



# **BUT2 Initiaux**

## **SAÉ4.Real.01 : Développement d'une application complexe**

### **Partie : R4.Real.12 | Automates et Langages**

Rapport d'analyse d'automates

LEPORT Clovis  
MARTEL Floran  
GROMARD Alexys  
JOUAULT Lancelot  
CHEVREUX Arthur

<b>Introduction.....</b>	<b>2</b>
A. Présentation de NaoLibre.....	2
B. Présentation du document.....	2
<b>1. Nos fonctionnalités.....</b>	<b>2</b>
A. Présentation des fonctionnalités.....	2
B. Granularité des micro-services.....	3
<b>2. Modélisation et Conception.....</b>	<b>4</b>
A. Micro-service de connexion.....	4
B. Micro-service d'ajout d'arrêts favoris.....	5
C. Micro-service ajout d'encombrement.....	6
D. Micro-service de demande de temps d'attente pour un arrêt.....	7
E. Micro-service moyenne d'encombrement sur une heure.....	8
<b>3. Les automates dans notre projet.....</b>	<b>9</b>
<b>4. Conclusion.....</b>	<b>9</b>

# Introduction

## A. Présentation de NaoLibre

Dans le cadre de notre deuxième année de BUT, nous avons dû développer un site web répondant à un problème actuel. Nous avons décidé de simplifier la vie des utilisateurs de transports en commun au sein de Nantes. Pour ce faire, nous avons voulu créer un site web permettant de donner l'encombrement de chaque transport pour que les personnes qui sont repoussées par l'utilisation des bus de par sa surpopulation, puissent constater que ce n'est pas toujours le cas.

Notre site web permet aux utilisateurs de pouvoir consulter le temps d'attente des bus et tram de leurs arrêts favoris et ils peuvent aussi observer les moyennes d'encombrement par heure sur chaque ligne. Les utilisateurs sont récompensés par un système de points à chaque fois qu'ils évaluent une ligne. De cette façon, ils se sentent plus impliqués et prennent le temps de nous aider à chaque fois.

Afin de développer ce site, nous avons utilisé des technologies variées, à savoir : React pour le développement de la partie client, Express et Node.js pour la partie serveur, ainsi que MongoDB et Mongoose pour l'utilisation de la base de données.

## B. Présentation du document

Ce document a pour but de montrer le cheminement emprunté par notre équipe pour pouvoir réaliser des automates qui nous servent pour notre site et qui sont réutilisables pour d'autres projets. Pour cela, ce document est composé d'une première partie présentant nos fonctionnalités. Une deuxième partie expose une partie de nos micro-services sous forme d'automates. Une dernière partie explique l'intégration de nos automates dans le code.

# 1. Nos fonctionnalités

## A. Présentation des fonctionnalités

### **Service d'authentification :**

- Micro-service de connexion
- Micro-service d'inscription

### **Service de gestion de comptes :**

- Micro-service de modification d'informations (Prénom, Nom, Point, ...)
- Micro-service d'ajout d'arrêts favoris

#### **Service de requêtes API Naolib :**

- Micro-service de demande des lignes
- Micro-service de demande des arrêts
- Micro-service de demande de temps d'attente pour un arrêt

#### **Service d'encombrement :**

- Micro-service ajout d'encombrement
- Micro-service moyenne d'encombrement sur une heure
- Micro-service moyenne d'encombrement sur une ligne

#### **Service de gestion des récompenses :**

- Micro-service pour attribuer des points de récompense aux utilisateurs en fonction de leurs interactions avec le site.

## **B. Granularité des micro-services**

La granularité des micro-services fait référence à leur taille et à leur portée fonctionnelle. Dans notre système, nous avons déjà défini certains micro-services, mais nous pouvons les examiner de plus près pour évaluer leur granularité et identifier d'autres opportunités de décomposition.

En examinant ces micro-services, nous pouvons constater que certains d'entre eux pourraient être décomposés davantage pour une granularité plus fine. Par exemple :

- Le service d'authentification pourrait être étendu pour inclure des micro-services de récupération de mot de passe, de déconnexion, etc.
- Le service de gestion de comptes pourrait être divisé en micro-services distincts pour la gestion des informations personnelles, la gestion des préférences, etc.
- Le service de requêtes API Naolib pourrait être segmenté en micro-services distincts pour les requêtes de lignes, d'arrêts et de temps d'attente.

