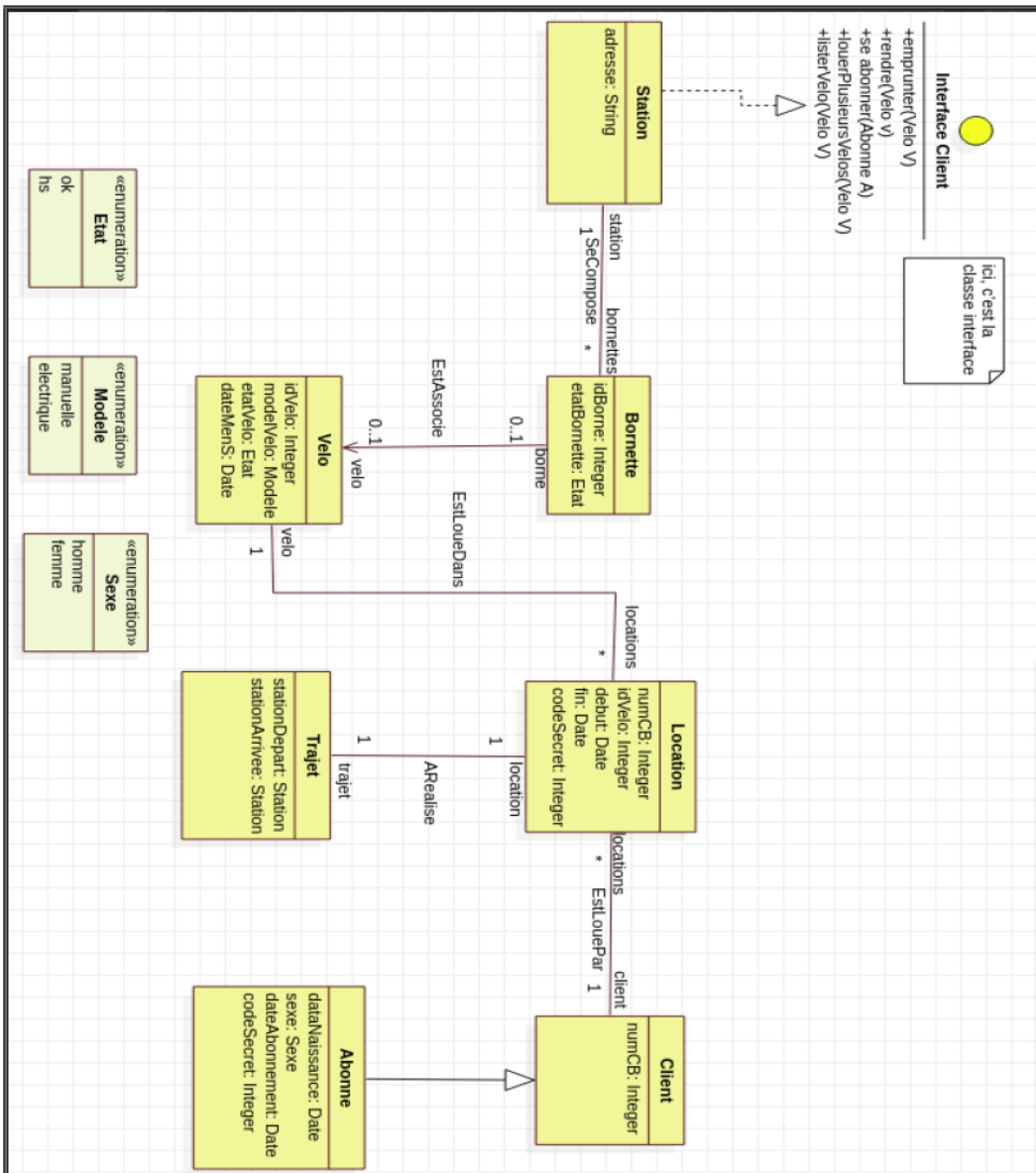
Version 2 : Avec classe Client & Locations non confondues (Clés primaires + Clés étrangères).




Nous avons passé moins de temps sur ce modèle, l'objectif était principalement de le comparer à notre première version.

Enfin nous avons mis en place une dernière représentation qui prend en compte l'interface et où les classes Location et Client ne sont pas confondues.

