PRESTI LOUIS
SOUCHET MATHEO
COSSEC ALBAN
LEFEVRE LILIAN
MILLET ARTHUR

RAPPORT DE R412 - AUTOMATES ET LANGAGES

SAE - DÉVELOPPEMENT D'UNE APPLICATION COMPLEXE

EQUIPE_03_00

BUT INFORMATIQUE / IUT DE NANTES

2023/2024

11/04/2024

TABLE DES MATIÈRES

1- Introduction

1.0 - Avant-propos

Il est important d'expliquer les bases fondamentales de notre rapport avant de pouvoir continuer. Nous allons présenter deux applications, une application sur un site web "one page" que nous avons appelé "Where is my Good Stuff " ainsi qu'une application Android pour téléphone.
Dans ce rapport seront expliqués les divers micro-services de nos applications à travers des automates (ou machines à états). Ces derniers seront primordiaux pour pouvoir clairement expliquer les différentes fonctionnalités qui dirigent nos applications.

Nous aurons globalement deux types d'automates:

- Représentation des actions côté client (ce que l'utilisateur ne voit pas et ce qu'il envoie, sans forcément en avoir conscience)
- Représentations des actions côté serveur (ce que l'utilisateur ne voit pas, et ce qu'il n'est pas censé savoir).

1.1 - Présentation du Site web

Notre site web “Where is my Good stuff” est une “One Page” (Autrement dit, le client ne change jamais de page) qui permet aux utilisateurs de pouvoir accéder à une carte de Nantes (et ses alentours) présentant l’emplacement des toilettes ainsi que des vélos en libre-service (Bicloo).
Pour pouvoir récupérer nos données, nous avons fait appel à l’API de l’Open Data de Nantes Métropole. L’API Open Street Map sera également utilisée pour pouvoir trouver les coordonnées pour un nom donné (exemple: Bouffay, rue de XX, etc.)
Pour pouvoir afficher une carte ainsi que ses options, nous avons utilisé la Bibliothèque “Leaflet”.

Chaque utilisateur pourra par exemple noter chaque toilette sur 5 (système d’étoiles).
Les fonctionnalités seront présentées et expliquées dans la partie [2.1 - Présentation des fonctionnalités.]

1.2 Présentation de l’Application mobile

Notre application permet de trouver la recette de cuisine voulue par l’utilisateur. Voici comment se déroule le scénario nominal:

- Au lancement de l’application, l’utilisateur peut choisir un type de cuisine ou non (exemple: Européenne, française, Mexicaines...). Il doit obligatoirement spécifier un mot clé rappelant au minimum l’aspect principal de sa recette (exemple: beef, pasta, rice...) Il décide ensuite d’afficher le nombre de résultats qu’il désire (de 1 à 10).
- Il se trouvera donc sur une page où se situeront les nom des recettes . Ces noms sont obtenus grâce aux résultats de la requête envoyée à l’API que propose le site “Spoonacular”.
- Quand l’utilisateur cliquera sur une recette, une seconde requête sera envoyé pour avoir les informations générales de la recette (description, images, prix par personne ...)

2- Site web

2.1- Présentation des fonctionnalités

L'entièreté des fonctionnalités de notre site ne pourrons pas être présentées ici. Certaines se ressemblent sur le mode de fonctionnement et d'autres ne sont pas schématisables.

Voici les différents micro-services client/serveur que nous présenterons ci-dessous:

- Se créer un compte (pour pouvoir publier des avis)
- Se connecter
- Afficher les résultats avec ou sans filtrage des données
- Récupérer la localisation de l'utilisateur
- Donner un avis sur une toilette
- Construire un itinéraire
- Mise à jour de la carte selon la localisation nominale rentré dans un champ de texte.

2.2 Représentation des micro-services sous forme d'automates

2.2.1 Créer un compte

Coté client

Grâce à une fenêtre modale, l'utilisateur peut soit se connecter, soit se créer un compte.
Ici, nous partons d'abord du scénario "se créer un compte". Si le login existe déjà, alors une fenêtre d'alerte s'affiche à l'écran.