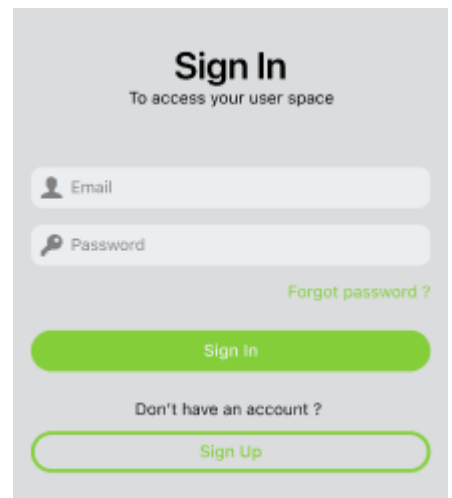
Une granularité plus fine offre une meilleure réutilisabilité, évolutivité et facilite l'isolation des erreurs. Cependant, il est important de trouver un équilibre entre la granularité et la complexité du système pour garantir une gestion efficace des micro-services.

## 2. Modélisation et Conception

Dans cette partie nous allons modéliser certains micro-services sous forme d'automates en montrant le côté client et le côté serveur pour chaque micro-service. Nos automates ne vont traiter que le micro-service en question. Ils seront donc déclenchés par une action et se termineront à la fin, cependant leur début et leurs fins n'ont aucun lien avec le début et la fin du serveur ou de l'utilisation client. Ce sont des fonctionnalités asynchrones.

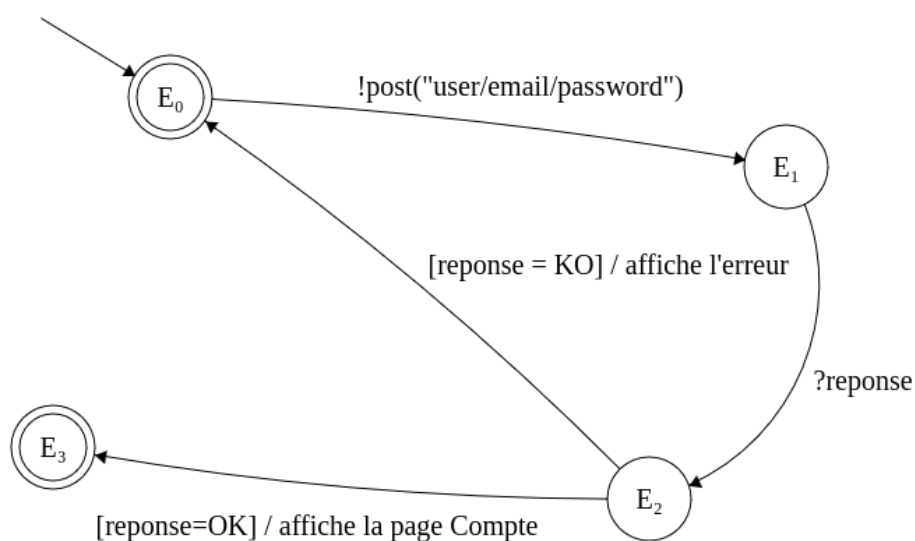
### A. Micro-service de connexion

Pour ce micro-service, on part du principe que l'utilisateur se retrouve sur la page de connexion et qu'il rentre ses données à savoir une adresse mail et un mot de passe. L'automate client débute lorsque le client appuie sur le bouton de connexion. L'automate serveur débute lorsqu'il reçoit la requête et les données de l'utilisateur à savoir son email et son mot de passe.

