Table des matières

[Enoncé du travail 2](#_Toc9444161)

[Planning des réunions 2](#_Toc9444162)

[L’analyse 3](#_Toc9444163)

[Diagramme de contexte statique 3](#_Toc9444164)

[Diagramme des Use Case 4](#_Toc9444165)

[Le diagramme de classe 4](#_Toc9444166)

[Le Lifeline 6](#_Toc9444167)

[Conclusion 7](#_Toc9444168)

[Zaretti Quentin 7](#_Toc9444169)

[Bol Axel 8](#_Toc9444170)

# Énoncé du travail

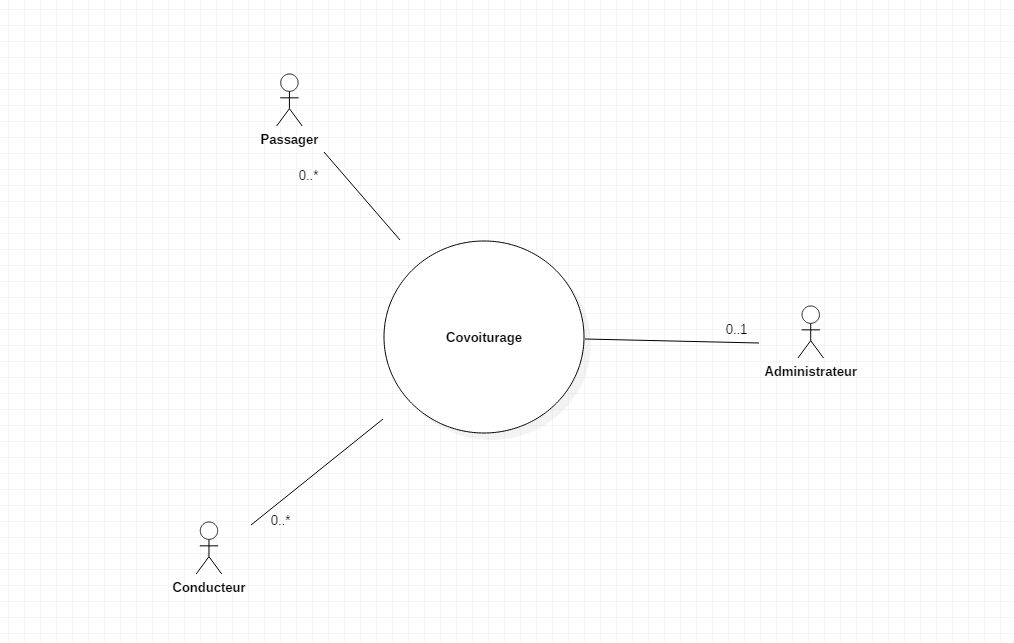
La gestion d’un système de covoiturage (au sein de l’école). L’objectif de l’application est d’automatiser un maximum de tâches. Le but du projet de cette année est la réalisation d’une application qui doit permettre aux différents acteurs d’effectuer des trajets vers l’école de façon conjointe et organisée.

# Planning des réunions

|  |  |  |
| --- | --- | --- |
| Date et lieu | Objectifs | Répartitions des tâches |
| Le 13/03/19  École Condorcet   |  |  | | --- | --- | | Q | A | | Définir quels types de diagrammes devra être fait, ainsi que le partage des tâches. | Quentin : Diagramme de classe  Axel : Use case |
| Le 20/03/19  Discord   |  |  | | --- | --- | | Q | A | | Correction ensemble du diagramme des uses case |  |
| Le 27/03/19  Facebook   |  |  | | --- | --- | | Q | A | | Discussion à propos des éventuelles façons de créer le diagramme de classe et mise en place d’un github | Quentin : Création github  <http://github.com/>Tisax190/Covoiturage |
| Le 1/04/19  Discord   |  |  | | --- | --- | | Q | A | | Création du diagramme de classe, colorisation du diagramme des uses cases en vue de demander une correction. | Quentin : Premier jet du diagramme de classe  Axel : Finalisation du use case et correction |
| Le 5/04/19  Discord   |  |  | | --- | --- | | Q | A | | Amélioration du diagramme de classe et création de lifeline | Quentin : correction du diagramme de classe  Axel : Ajout de schéma lifeline avec et sans exception |
| Le 6/04/19  En classe   |  |  | | --- | --- | | Q | A | | Commencer le projet  Et partage des tâches |  |