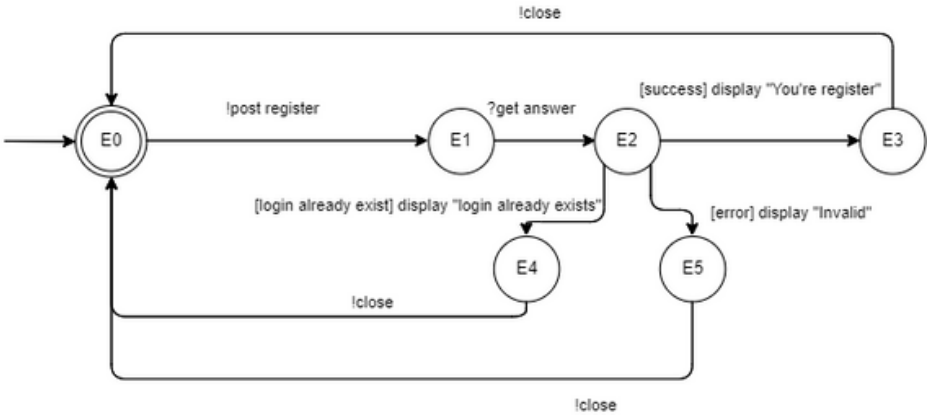

Fig1 - Automate_Client_Compte

Coté serveur

Quant au serveur, il vérifie si le login n'est pas déjà utilisé. Pour ça, il appelle notre base de données pour vérifier une égalité (avec la fonction findById()). Si le login choisi n'existe pas déjà, alors le compte est bien créer et est ajouté dans notre base de données. L'utilisateur en est informé. Sinon, une alerte est envoyé à l'utilisateur pour l'informer que son login existe déjà.

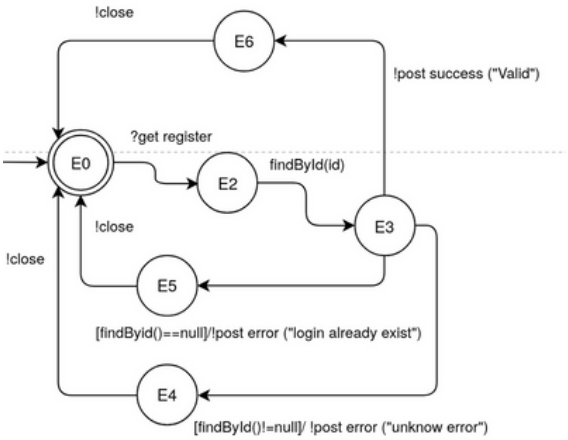

Fig2 - Automate_Serveur_Compte

2.2.2 Se connecter

Coté client:

Le fonctionnement est presque similaire au scénario précédent, lorsque le client veut se connecter.
Si jamais le login où le mot de passe est incorrect, une fenêtre d'alerte indique l'erreur de saisie. Sinon, l'utilisateur est informé qu'il est bien connecté.

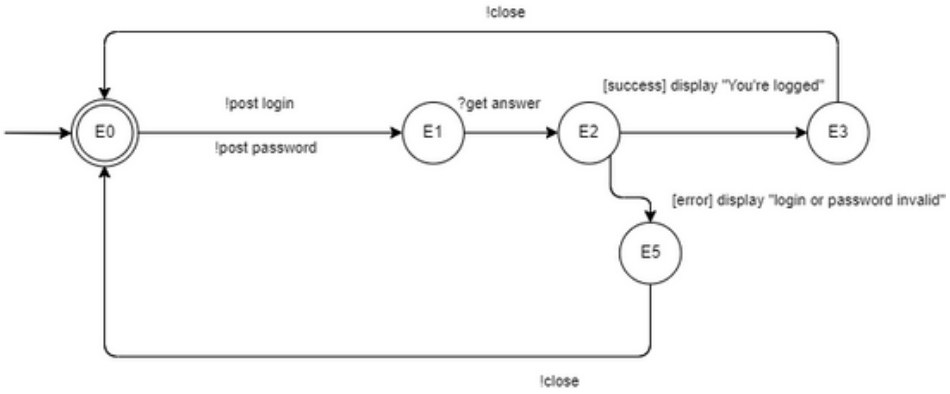

Fig3 - Automate_Client_Co

Côté serveur

Le serveur récupère le login et le password et vérifie qu'ils sont présents dans la base de données. Il enverra un message en conséquence au client, "success" si le login et password ne sont pas dans la base et "login or password invalid" sinon.