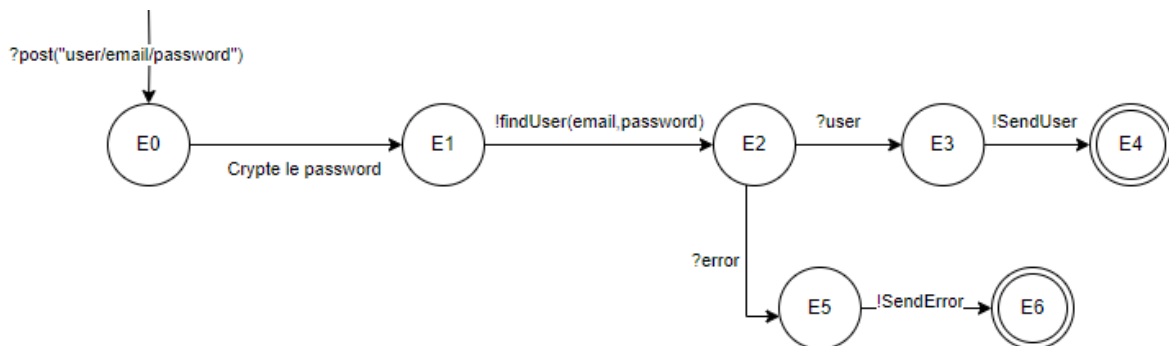


The image shows a 'Sign In' form with the title 'Sign In' and subtitle 'To access your user space'. It contains two input fields: 'Email' and 'Password'. Below the password field is a link 'Forgot password?'. There are two buttons: a green 'Sign In' button and a 'Sign Up' button outlined in green. At the bottom, there is a link 'Don't have an account?'.

Côté client :



Côté serveur :



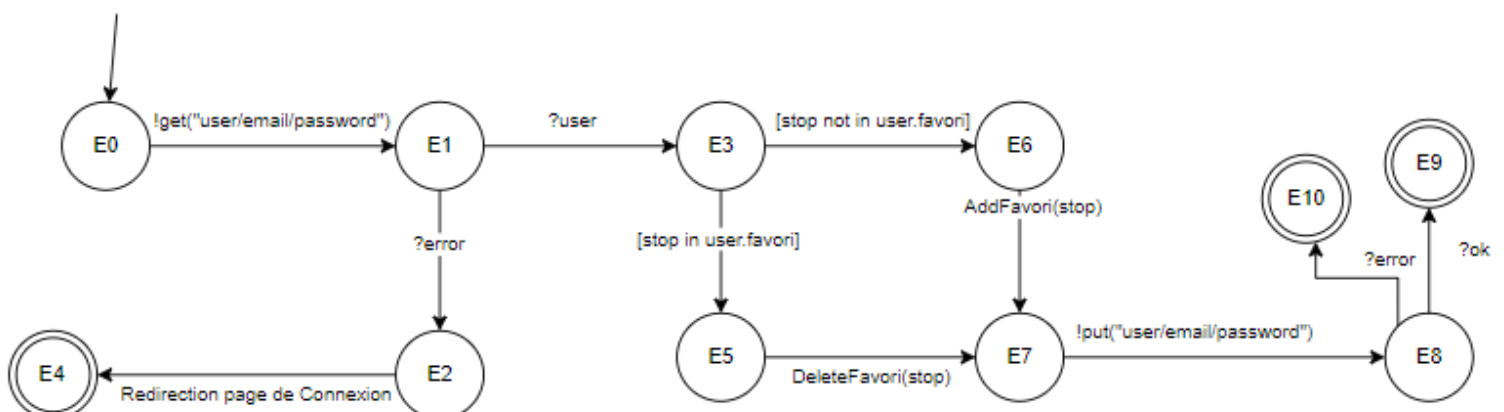
Pour vérifier si l'email et le password sont correcte nous utilisons `findUser` qui va faire un appel à la base de données et renvoyer l'utilisateur s'il a été trouvé ou une erreur

