# Documentation Technique

## Réflexions initiales technologiques

Frontend :

- HTML5/CSS3 (Bootstrap 5.3)
    - HTML, ou HyperText Markup Language, est le langage standard et incontournable utilisé pour créer et structurer le contenu des pages web.
    - Bootstrap est un framework CSS open source populaire. Cette collection de composants et de styles pré-conçus en système de grille permet de créer rapidement des sites web responsives et mobiles, et d'avoir ainsi plus de temps à consacrer au reste du projet.
- JavaScript (jQuery 3.7.1/AJAX)
    - Utiliser AJAX permet de rendre les pages web plus dynamiques et interactives, en mettant à jour le contenu sans avoir à recharger la page entière. Cela me permet aussi de démontrer ma compréhension des bases javascript et de l'utilisation de requêtes asynchrones.
    - La bibliothèque Chart.js nous sera utile pour afficher des diagrammes.

Backend :

- PHP 8.2

  PHP est un langage de script backend utilisé pour le développement web. PHP est flexible et compatible avec de nombreuses bases de données et systèmes d'exploitation, facilitant le travail en local et l'hébergement. De plus, sa communauté très active assure une abondance de support et de ressources.

- PDO (PHP Data Objects)
    - PDO est utilisé en tant qu'interface pour accéder aux bases de données depuis PHP. PDO utile la même syntaxe quel que soit le SGBD (MySQL, PostgreSQL, etc). De plus PDO offre une protection contre les injections SQL grâce aux requêtes préparées et aux paramètres liés.

- PDO me permet aussi de démontrer ma connaissance du langage SQL.

- XAMPP

  Utilisé pour créer un environnement de développement web local, gratuit et open source, idéal pour un projet PHP/MySQL. XAMPP est facile a installer et configurer. De plus les outils intégrés (shell et phpmyadmin par exemple) permettent de manipuler facilement une base de données.

- Bdd relationnelle (MySQL/MariaDB)

  Les bases de données relationnelles sont simple d'utilisation et permettent une gestion structurée et efficace des données. L'organisation en tables liés les unes aux autres est intuitive et le langage facile à maîtriser. Le système de clé et le principe ACID assurent l'intégrité des données. Les bdd relationnelles sont aussi rapides et efficaces au niveau performances. Ce qui correspond a l'aspect écologique de notre projet.

- Bdd NoSQL (MongoDB)
  - On utilisera MongoDB pour enregistrer des logs des objections. Cette fonction sert principalement à démontrer mes capacités à réaliser un CRUD en NoSQL.
  - Néanmoins, on peut imaginer une expansion de ce système, permettant de collecter toute sorte de données sur le site. Ce qui qui est parfaitement adapté à l'aspect souple et évolutif de MongoDB.
  - On pourrait même envisager de stocker toutes les info de covoiturages concernant la localisation en NoSQL ; MongoDB permet de traiter facilement des informations de géolocalisation, ce qui pourrait être très intéressant si on décide d'ajouter des capacité de localisation GPS à l'application.

- Bibliothèques/packages
  - composer/composer : Composer et son autoload sont utilisés pour gérer les dépendances.
  - mongodb/mongodb : necessaire pour communiquer avec la bdd en NoSQL.
  - vlucas/phpdotenv pour gérer le .env contenant les information de connection à MongoDB. Utilisé essentiellement pour tester une façon différente de config.php pour charger les variables de configuration.

    - phpmailer/phpmailer : PHPMailer est très utilisé et fiable pour l'envoi d'emails via smtp.
    - symfony/var-dumper : utile pendant le développement.

- Outils divers/autres
    - Gestion de version et dépôt : Git et GitHub.
    - Outil de gestion de projet : Trello, sous la forme d'un kanban contenant les liste de fonctions sous forme de check-lists contenues dans des cartes correspondant aux pages du site.
    - Production/Hébergement : Alwaysdata pour PHP/MySQL et MongoDB Atlas pour MongoDB. Ces deux services proposent des offres gratuites pour les petits projets.