# L’analyse

## Diagramme de contexte statique



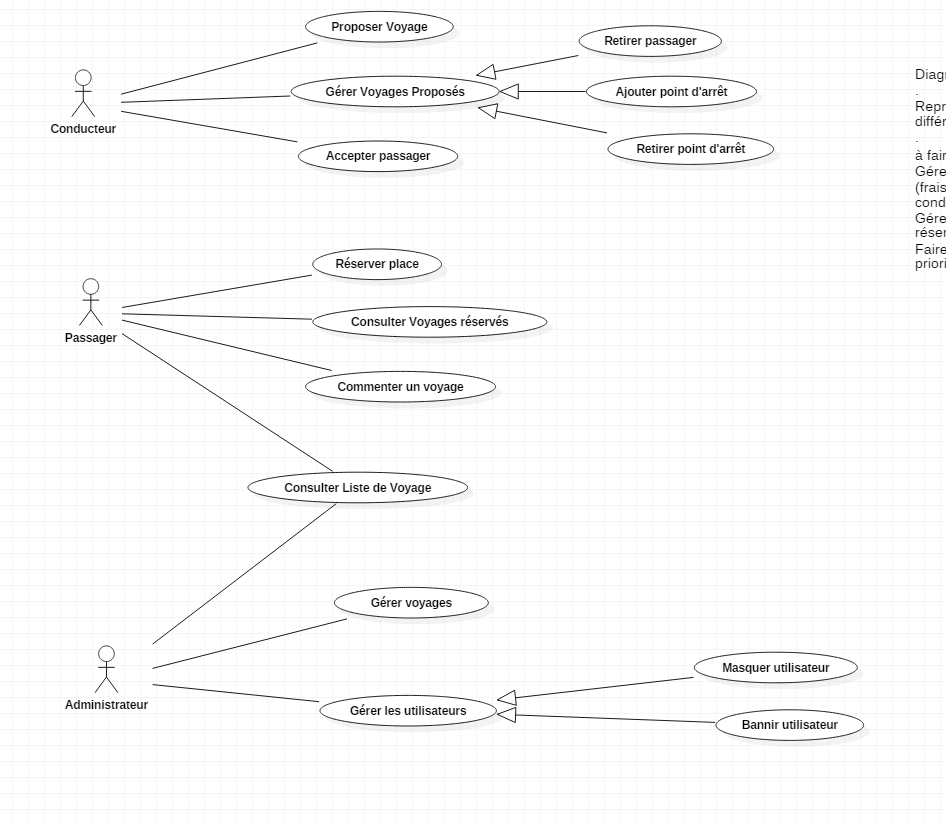
Le diagramme de contexte statique représente la liste des « acteurs » qui rentre en contact avec l’application. Cela permet de m’avoir une première approche de comment l’application va être gérée et à quelle portée.

Ici nous pouvons donc voir que seuls 3 types de personne seront en contacte avec l’application soit les passagers, conducteurs et administrateurs.