## B. Micro-service d'ajout d'arrêts favoris

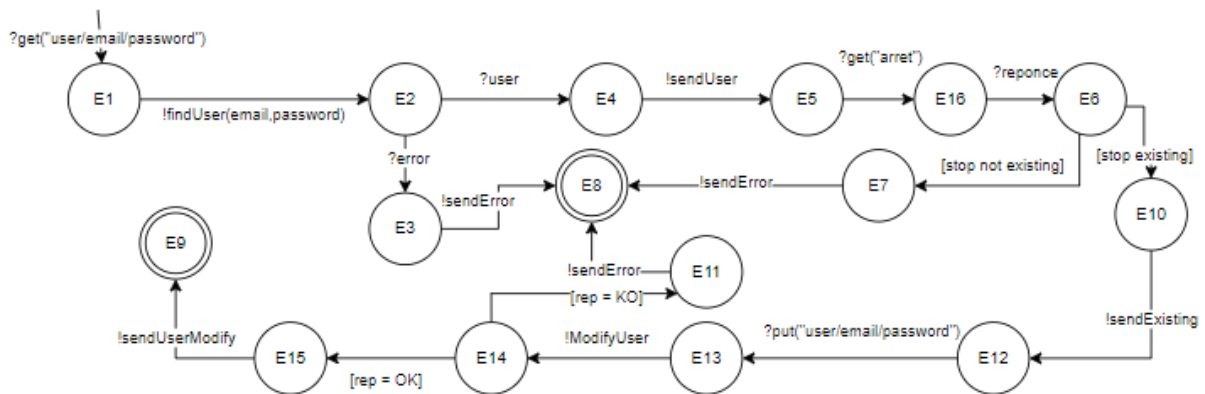
Pour ce micro-service, on part du principe que l'utilisateur se retrouve sur la page de recherche et a déjà cherché son arrêt. L'automate client débute lorsque l'utilisateur clique sur l'étoile d'ajout en favori. L'automate serveur débute lorsque le clic a été effectué.



Côté client :



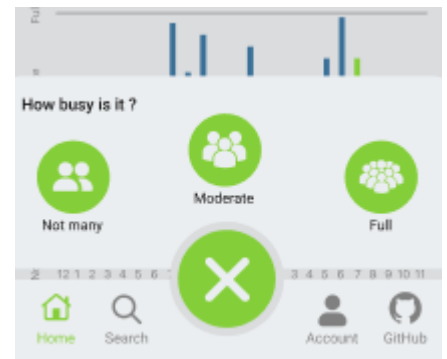
## Côté serveur :



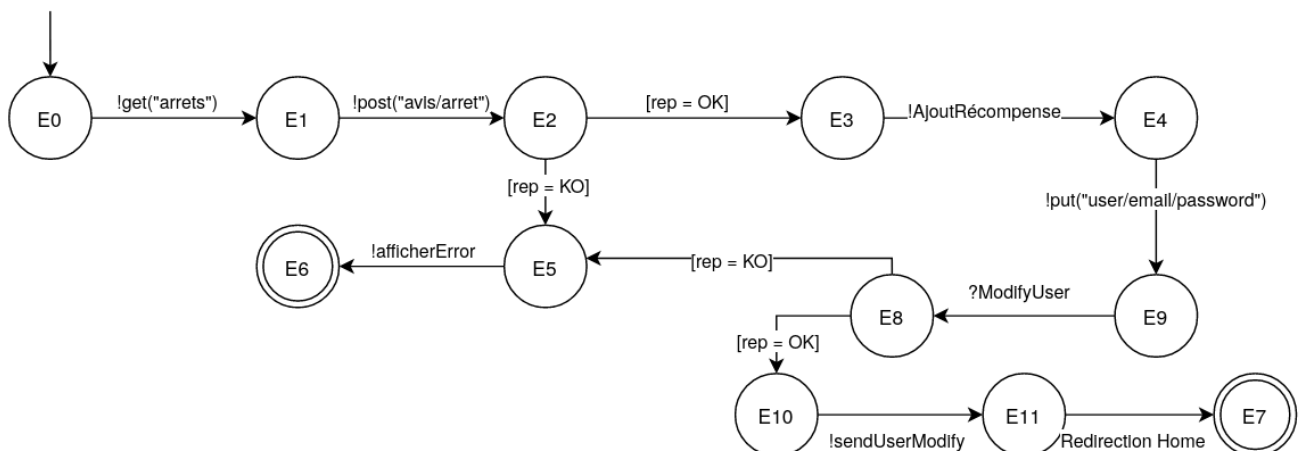
Dans la partie Serveur, notre automate commence par vérifier si l'utilisateur à modifier existe bien. Ensuite, il va demander tous les arrêts de Nantes à notre 1ère API qui est une API qui filtre l'API de TAN. Une fois les arrêts collectés, on vérifie que les arrêts en favori sont bien des arrêts qui existent, enfin, on met à jour l'utilisateur dans la base de données et s'il y a une erreur on l'envoie sinon on envoie l'utilisateur modifier.