---

## Configuration de l'environnement de travail

Programmes utilisés pendant le développement :

- draw.io pour divers diagrammes (MCD, utilisation, séquence, etc)
- Figma Desktop App (pour les maquettes)
- Visual Studio Code (v1.102) comme IDE (HTML/CSS, js et php)
    - Composer pour gérer les dépendances PHP
- Outils de développement des navigateurs Chrome et Firefox
- XAMPP (Apache, MariaDB, PHP 8.2) pour tester le relationnel localement
    - Dbeaver (v25+) pour la gestion de bdd
- MongoDB Community (v8.0.11) pour tester le noSQL localement
    - MongoDB Compass (v1.46+) pour la gestion de bdd
- Git (et GitHub pour le dépot)
- FileZilla Client pour transférer les fichiers de l'application chez l'hébergeur

Configuration :

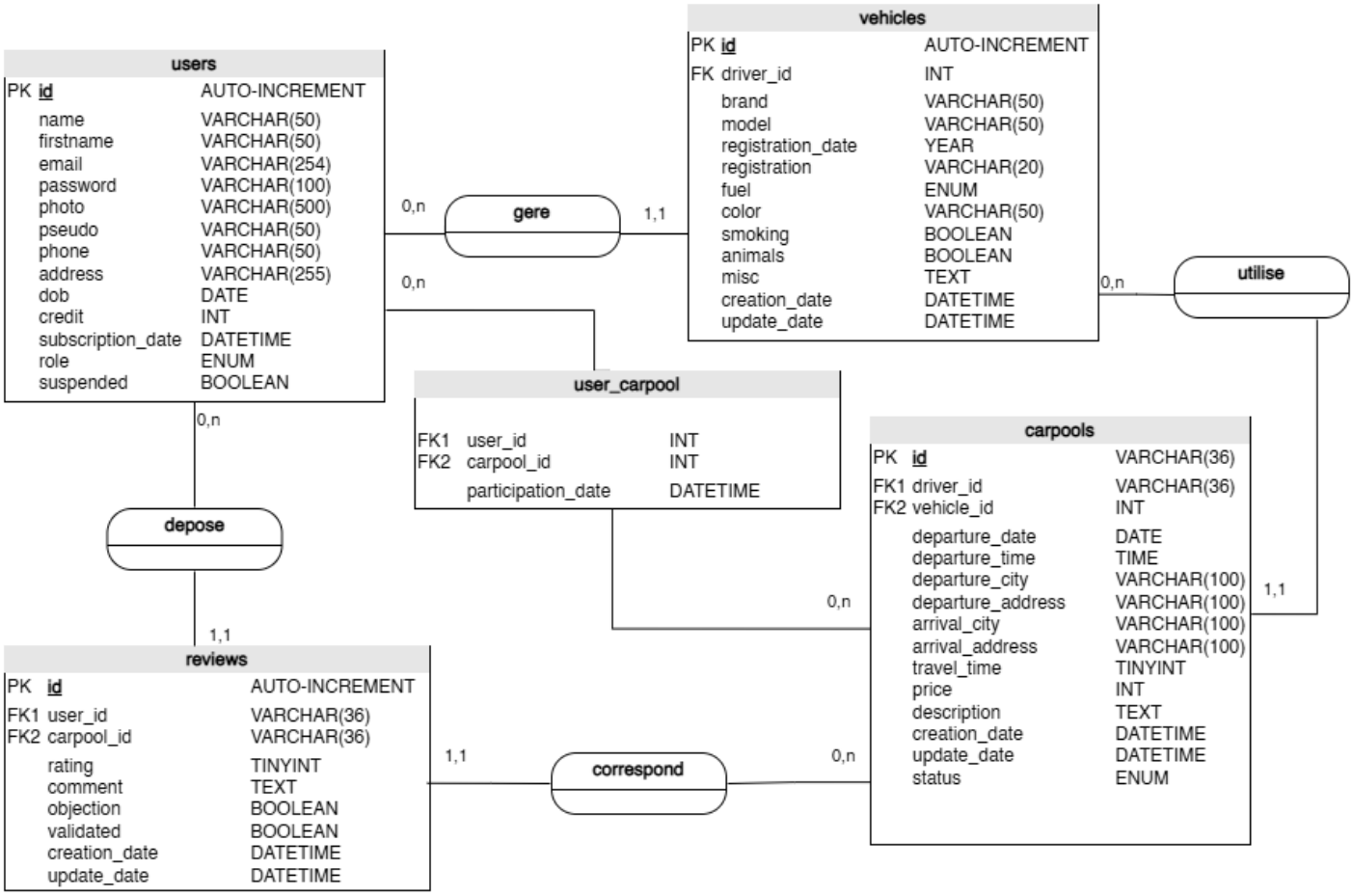
- Modification des variables d'environnement système de l'OS pour une utilisation plus simple des commandes de php, mysql, mongoDb et Composer.