Version 3 : Avec classe Client & Locations non confondues + Interfaces :





Suite à la correction apportée par l'enseignement, nous avons amélioré notre modèle. Ce dernier prend en compte les critiques émises sur le rapport intermédiaire mais aussi l'implémentation d'une nouvelle table `LocationDetails` qui nous permet d'y insérer les informations propres à la location. Ainsi, elle permet de gérer tout ce qui concerne la gestion du trajet et facilite la lisibilité du modèle.

L'ensemble du groupe s'est mis d'accord pour valider ce modèle comme représentation définitive de notre application.

Diagramme de classe final:

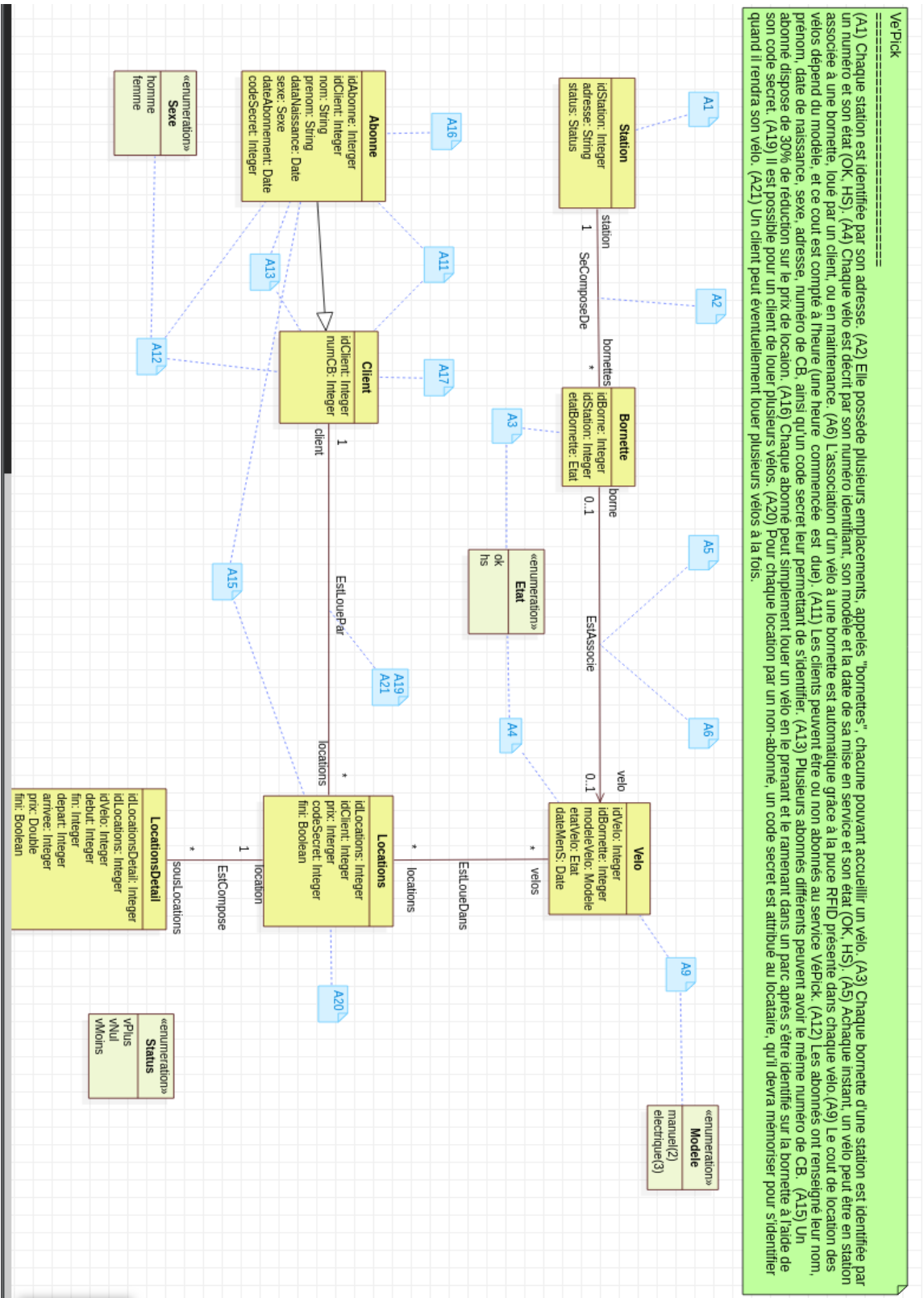
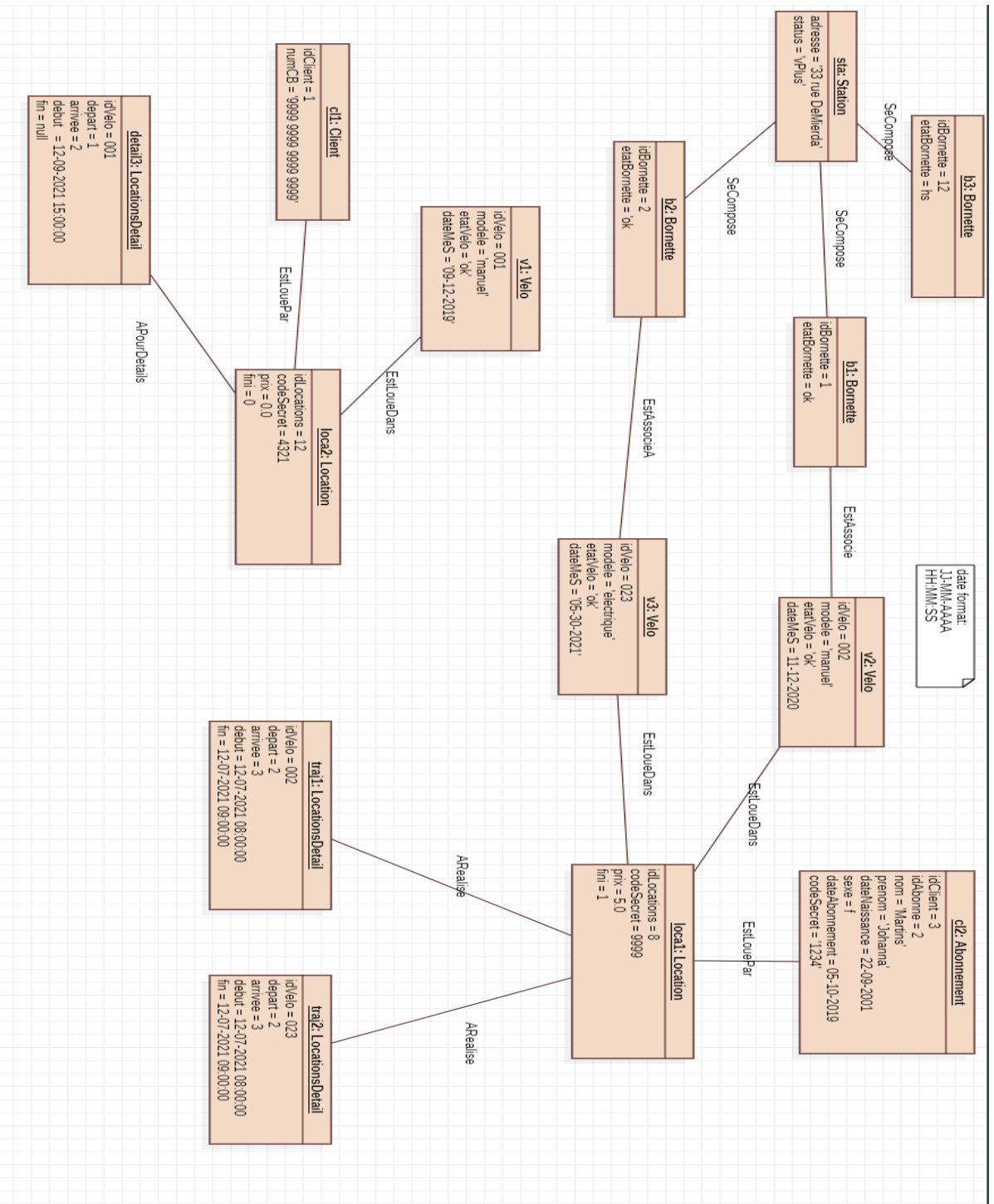


Diagramme d'objet final:



Fonctionnement de l'application et état d'avancement

Pour tout ce qui concerne les tests de l'application, l'interface est représentée par un interpréteur de commande. L'application, via son interface, permet à un client potentiel d'exécuter les différentes commandes.