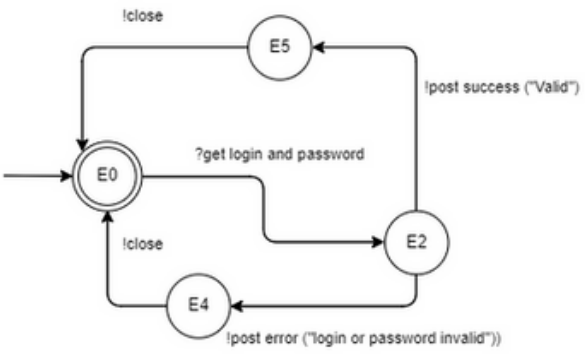

Fig4 - Automate_Server_Co

2.2.3 Affiche les résultats (avec filtres)

Coté client:

Lorsque l'utilisateur arrive sur la page cet automate sera appelé et pourra être rappelé selon les exigences de l'utilisateur.
En d'autres termes, l'utilisateur peut choisir d'afficher uniquement les vélos ou les toilettes par exemple, et c'est ce même micro-service qui sera utilisé.

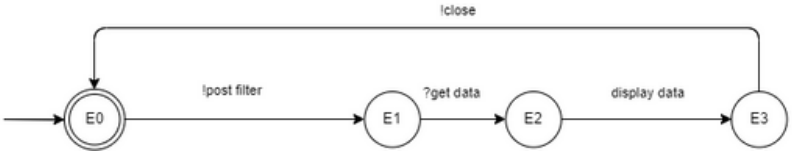

Fig5 - Automate_Client_Results

Côté serveur:

Le serveur attend que le client envoie ces exigences. Si il n'en a pas il envoie toutes les datas, sinon il filtres les données et les envoie au client.

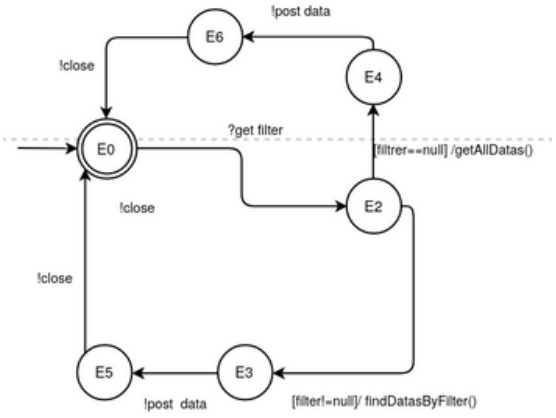

Fig6 - Automate_Server_Results

2.2.4 Récupérer/Partager sa/ma localisation

Coté client:

Le client web "Post" d'abord la localisation de l'utilisateur (ou non). Si jamais il refuse ou que sa localisation n'a pas pu être trouvée pour x raisons, alors il n'aura pas toutes les fonctionnalités et la carte s'affichera sur une localisation par défaut. Sinon la carte se met à jour en zoomant et en plaçant un marqueur sur la position de l'utilisateur.

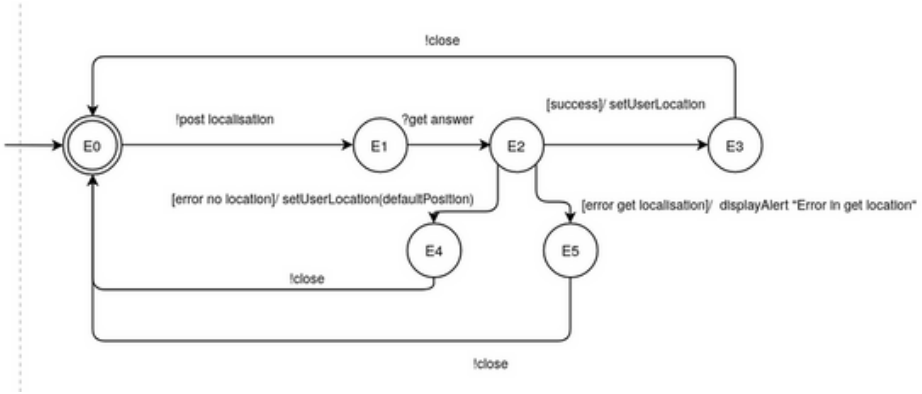

Fig7 - Automate_Client_Loc

Coté Serveur:

Le serveur attend l'accord de l'utilisateur afin de renvoyer les coordonnées permettant au client de mettre à jour la carte selon le choix de l'utilisateur.

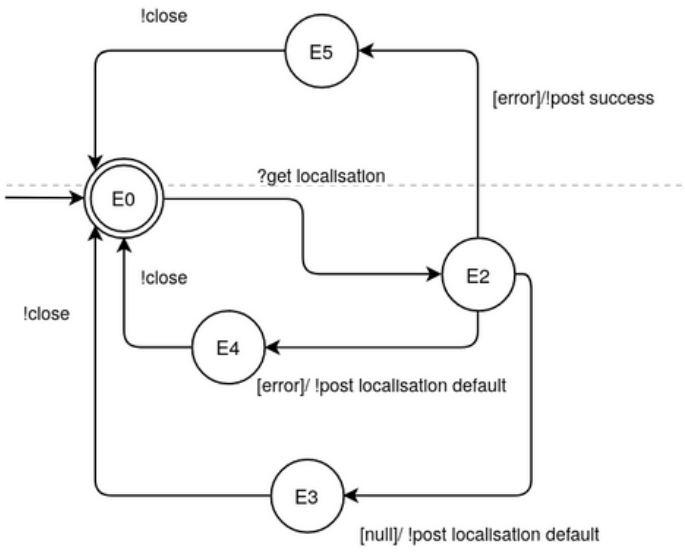

Fig8 - Automate_Server_Loc

2.2.5 Donner un avis sur une toilette

Coté Client:

Lorsqu'il est connecté, le client pourra donner un avis sur une toilette. Il postera donc son id et son avis contenant l'id du toilette ainsi que sa note.
La vue se met à jour dynamiquement.

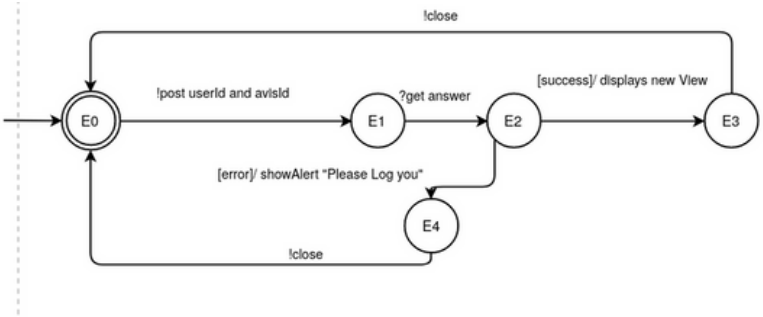

Fig9 - Automate_Client_Review

Côté serveur:

Le serveur récupère donc les informations du client. Deux cas sont possibles:

- Le client n'a jamais donné d'avis, dans ce cas un nouvel avis est ajouté (!post) dans son tableau et la moyenne de la toilette concernée est recalculée.
- L'avis du client existe déjà. Son avis est modifiable, il aura donc un avis mis à jour (!put) dans la base de données.