## Diagramme des use case

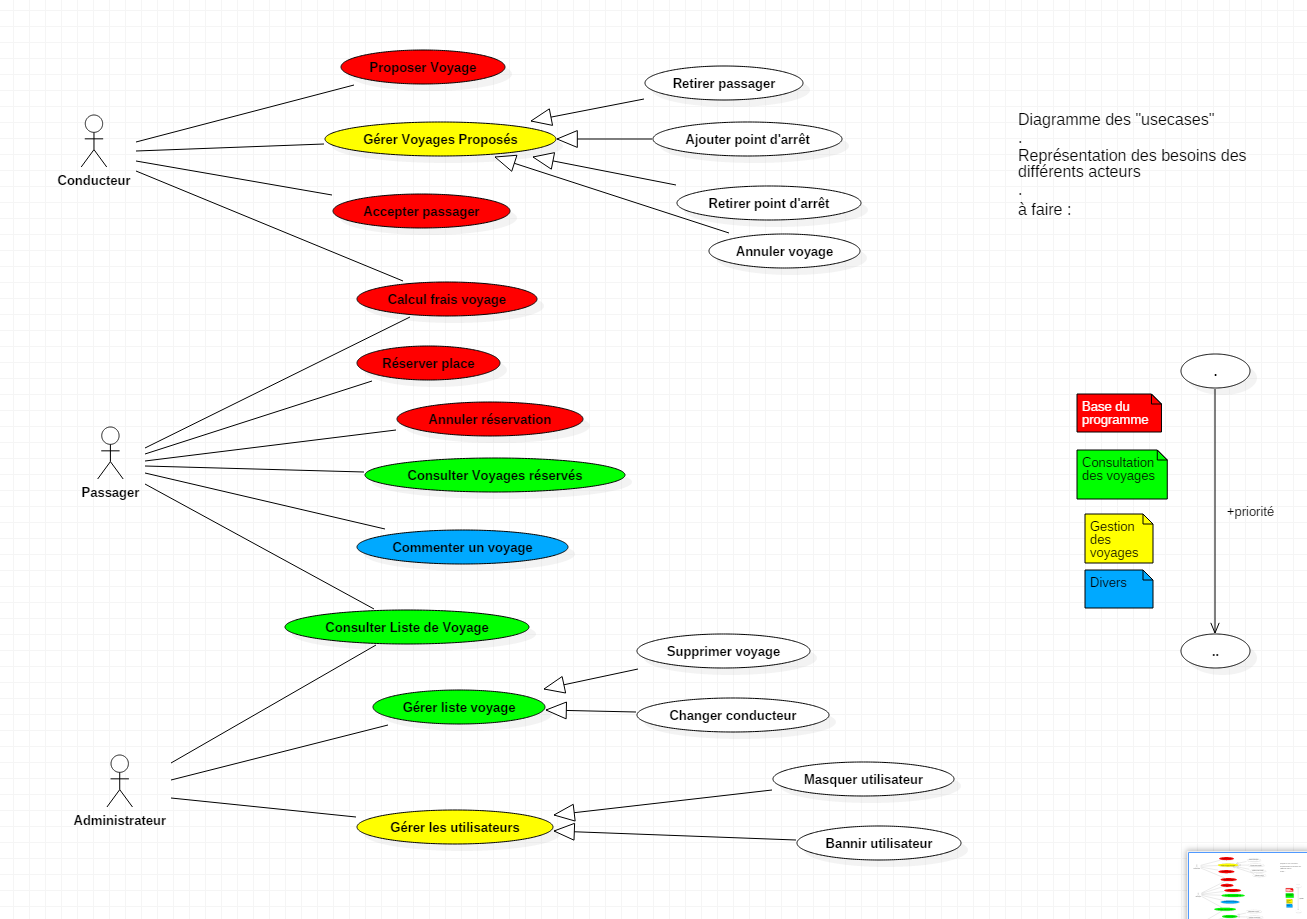
Le diagramme « use case » représente toutes les fonctionnalités disponibles de l’application. Ce digramme permet de visualiser les éventuelles méthodes à ajouter dans le diagramme de classe.

## 1er jet



Ce diagramme n’est pas complètement mauvais mais il manque des choses notamment la gestion de la réservation.

## Version finale



Sur le diagramme ci-dessus nous pouvons donc voir que les conducteurs peuvent donc proposer un voyage et seront en charge de l’édition de ce dernier que ce soit pour retirer, ajouter ou accepter des passagers.

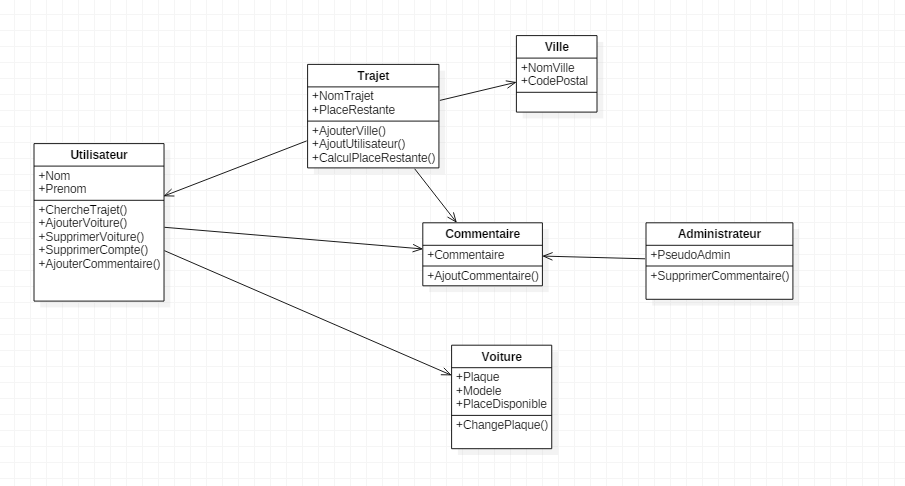
Les passagers peuvent quant à eux se rajouter dans un trajet proposé par le conducteur. Ils peuvent aussi bien évidemment consulter la liste des voyages et même commenter les voyages effectués.

Les administrateurs eux ont un rôle de contrôle et de maintien du site. Si les utilisateurs commettent un abus, ils peuvent les bannir ou supprimer un voyage.