- Installation d'extension facilitant le développement dans l'IDE : W3C Web Validator (html), PHP IntelliSense, Composer, GitHub Pull Requests/Repositories, etc.
- Activation des extensions PHP nécessaires (en dé-commentant dans php.ini) : pdo_mysql, mongodb, curl, fileinfo, etc.
- Configuration des bdd via scripts SQL.

---

## Modèle conceptuel de données (ou diagramme de classe)

On utilise deux types de bases de données. La base de données NoSQL MongoDB ne contient qu'une collection de logs d'objections des utilisateurs et des décisions les concernant par les employés pour que l'admin les consultes.
La base de donné SQL est organisée de la manière suivante :




---

## Diagramme d’utilisation et diagramme de séquence

Diagramme de cas d’utilisation :

Il n'y a que trois acteurs, correspondant à trois rôles: utilisateur, employé et administrateur. Néanmoins, pour évité un diagramme trop confus, on aura un visiteur à l'origine d'un futur utilisateur (un visiteur qui veut s'inscrire) et un utilisateur (un visiteur qui s'est connecté).

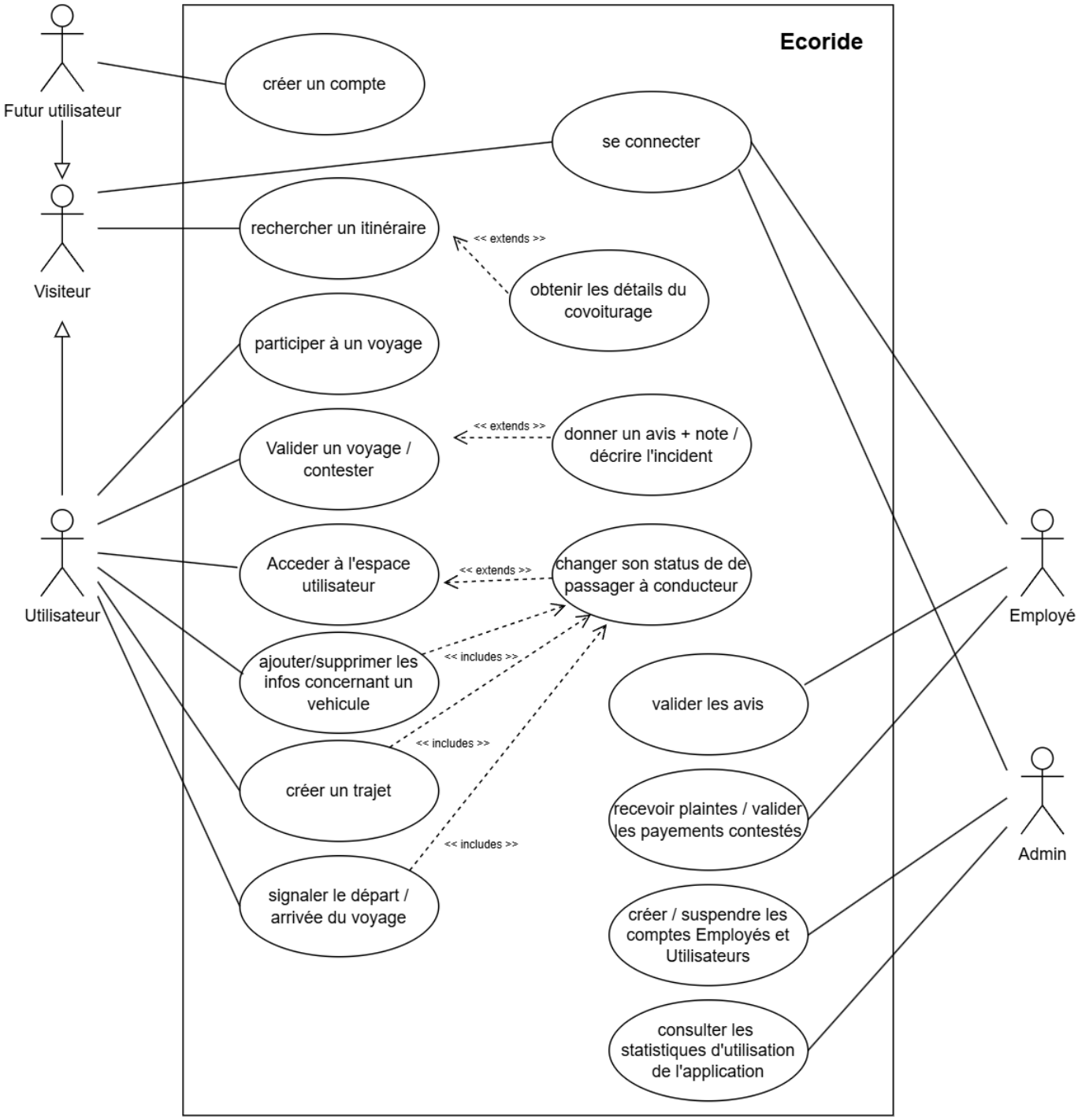
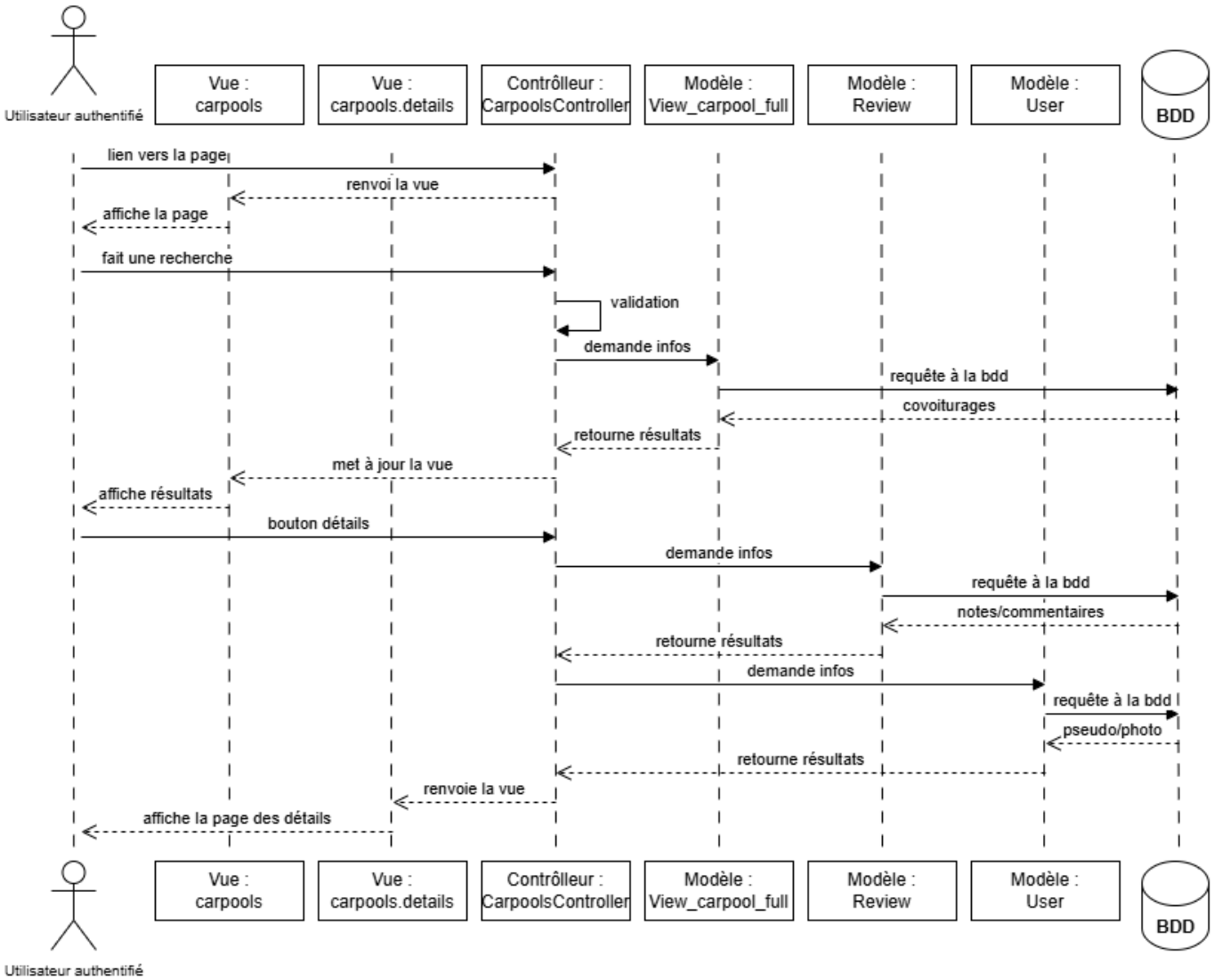