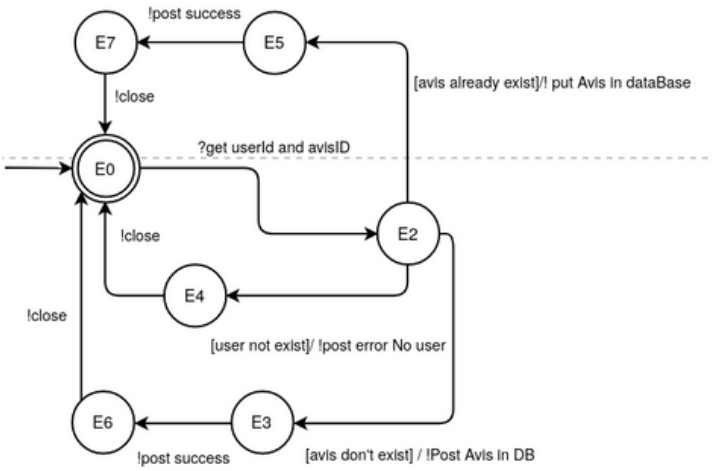

Fig10 - Automate_Server_Review

2.2.6 Construire un itinéraire

Coté Client:

Au clique sur un objet (toilette ou vélo), à l'unique condition que l'utilisateur ait activé sa localisation, le client post ce dernier. Ensuite le client récupère du serveur un itinéraire dessiné grâce à la bibliothèque leaflet.

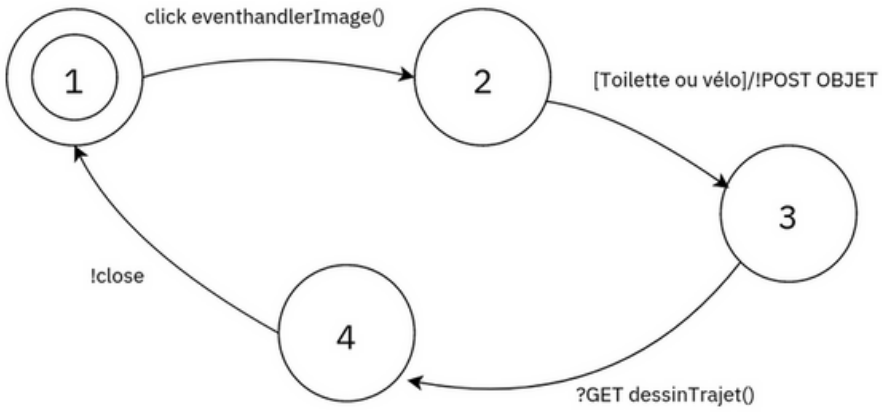

Fig11 - Automate_Client_way

Coté serveur

De son côté, le serveur récupère l'objet et avec les coordonnées, trace un trajet entre la position de l'utilisateur et sa localisation, s'il la activé. Il poste ensuite le Marker qui est un objet complexe qui permet de tracer le trajet sur la carte.

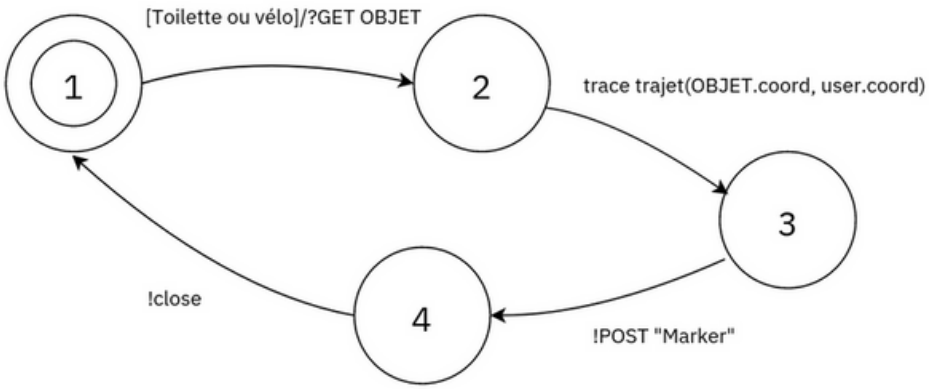

Fig12 - Automate_Server_way

2.2.7 Mise à jour de la carte selon la localisation nominale rentrée dans le champ de texte.

Coté Client:

Dès qu'il post sa recherche, L'API Open street Map (détaillé dans le côté serveur). récupère la position envoyé par le serveur et le client met à jour la carte avec sa position.