Certaines fonctionnalités sont essentielles pour pouvoir présenter une application répondant au besoin du client:

1 – **Emprunt de vélo.** (B1) Abonné ou non, un client doit pouvoir récupérer un vélo non endommagé à une borne. (B2) L'interface doit lui signaler le code secret pour rendre le vélo s'il n'est pas abonné. (B3) Le client doit pouvoir alerter de l'état d'un vélo en panne.

2 – **Rendu de vélo.** (B4) Un client qui a emprunté un vélo doit pouvoir le ramener à une station s'il y a un emplacement vide et qu'il connaît le code secret. (B5) Le client doit pouvoir informer de l'état d'un vélo rendu en panne. (B6) Le client est débité du temps de location.

3 – **Abonnement au service.** (B7) En renseignant les informations nécessaires, un client doit pouvoir s'abonner à l'année.

4 – **Programme Vplus.** (B8) L'application doit gérer le système de prime de ce programme. L'application doit alors signaler à un client abonné qui prend ou rend son vélo s'il bénéficie d'une remise **Vplus** pour son prochain trajet.

5 – **Location de plusieurs vélos.** (B9) Un client doit pouvoir louer plusieurs vélos à la fois (pour sa famille ou des amis par exemple).

6 – (B10) Consulter le nombre de vélos dans chaque station, le nombre de vélos endommagés, et le nombre de places libres.

Fonctionnalités non-réalisées:

(B3) Le client doit pouvoir alerter de l'état d'un vélo en panne.

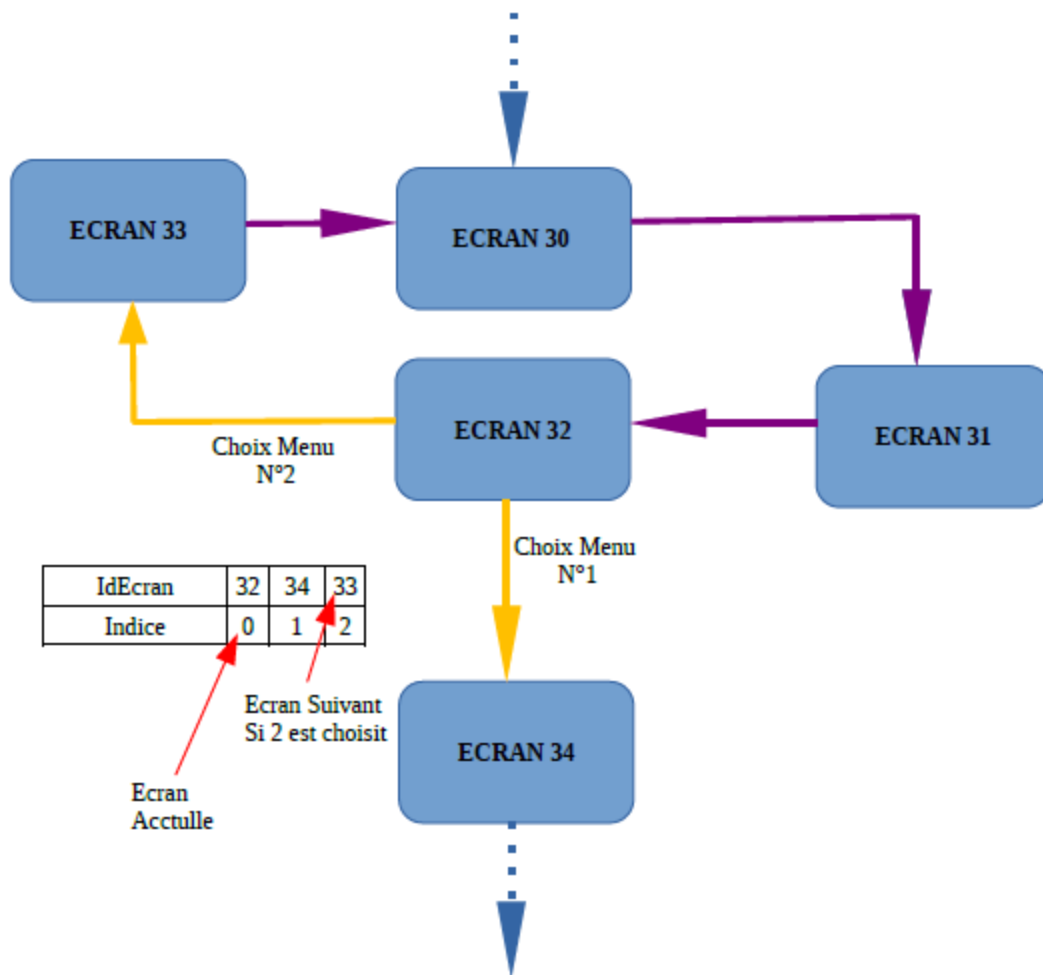
(B5) Le client doit pouvoir informer de l'état d'un vélo rendu en panne.