## C. Micro-service ajout d'encombrement

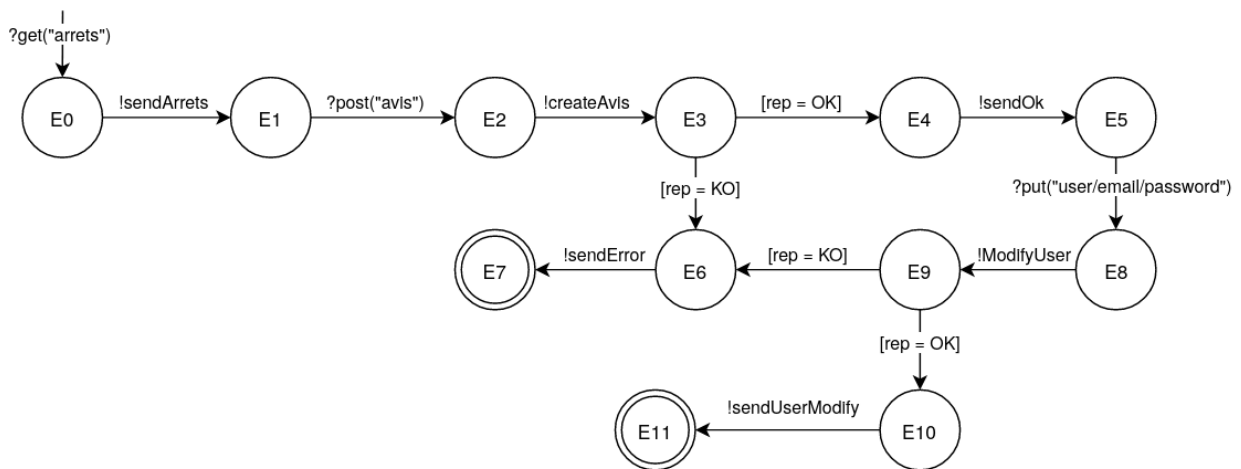
Pour ce micro-service, on part du principe que l'utilisateur se retrouve sur le site web. L'automate client débute lorsque l'utilisateur clique sur l'un des trois boutons d'encombrement. L'automate serveur débute lorsque l'utilisateur clique sur l'un des trois boutons.