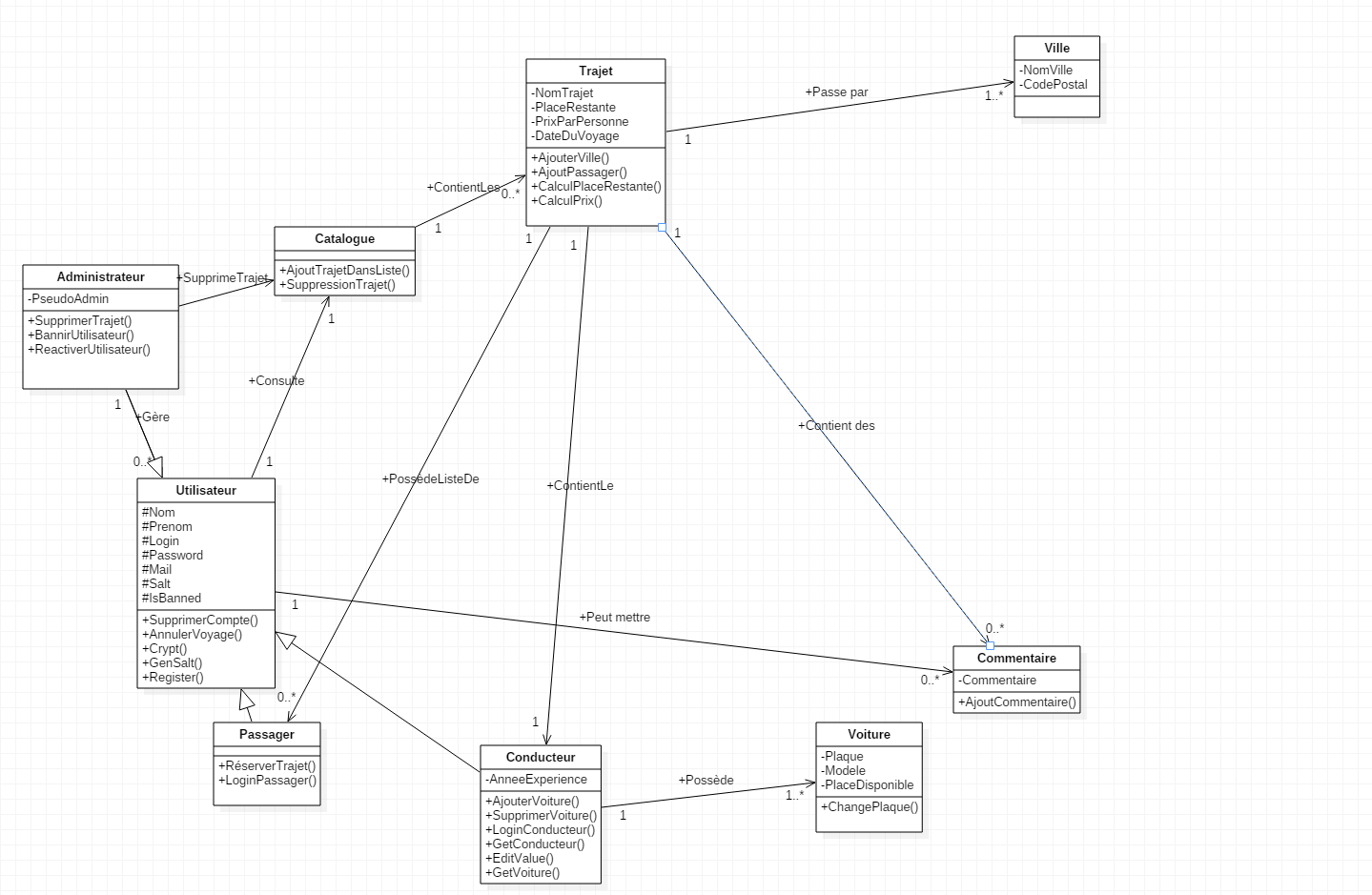
## Le diagramme de classe

Le diagramme de classe est un schéma qui représente l’ensemble des classes, attributs, méthodes et les liens entre ses dernières. Cela donne un visuel direct et le schéma de pensée de comment l’application va être réalisée. C’est le diagramme qui représente un peu le « plan de montage » du programme.

## 1er jet



Voici le 1er jet du diagramme de classe. Plusieurs erreurs son présente comme le manque de conducteur et de lien d’utilisateur vers trajet. Sans lien impossible pour un utilisateur de consulter la liste. Cette version était prévue pour qu’un utilisateur puisse être passager ET conducteur. Ce qui fut changer par la suite.