(B6) Le client est débité du temps de location.

(B8) L'application doit gérer le système de prime de ce programme. (Partiellement réalisée)

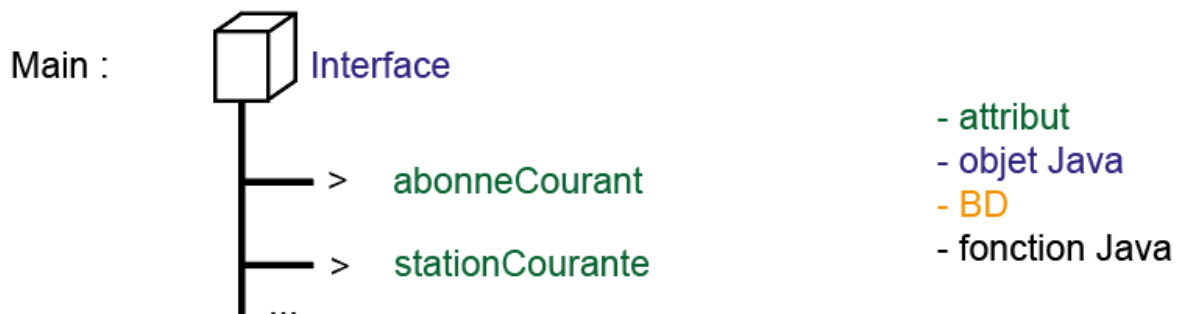
(B10) Consulter le nombre de vélos dans chaque station, le nombre de vélos endommagés, et le nombre de places libres. (SQL fait, JAVA non fait)

Interface

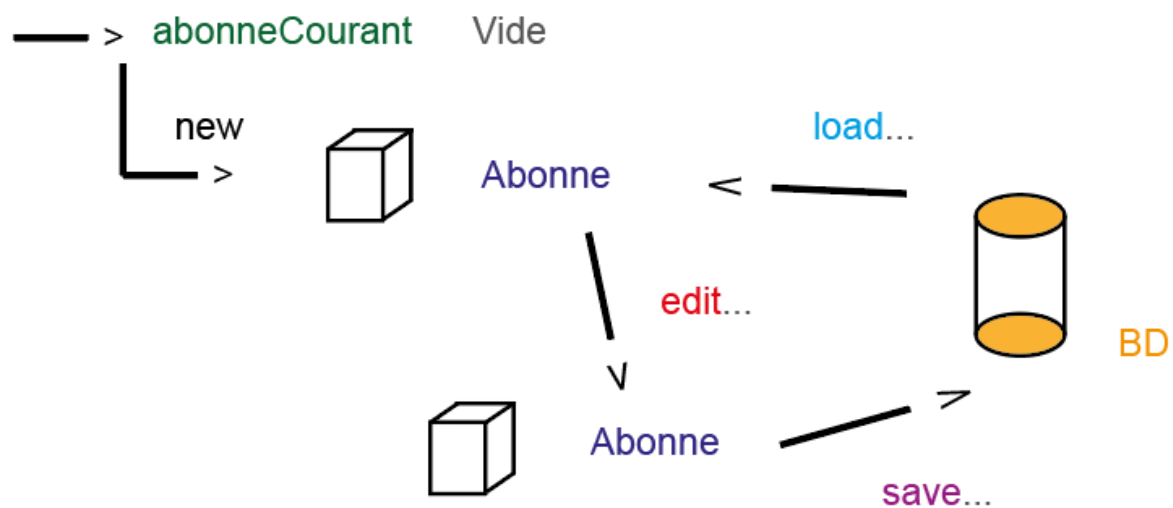


En ce qui concerne l'interface, nous avons mis en place un modèle permettant de suivre une logique de succession d'écran. Sur le graphique par exemple, l'écran suit une succession logique allant de l'écran 30 - 31 - 32, puis l'utilisateur est soumis à un choix de menu, en fonction de son choix, il va être envoyé, soit vers l'écran 33, soit vers l'écran 34, ce choix est fait selon la table de redirection.

