Specifications techniques
Auto 95Clean

V0.1

Mehdi NAOUI 2026

1. Cadre du projet

Résumé

Ce document décrit l'architecture technique envisagée et les principes généraux de mise en œuvre du projet AUTO 95 CLEAN.
Il ne fige pas les choix techniques définitifs et pourra évoluer en fonction des validations client et des contraintes métier.

Il est très importants de noté que les choix sont susceptibles d'évoluer.

2. Vue d'ensemble de l'architecture

a. Architecture logique

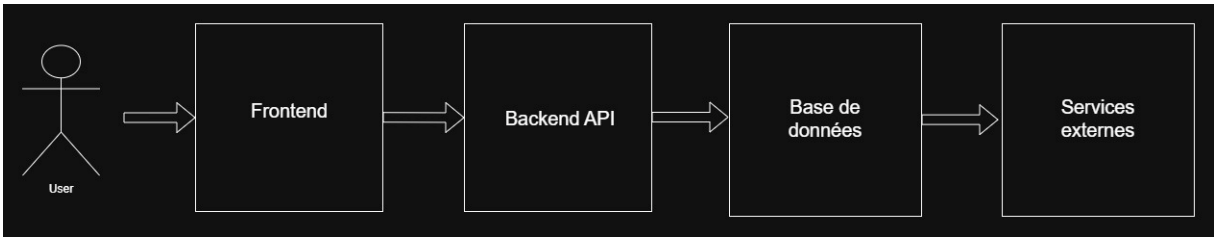
L'application se décline en **cinq sous-ensembles**.

L'application web client assure l'accés aux services de réservations pour les clients.

L'application web administration assure l'accès ainsi que la gestion pour l'administrateur.

Le backend ce présente sous forme d'une API accessible via des « Endpoint ». Ces « Endpoint » sont des adresses URL. Le frontend utilise ces « Endpoint » pour opérer les actions nécessaires au bon fonctionnement de l'application.

Enfin les services de mail ainsi que les paiements sont assurés par des services externes.

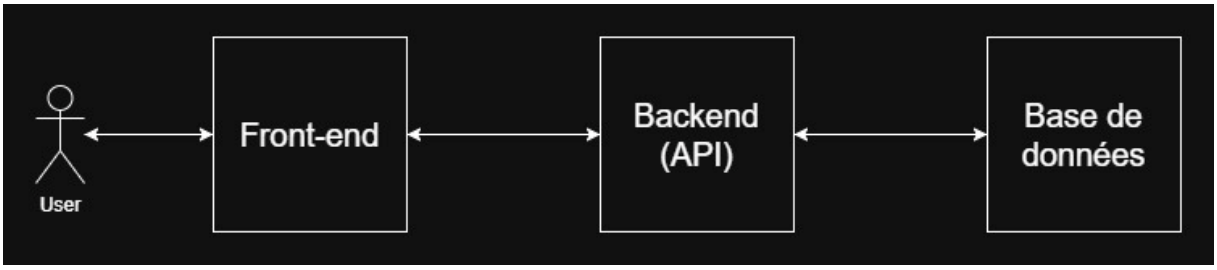


3. Frontend (principes)

a. Rôle

Le frontend ce présente sous forme de page html accessible via un navigateur web. Il sert d'interface utilisateur coté client ainsi que pour les agents de nettoyages ou bien l'administrateur.

Le frontend communique avec le backend via une API. Cette API permet au frontend de réaliser des opérations de lecture, mise à jour et suppression sur les bases des données.



b. Technologies envisagées

React est la technologie envisagée pour la réalisation du frontend.

TypeScript est le langage de programmation envisagé pour le développement du frontend.

L'approche **mobile-first** a été choisis avec un **responsive desktop** pour la navigation sur ordinateur de bureau.

4. Backend (principes)

a. Rôle

La partie backend réalise les opérations de CRUD sur la base de données. Ces opérations de CRUD sont nécessaires à la gestion :

- Des utilisateurs
- Des rendez-vous
- Des créneaux
- Des points de fidélités et du parrainage
- Et enfin l'interface avec les services externes

b. Technologies envisagées

L'environnement envisagées pour la réalisation du backend est **Node.Js.**

NestJs est la technologie envisagée pour la réalisation du backend.

TypeScript est le langage de programmation envisagé pour le développement du backend.

Enfin l'API utilise une architecture de type **REST**.

5. Base de données (principes)

a. Type

La base de données est dite relationnelles. Chaque entité est liée à une autre.

Chacune des ces entités possèdes des attributs qui pourront être lues et mises à jour.

b. Entités principales (non détaillées)

Voici une liste des entités potentielles :

- Utilisateur
- Rendez-vous
- Créneau
- Service/Formule
- Paiement
- Points de fidélité
- Parrainage

6. Paiement (intégration externe)

a. Principe

Le paiement se fait via une solution externe type SumUp ou bien Qonto.

Lorsque la formule, le type de véhicule et le créneau horaire sont choisis par l'utilisateur, au moment de payer, une API venant du de la solution (SumUp par exemple) ce présente sur la page pour effectuer le paiement.

b. Rôle du système

Le système a pour le rôle d'associer un de paiement à un rendez-vous.
Le système a pour rôle d'attendre la confirmation de la solution externe et de présenter la réussite ou bien l'échec du paiement.

Il stock le statut du paiement.

c. Statut du paiement

- Payé
- Echec
- Remboursé
- Annulé

7. Gestion des mails

a. Cas d'usage

Les mails ont pour rôle principales de confirmer les réservation à l'utilisateur et de communique les modalités de gestion de compte (mot de passe, etc)

b. Principe technique

Envoi via service SMTP ou prestataire tiers(a décider).

Dans le cas de l'envoi directement via le backend, il existe un module dédié nommé « MailModules » disponible au sein de NestJs.

La seconde solution sera un envoi via un service externe comme « Amazon » SES ou bien « Sensgrid ».Elle sera appelé par le backend.