Diagramme de séquence :

On prendra l'exemple d'un utilisateur qui veut des informations précises sur un covoiturage particulier. Ce diagramme illustre l'architecture MVC de notre projet.




---

## Déploiement de l'application

Les hébergeurs :

- Alwaysdata propose une large gamme de fonctionnalités pour l'hébergement web et la gestion de bases de données. Alwaysdata est compatible PHP/js et permet de gérer facilement notre bdd MySQL. De plus, ce site propose une offre gratuite (256 Mo).
- MongoDB Atlas nous permet d'héberger la bdd NoSQL. Ils proposent une large

documentation multi-langue, y compris spécifiquement pour utiliser la package PHP `mongodb`. Ils ont aussi une offre limitée gratuite.

- On créé une adresse sur Google Mail, qui propose un accès gratuit aux serveurs SMTP pour 100 emails par jour.

Configuration :

- Alwaysdata :
  - On créé le site (option PHP spécifiquement): `tony-s-pro.alwaysdata.net`.
  - On force le HTTPS pour des raisons de sécurité.
  - On doit autoriser l'accès distant par SSH pour installer `mongoDB` (`ad_install_pecl mongodb`) avec pecl via la shell.
  - On ajoute les modifications a apporter au `php.ini` pour activer les extensions nécessaires (notamment `extension=/home/tony-s-pro/mongodb-8.2.so`).
  - Optionnellement , la config du serveur Apache également (on utilise les fichiers `.htacess`).
  - On change les constantes dans `config.php` (racine du site, etc).
  - On récupère les identifiants pour FileZilla.
  - On créé la bdd MySQL.
    - On créé un utilisateur avec les droits administrateur.
    - On récupère les identifiants et on les ajoute à notre fichier `config.php`.
- MongoDB Atlas :
  - On créé notre Cluster0 dans la timezone appropriée.
  - On créé un utilisateur avec les droits administrateur.
    - On récupère la chaîne de connexion et on l'ajoute à notre fichier `.env`.
  - En prod, on autorise l'accès au réseau depuis l'IP 0.0.0.0 (i.e. n'importe quelle adresse).
- Gmail :
  - On autorise le 2FA pour obtenir un mot de passe d'application qui ira dans `config.php`.

Finalisation :

- On transfert l'application vers le dossier `www` sur Alwaysdata en utilisant FileZilla Client.
- On initialise les bdd avec Dbeaver.
- On teste l'application.