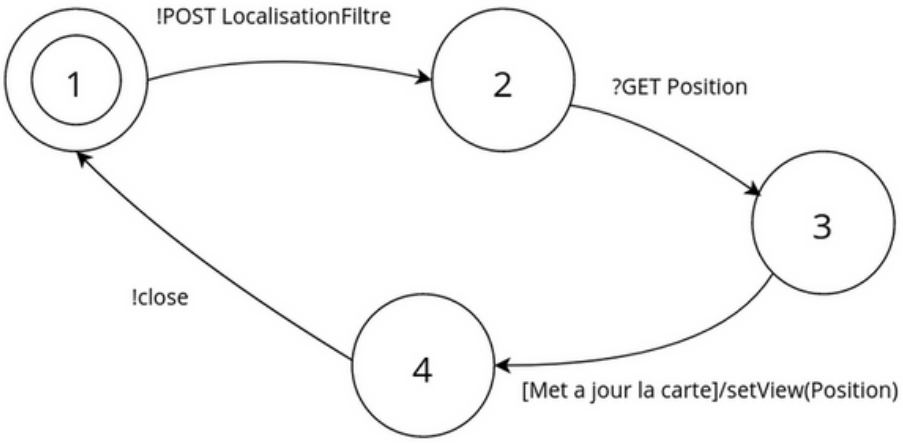

Fig13 - Automate_Client_WordInput

Coté Serveur:

Pour pouvoir rechercher des coordonnées pour un nom donné, nous avons fait appel à l'API Open Street Map avec comme filtre les coordonnées qui délimite Nantes ainsi que le nom récupéré du client.
Les coordonnées récupérés sont transmises au client.

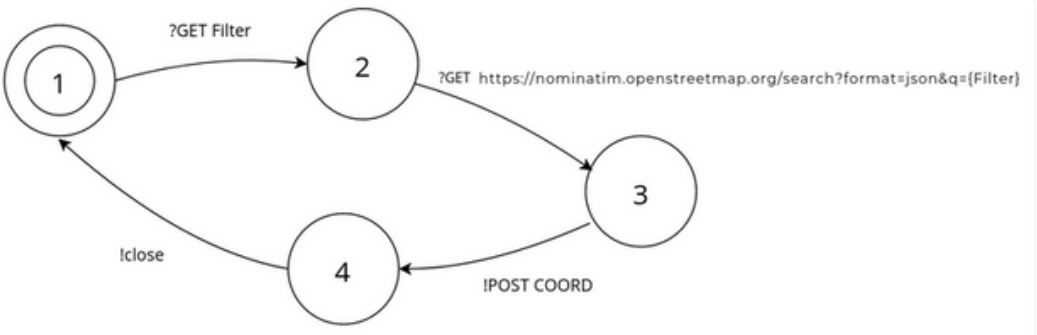

Fig14 - Automate_Server_WordInput

3- Application Android

3.1- Présentation des fonctionnalités

Une grande différence vis à vis de notre site web est que le sujet de l'application a été clairement donné et guidé. C'est en parti pour cela que nous n'avons pas de lien entre ces deux applications.

Aussi, la partie serveur est presque inexistante car le client Android est considéré comme un tout et se gère lui même. Ici on considérera le serveur comme l'API (car on lui fait des Appels).

Deux micro-services ont été développés pour cette application:

- Récupérer des recettes selon les exigences du client
- Afficher les détails de la recette voulue.

3.2- Représentation des micro-services sous forme d'automates

3.2.1 Récupérer les recettes selon les exigences

On récupère premièrement les paramètres que l'utilisateur a rentré (voir 1.2).
Les paramètres sont ensuite passés dans l'URL.
Il y a donc plusieurs issues possibles:

- Soit l'API a été trop sollicitée et demande de payer un abonnement, dans ce cas un message Toast Android sera affiché indiquant cette Erreur.
- Soit la requête n'a pas fonctionné pour une raison inconnue et est aussi considéré comme une erreur.
- Soit la requête a fonctionné mais rien n'est renvoyé car aucune recette correspond aux critères. Un message Toast en informe l'utilisateur.
- Soit une liste est affichée montrant les recettes correspondant aux demandes.