Logique de fonctionnement:



Pour chaque attribut :



A chaque fois qu'un nouvel objet est créé, cet objet est nommé xxxCourant, il s'agit de l'objet sur lequel on travaille. Cet objet va ensuite load les données de la BDD puis il va y avoir un edit sur les données, l'objet ainsi modifié va être save sur la base de données.

Chaque objet courant est relié à un autre, ex: Vélo à Bornette, Bornette à Station...

Réalisation des tests en interne

Mise en place de la BDD:

Pour créer la base, il faut créer la database veli en local sur MySQL, ligne de commande (cf fichier création createBD.sql dans la package groupe7.bd.sql). Ensuite il faut créer les tables (cf fichier création TableBd.sql dans la package groupe7.bd.sql). Enfin il faut ajouter nos valeurs de test (cf addvaluesDataBase.sql dans la package groupe7.bd.sql).

Liste des commandes SQL utiles au projet: viewTables.sql

L'ensemble de nos tests ont été réalisés directement sur l'application, nous n'avons pas eu le temps de mettre en place des tests automatiques.

Gestion de projet

En ce qui concerne la gestion du temps de travail, nous avons travaillé sur l'ensemble des heures de TP mais nous avons aussi apporté du travail supplémentaire en réservant des créneaux sur notre temps libre pour avancer sur la programmation de l'application.

Pour tout ce qui concerne les décisions de groupe ou une présentation de solution impactant la structure de l'application, l'ensemble des membres du groupe proposait leurs solutions et leur avis. Ainsi, toute l'équipe pouvait participer et anticiper un changement dans la modélisation de la solution, évitant ainsi les possibles problèmes de réalisation lors de la mise en place de l'application.

En ce qui concerne les différentes tâches pour chacun des membres, les premières séances consistaient à réaliser en commun les différents diagrammes de classes et objets essentiels à la compréhension de notre solution. Tous les membres de l'équipe ont activement participé à cette tâche. Une fois le rapport intermédiaire envoyé et sa correction reçue, une bonne partie du temps de la deuxième séance a été consacrée à la prise en compte des différentes remarques et critiques de nos modélisations.

Nous sommes alors convenu à une conception finale qui répondaient aux différentes problématiques soulevées par l'enseignant correcteur.

A partir de là, les séances consistaient à réaliser les différentes fonctions nécessaires au fonctionnement de l'application.

De plus, chaque moment important dans l'avancement du projet devait être noté dans un compte rendu qui sert de support avant la rédaction du rapport final. A chaque séance, un certain temps a été accordé à la rédaction des différentes informations et réalisations de la séance.

En ce qui concerne les tâches spécifiques réalisées par chaque membre, Jordan a mis en place l'architecture et la conception de l'application ainsi que le débogage de certaines fonctionnalités liées aux interfaces.

Khoi et Marhez ont réalisé les fonctions nécessaires au fonctionnement de l'application tel qu'il est demandé dans le sujet du Projet.

Thierno s'est occupé de toute la mise en place de la conception de la base de données et DAO.

Yohan s'est occupé des différents comptes rendus de séance, de la rédaction du rapport de stage (graphique, texte...) ainsi que le débogage de certaines fonctionnalités liées à l'interface.

Conclusion

L'objectif de ce projet était de réaliser une application regroupant l'ensemble des connaissances acquises en base de données et en système d'information. De par nos différentes réflexions et analyses, nous avons mis en place une application répondant aux besoins exprimés dans la problématique du sujet.

Durant la réalisation, nous avons dû faire face à plusieurs problématiques nouvelles comme la gestion du temps, le travail de groupe ou encore la répartition des tâches.

Réaliser ce travail en groupe nous a permis de développer nos capacités de travail en commun et nous a fait comprendre l'importance d'organiser son temps de travail, son code et de prendre du temps pour mettre en place une méthode de conception commune au groupe pour faciliter les implémentations des fonctionnalités de chacun des membres du groupe.