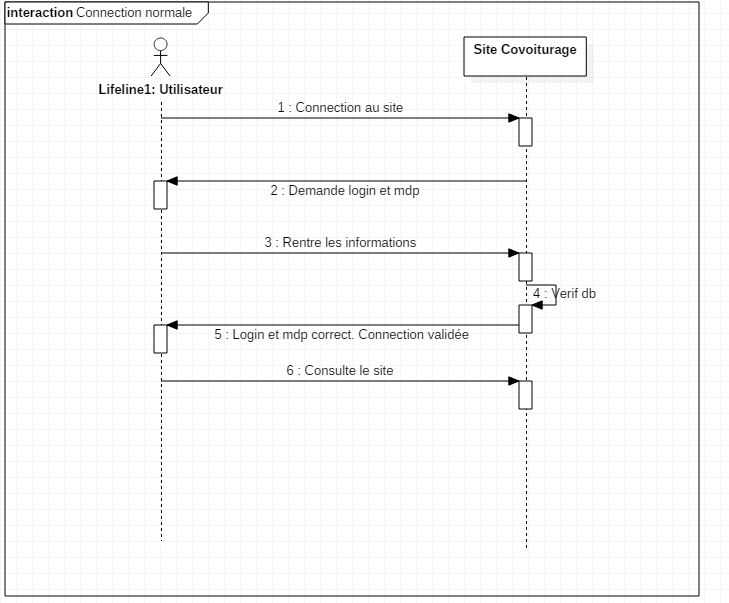


Le diagramme démarre au niveau des utilisateurs qui peuvent être soit des conducteurs soit des passagers ou même des administrateurs. Un héritage a été fait, car les 3 types d’utilisateurs regroupent des informations communes comme le nom, prénom ou le login qui ici représente-le « pseudo ». Les conducteurs eux, peuvent avoir une ou plusieurs voitures.

La classe catalogue représente un catalogue contenant la liste des trajets. Il peut être consulté par tout type d’utilisateurs.

Les trajets représentent les « voyages » qui sont créés par les conducteurs. Le prix est calculé automatiquement et contient des « villes » qui sont des points de passage. Chaque trajet peut être commenté par les utilisateurs qui ont participé.

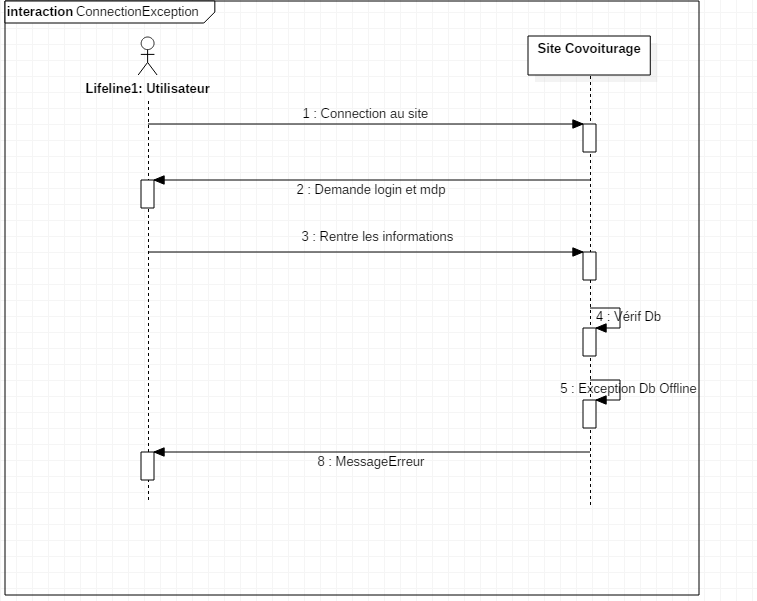
## Le Lifeline



Ce diagramme nous permet de définir toutes les méthodes possible et utilisable entre deux classes.

Ici nous pouvons voir les interactions entre les utilisateurs et l’interface principale du site dans le cadre d’un fonctionnement normal. L’utilisateur va sur la section connexion du site. Rentre les champs login et mdp. Le programme va vérifier dans la base de données et valider la connexion de l’utilisateur.

Maintenant, prenons le même schéma mais avec une exception.



Ici nous pouvons donc apercevoir un cas d’exception possible que devrait être géré par le programme pour ne pas juste renvoyer un message d’erreur incompréhensible à l’utilisateur.

# Conclusion

## Zaretti Quentin

Ce projet covoiturage plus facile pour moi que le projet précédent. L’analyse semblait plus claire et facile à mettre en place. Le travail en groupe a été bref mais efficace et les taches bien réparties malgré quelques soucis avec le github. Encore une fois le projet a permis d’aborder des points non vus en classe et en apprendre aussi plus sur le travail d’équipe.

## Bol Axel