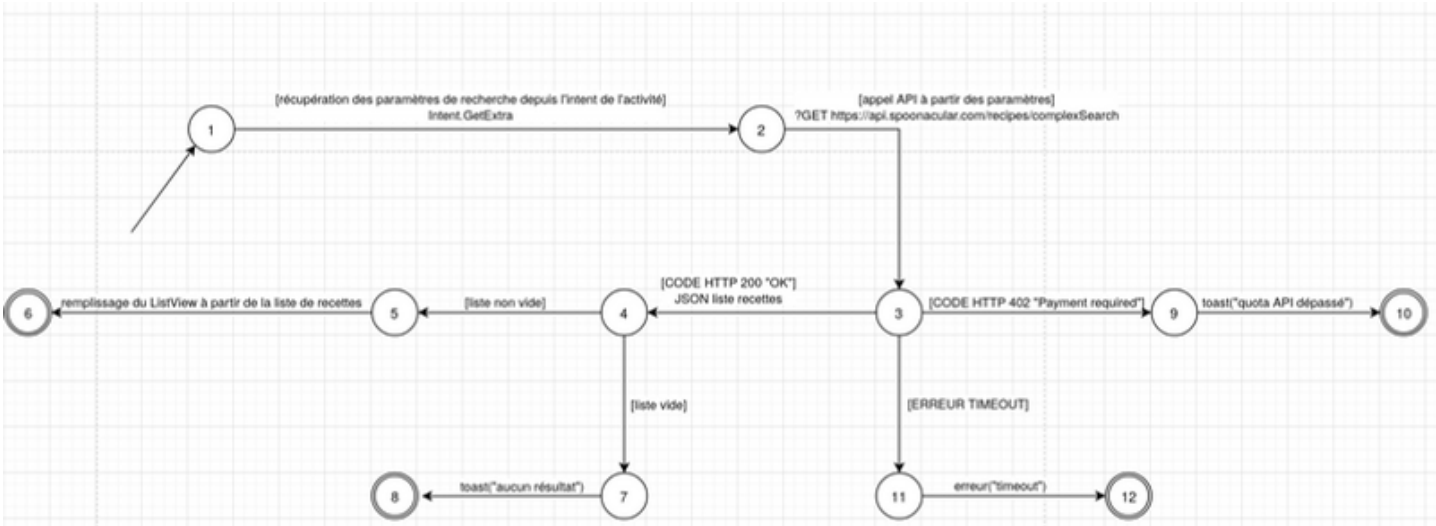

Fig15 - Automate_Client_FindRecipe

3.2.2 Afficher les détails de la recette voulue.

Une fois la liste affichée, l'utilisateur a le choix de cliquer sur une des recettes.
l'id de la recette est transmis à la vue suivante.
Puis, le client effectue une nouvelle requête à l'API Spoonacular qui lui renverra la recette avec de nombreux éléments décrivant la recette sous toutes ses formes.
Tout comme le micro-service précédent, l'API peut être soit trop sollicitée, soit renvoyer une erreur lorsqu'un résultat est inattendu.
Sinon on affiche dans la vue tous les éléments qui attendent d'être initialisé par ceux contenus dans le JSON récupéré.

