

# DOCUMENTATION TECHNIQUE

## REFLEXION INITIALE TECHNIQUE SUR LE PROJET

Le cahier des charges du client pour le développement de l'application ecoRide ma conduit à utiliser plusieurs technologies nécessaires à l'élaboration d'une solution adaptée aux besoins du client.

### HTML5

En tant que base de toute application web, j'ai utilisé HTML5 pour structurer le contenu de toutes les pages web.

### SCSS (SASS)

Pour mettre en forme l'application, j'ai pris le parti d'utiliser SCSS plutôt qu'un CSS classique pour bénéficier de l'utilisation des variables, facilitant la maintenabilité et la modularité du code. De plus, SCSS me permet de customiser Bootstrap.

### BOOTSTRAP 5.3

En tant que framework CSS, Bootstrap permet de gérer de façon plus intuitive le côté responsive d'une application (devenu primordial pour les habitudes des utilisateurs actuels), ainsi que des composants utilisables facilement. Comme mentionné dans le paragraphe précédent, les couleurs basiques de Bootstrap peuvent être surchargées via des variables SCSS pour correspondre à la charte graphique du projet.

De plus, je choisis également Bootstrap pour la qualité de sa documentation sur son site, ainsi que la qualité et la quantité de sa bibliothèque d'icônes.

Enfin, je choisis d'installer Bootstrap via npm afin de limiter les requêtes nécessaires à l'utilisation de Bootstrap en cdn (éco développement).

### JAVASCRIPT ES6+

Pour concevoir une application agréable à l'utilisation et limiter les rechargement de pages pour mettre à jour l'affichage, j'opte pour le langage JavaScript qui, relié au HTML, permet d'en modifier le contenu de façon dynamique pour répondre aux actions de l'utilisateur.

Je choisis ce langage également pour reprendre un système de router déjà utilisé dans un autre projet et développer mon application en SPA (Single Page Application). Cela permet de fluidifier le comportement de la navigation et de

faciliter la maintenance car certaines sections, comme le header et le footer, se répètent à chaque page et ne sont codées qu'à un endroit.

Pour supporter ce système, j'ai ajouté un fichier `_redirects` pour assurer le déploiement sur Netlify et gérer les erreurs de direction.

### JSON

J'opte également, à défaut de développer la partie back-end, pour l'utilisation du langage JSON pour un fichier `data.json` qui fait office de simulation de base de données pour une utilisation en démo front-end uniquement. J'ai conscience des limites de cette pratique, notamment sur la sécurité des données personnelles, mais cela reste uniquement pour la démonstration de la logique applicative.

### NETLIFY

Pour déployer l'application, j'opte pour l'utilisation de Netlify, qui propose un service de déploiement simple et efficace pour un site statique. En l'occurrence, lorsque le back-end sera développé, Netlify ne suffisant plus, je m'orienterais vers une plateforme telle qu'Heroku.

## CONFIGURATION DE L'ENVIRONNEMENT DE TRAVAIL

### MAQUETTAGE

#### DRAW.IO

J'ai utilisé draw.io pour schématiser le diagramme d'utilisation de l'application. C'est un logiciel très intuitif qui permet la réalisation de schémas et de diagramme facilement.

#### FIGMA

Pour le maquettage des visuels de l'application, que ce soit les wireframes et les mockups, j'ai utilisé le logiciel Figma, avec différents plugins. Figma est extrêmement utile et complet pour créer des maquettes visuelles et également dynamiques pour simuler une utilisation d'une application. Figma dispose de fonctionnalités permettant d'anticiper le côté responsive d'un projet.

Mes plugins utilisés sont principalement Bootstrap icons et Unsplash pour avoir des visuels compatibles avec Bootstrap pour le développement futur et pour utiliser des images libres de droit.

### CODE

Le développement s'est fait grâce au logiciel Visual Studio Code, qui est un IDE complet et intuitif permettant de personnaliser son environnement de travail facilement grâce à des extensions très pratiques et une intégration native de Git.

#### PRETTIER

Extension permettant d'indenter et de formater automatiquement le code.

#### LIVE SASS COMPILER

Extension permettant la pré-compilation du code écrit en SCSS pour générer le CSS.

#### AUTO RENAME TAG

Extension qui facilite ma manipulation du code HTML.

#### PHP SERVER

L'extension PHP Server permet de simuler un serveur PHP et me permet d'exécuter mon code sur un navigateur en local.

## GESTIONNAIRE DE DEPENDANCES

---

### NPM

Node.js et npm me permettent d'installer Bootstrap en local sur ma machine et le surcharger via SCSS. Cela crée un dossier node\_modules avec toutes les dépendances dedans, que j'ai choisi d'ignorer au dépôt Git (avec fichier .gitignore) pour ne pas surcharger le dépôt distant.

### TEST

---

### GOOGLE CHROME

J'utilise le navigateur internet Google Chrome pour effectuer les tests de rendu car il dispose de nombreuses extensions utiles. J'utilise également les outils de développeur pour tester des éléments visuels en direct avant de les implémenter dans mon code source. Cela me permet d'avoir un environnement de test complet.

## GESTIONNAIRE DE VERSIONS

---

### GIT

J'utilise Git, installé sur ma machine pour versionner mon projet. J'applique les bonnes pratiques Git de développement, à savoir une branche main, qui contient le code qui est pushé en production et une branche dev qui sert à faire le développement et les test, grâce à des sous-branches feature/\* pour chacune des fonctionnalités en cours de développement.

### GITHUB

Mon dossier versionné par Git est également poussé vers un repository distant sur GitHub qui permet un stockage cloud de mon projet et de toutes ses versions. De plus, il fait le lien avec le déploiement sur Netlify.

## DEPLOIEMENT

---

### NETLIFY

J'ai déployé l'application via Netlify, en utilisant un fichier \_redirects pour gérer le SPA avec mon router JavaScript. Le build est simple car aucune dépendance n'est à installer, le scss est précompilé sur ma machine et le repo GitHub contient le fichier main.css qui contient toutes les données d'affichage du code.

## MODELE CONCEPTUEL DE DONNEES

N'ayant pas intégré de réelle base de données à mon projet pour l'instant, je simule cela grâce à un fichier data.json qui est structuré comme suit :

```
{
  "trips": [
    {
      "id": 1,
      "pseudo": "Noémie",
      "note": 4.8,
      "photo": "../images/Avatars/profile-pic (1).jpg",
      "ville_depart": "Paris",
      "ville_arrivee": "Lyon",
      "date": "2025-12-01",
      "heure_depart": "05:00",
      "heure_arrivee": "09:40",
      "credit": 22,
      "places_disponibles": 1,
      "marque_vehicule": "Hyundai",
      "modele_vehicule": "Kona",
      "energie_vehicule": "Electrique",
      "preferences": ["Pause café possible", "Climatisation", "Bagages volumineux acceptés"],
      "annee_vehicule": 2020
    }
  ],
  "users": [
    {
      "id": 1,
      "nom": "NomTest",
      "prenom": "PrenomTest",
      "pseudo": "PseudoTest",
      "mail": "test@yahoo.fr",
      "password": "Mot2Passe2Test@",
      "token": "b15f9b905dad07a4411ed644c06560f3510b9a80",
      "role": "admin"
    }
  ]
}
```

Cette capture d'écran montre la structure des données avec, pour l'instant, les trajets et les utilisateurs, et tous les paramètres nécessaires à la mise en place de l'application front-end.