## Côté client :



## Côté serveur :

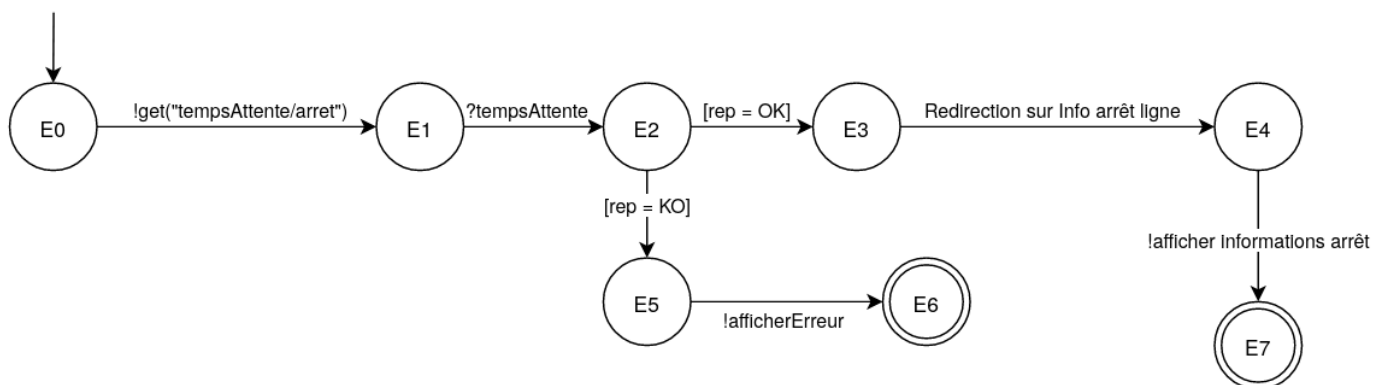


#### D. Micro-service de demande de temps d'attente pour un arrêt

Pour ce micro-service, on part du principe que l'utilisateur se retrouve sur la page de recherche et a trouvé son arrêt et sa ligne. L'automate client débute lorsque l'utilisateur clique sur un arrêt pour une ligne. L'automate serveur débute lorsqu'il reçoit une ligne et un arrêt.

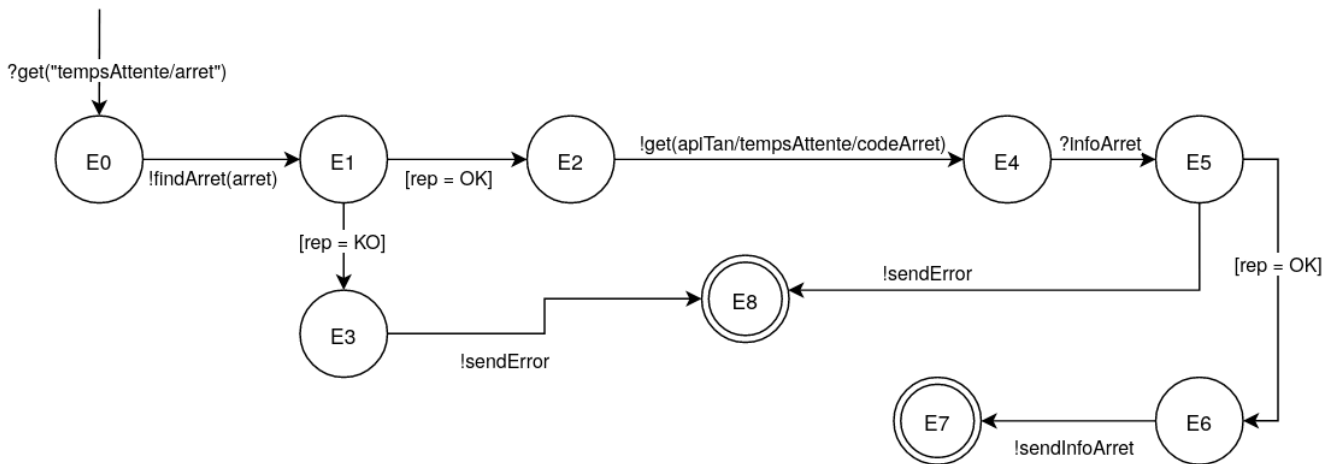


Côté client :



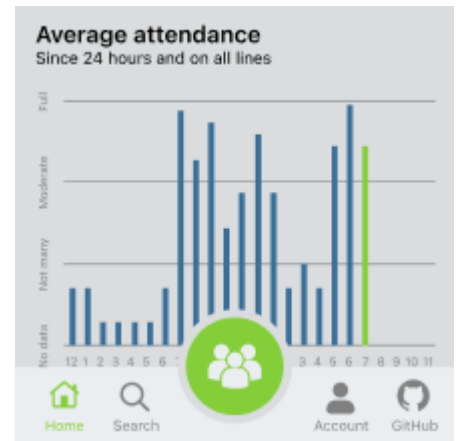
Côté serveur :