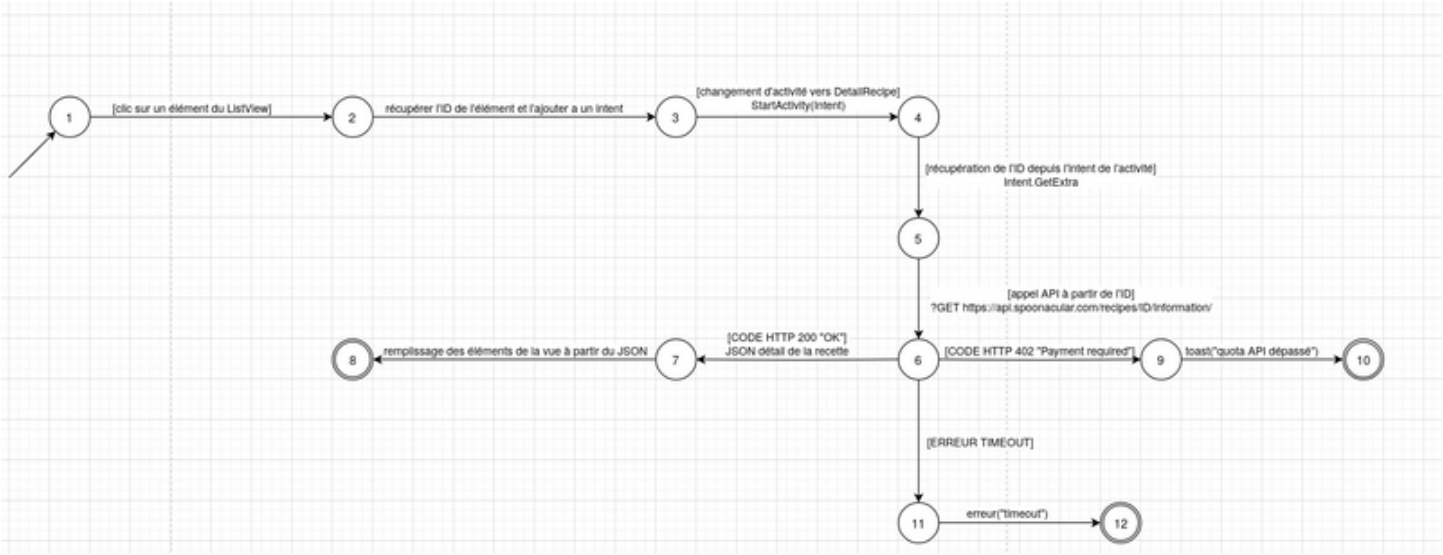

Fig16- Automate_Client_FindRecipe

4- Conclusion

4.1 Bilan

Pour résumé, les automates ont été essentiels afin d'implémenter nos micro-services. Aussi, cette phase d'avant code a été très importante pour nous lancer. En effet, nos idées étaient encore floues et nous ne savions pas par où commencer par peur de s'engager sur une mauvaise piste.
Ainsi, cette schématisation nous a permis de structurer nos idées dès le départ et a simplifié grandement l'implémentation de nos micro-services.
Enfin, cela nous a donné d'autres idées auxquelles nous n'aurions pas pensé. Cela nous a permis d'enrichir notre site web avec des fonctionnalités comme établir un système d'avis pour nos toilettes.

Malgré tout, certains automates ont été implémentés différemment au besoin de notre site. Ces "oublis" ont été remarqués au fur et à mesure que notre site web grandissait.
Nous avons constaté que certains automates manquaient de précision. Nous n'avons pas pensé à toutes les subtilités du code avant de programmer, bien que cela était nécessaire.

4.2 Limitations et Ouvertures

Nous avons fait face à quelques problèmes d'implémentation surtout à cause de React, une bibliothèque JavaScript, en Front-End (donc plus vers le côté client) Elle gérait à elle seule des actions normalement réalisés dans le back-end.
Cela rendait même impossible la délégation de ces actions dans le Back-end.

Dans notre application mobile, notre compte pour l'API étant sans abonnement, nous avons dû réguler notre nombre de requêtes effectuées à l'API car ces dernières étaient limitées.
Pour ce qui est des ouvertures, nous avons fait en sorte que le site soit facilement modifiable et améliorable en rajoutant par exemple, plus d'APIs. Cela rajouterait alors des micro-services, et par extension de nouveaux automates.

5- Remerciements et Références

5.1 Remerciements

Nous remercions M. Attiogbé pour l'aide ainsi que pour l'accès au support du cours et pour la proposition de certaines ressources supplémentaires.

5.2 Références

- Cours et TD de M. Attiogbé disponible sur la plateforme MADOC.
- Rapports d'anciens étudiants sur la même thématique que ce rapport.

- Schéma des automates réalisés dans un premier temps à partir de Visual Paradigm et utilisation de draw.io pour terminer.
- Canvas: Traitement de Texte et style.
- Aucun support externe n'a été utilisé pour réaliser les automates et l'écriture de ce rapport.