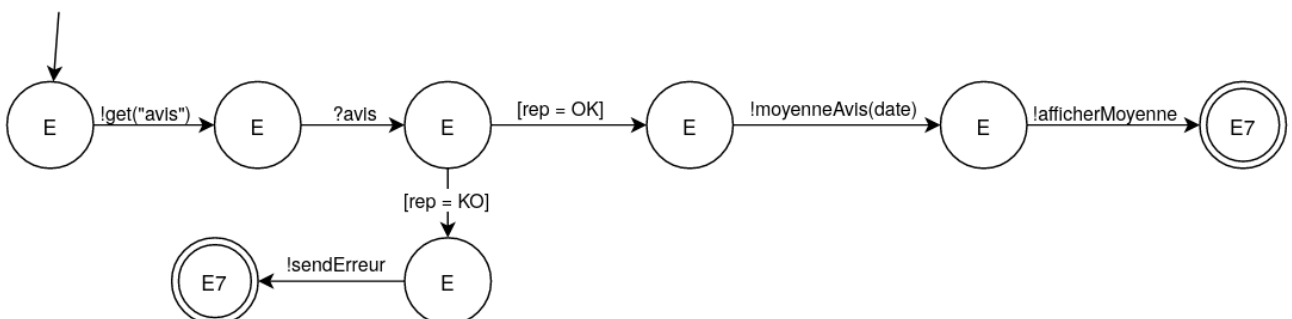


## E. Micro-service moyenne d'encombrement sur une heure

Pour ce micro-service, on part du principe que l'utilisateur se retrouve sur la page d'accueil. L'automate client débute dès que l'utilisateur lance se rend sur la page d'accueil du site web. L'automate serveur débute dès que l'utilisateur se rend sur la page d'accueil.



Côté client :



Côté serveur :

Le côté serveur n'a pas été développé car il va simplement faire un appel à la base de données une fois pour renvoyer tous les avis ou une erreur s'il y en a une.

### 3. Les automates dans notre projet

Les automates ont joué un rôle essentiel dès les premières phases de notre projet, en particulier lors de l'analyse et de la modélisation. Leur utilisation a permis une formalisation précise des spécifications en décrivant les différents états que le système peut traverser ainsi que les transitions possibles entre ces états. Cette approche a grandement contribué à clarifier les exigences du projet et à détecter d'éventuels conflits ou incohérences dans nos spécifications.

Par la suite, les automates ont été un pilier pour la création des tests de validation du comportement du système. En construisant un modèle d'automate déterministe, nous avons pu simuler le fonctionnement du système en soumettant diverses séquences d'entrées et en analysant les sorties correspondantes. Cette méthodologie a permis de valider le comportement du système avant son déploiement, garantissant ainsi l'adéquation avec les spécifications établies.

De plus, les automates ont servi de fondement à la construction d'une documentation détaillée. Cette documentation revêt une importance capitale, car elle facilitera la transition du projet vers d'autres technologies et permettra à d'autres équipes de reprendre le développement du site web de manière fluide et efficace.

Cependant, il est important de noter que malgré l'utilité indéniable des automates dans ces différentes étapes, leur développement a été influencé par des contraintes temporelles et de charge de travail. Ces contraintes ont limité la profondeur de l'analyse et de la modélisation, réduisant ainsi la possibilité d'exploiter pleinement le potentiel des automates dans notre processus de développement.

### 4. Conclusion

En conclusion, les automates ont représenté un élément essentiel dans la conception et la réalisation de notre projet NaoLibre. Leur utilisation dès les phases

d'analyse et de modélisation a permis de formaliser les spécifications, de clarifier les exigences et d'identifier les éventuels conflits ou incohérences. De plus, ils ont joué un rôle crucial dans la validation du comportement du système à travers la création de tests et la simulation des interactions.

Par ailleurs, les automates ont servi de base pour la documentation du projet, facilitant ainsi sa maintenance future et sa transition vers d'autres technologies ou équipes de développement.

Cependant, il est important de reconnaître que les contraintes temporelles et de charge de travail ont limité la pleine exploitation du potentiel des automates dans notre processus de développement. Malgré cela, leur intégration a apporté une valeur ajoutée significative à notre projet en termes de clarté des spécifications, de validation du comportement du système et de documentation.

Dans l'avenir, nous envisageons d'approfondir davantage l'utilisation des automates et d'intégrer des méthodologies plus robustes pour maximiser leur efficacité dans nos projets. Cette démarche permettra d'améliorer la qualité de nos produits et de renforcer notre processus de développement.

références :

Le cours de R4.Real.12, exemples de rendus de l'année dernière fournis pour la SAE4.Real.01