O.R.B.I.T.

(Optimal Route Booking Intergalactic Travel)

Compte Rendu d'Activité 2

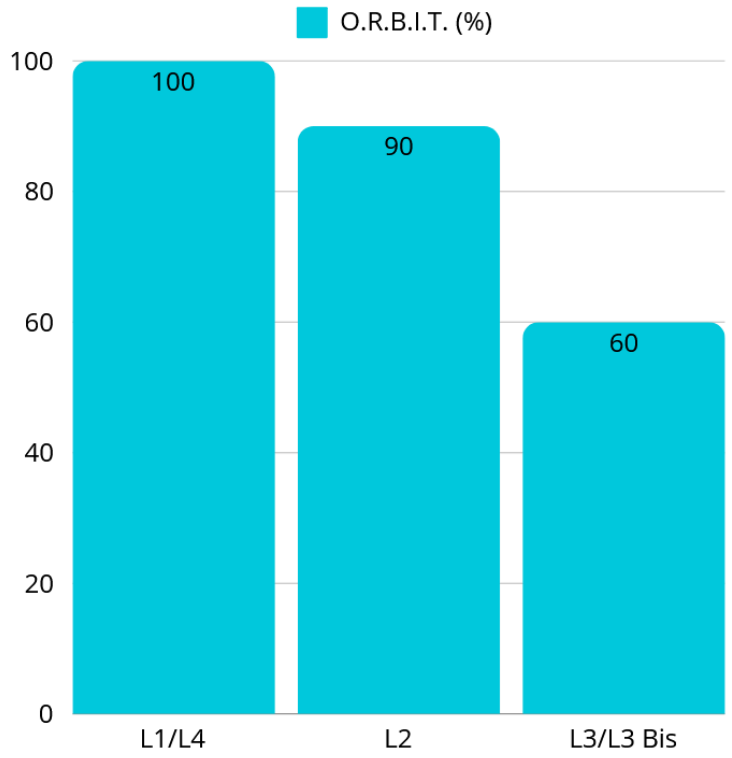
RICHE & TELLE & SYNAEVE

Niveau 1 : Présentation générale

État général

Le projet avance de manière progressive et méthodique. Les objectifs sont atteints étape par étape.

Avancement par modules (en % effectué) :



Résumé de l'état du projet

L'algorithme du plus court chemin en C est développé et terminé. La partie Java (gestion, construction, écriture des graphes de transport) est finalisée, la relation à l'application web n'est pas établie. L'ajout de fonctionnalités utilisateurs telles que la réservation reste manquante côté Front-End; authentication, navigation et carte dynamique intégrée.

Niveau 2 : État des modules

Module L1/L4 : Recherche du plus court chemin (C)

Statut : **Terminé.**

Description des travaux réalisés

Un fichier texte représentant un graphe est lu en argument du programme, ce dernier va appliquer l'algorithme A* pour permettre de renvoyer le plus court chemin à utiliser ainsi que sa distance.

Objectifs atteints

Stabilité confirmée à travers des tests unitaires et des tests d'intégration. L'algorithme répond à nos attentes, le plus court chemin est établi, le calcul de la distance est fiable.

Module L2 : Gestion des Graphes (Java)

Statut : **En finalisation…**

Description des travaux réalisés

La connexion à la base de données est établie. Chaque objet du graphe (Sommet, Arête, Planète) est créé en Java et alimenté par la base de données, l'ensemble est synthétisé dans un fichier texte et envoyé au programme C.

Prochaines étapes

Prise en charge d'arguments supplémentaires permettant une recherche approfondie grâce à de nouveaux paramètres (coût, capacité du vaisseau sélectionné, visible par l'empire ou non). Le programme Java construira le graphe en fichier texte uniquement à partir des planètes valides, l'algorithme en C devra parcourir ainsi un fichier plus léger (aucune modification ici).

Module L3/L3 Bis : Interface Utilisateur (Web)

Statut : **En cours…**

Description des travaux réalisés

- L'interface permet déjà de sélectionner des destinations et certaines contraintes de voyage comme le choix de la navette souhaitée et le nombre de personnes via un système de formulaire implémenté directement à la page d'accueil et via un lien dans la barre de navigation.

- Un chatbot a été mis en place afin de pouvoir guider les utilisateurs à travers leurs expériences.

- Carte dynamique contenant les planètes et une liaison entre les deux planètes choisies dans le formulaire. Possibilité de voir la carte par défaut sans voyage sélectionné.

- Système de photo de profil modifiable par les utilisateurs.

- Page Compte permettant aux utilisateurs de visualiser et modifier leurs informations de façon simple.

- Page Info trafic (statique) pour un suivi des lignes de transport.

- Page Horaire (statique) pour un suivi des horaires par lignes.

Prochaines étapes

- Améliorer les animations et transitions pour rendre l'expérience utilisateur fluide et intuitive.

- Ajout des contraintes restantes du formulaire de voyage dans la gestion du chemin final.

- Pouvoir choisir son départ/destination depuis la carte dynamique et construire son voyage de manière visuelle.

- Implémentation des données sur le chatbot.

- Réelles perturbations/horaires à ajouter.

- Ajout d’un miniserveur Node.js pour récupérer sous fichier TXT les entrées des utilisateurs que le bot n’a pas réussi à comprendre.

Niveau 3 : Analyse détaillée

Module L1/L4 : Recherche du plus court chemin (C)

Défis rencontrés

- Problème sur la gestion du transfert du graphe du Java jusqu'en C
- Problème rencontré sur les structures de données à utiliser pour appliquer l'algorithme du plus court chemin efficacement

Solutions envisagées

- Une des solutions les plus simples est de produire un fichier texte décrivant le graphe selon un pattern bien précis; en première ligne, nous décrivons le nombre d'arêtes du graphe (les voyages), puis son nombre de sommets (les planètes). Ensuite, nous représentons sur chaque ligne du fichier : les sommets, en mettant l'id de la planète, l'index de début de ses arêtes dans l'array des arêtes, ainsi que son nombre d'arêtes. Enfin, pour la description des arêtes, nous mettons d'abord l'id de la planète de départ, la distance, et l'id de la planète d'arrivée.
- Il a fallu réfléchir à un moyen efficace d'organiser les données en C. Nous avons alors créé plusieurs structures de données, une première décrivant un graphe, composé du nombre d'arêtes et du nombre de sommets, d'un tableau de structure décrivant les arêtes, et d'un tableau de structure décrivant les sommets.

Module L2 : Gestion des Graphes (Java)

Défis rencontrés

- Problème de configuration de IntelliJ sur un poste personnel (Celui d'Alexis) avec Maven.
- Communication difficile avec une base de données sans avoir à tout le temps la recréer. Configuration de la liaison entre la base de données du serveur personnel de Julien au code Java.
- Blocage lorsque l'on veut chercher à pouvoir exécuter du code Java sur un serveur Web.

Solutions envisagées

- Le projet Maven a été créé sur une session sur les ordinateurs de l'IUT, mais pour une raison qui nous est encore inconnue, le projet Maven n'était pas détecté par la machine d'Alexis, ce qui a posé problème pour le développement en dehors des cours. Il a alors fallu qu'on avance principalement en cours sur les machines de l'IUT, en faisant de lourds commits sur le code source hébergé sur GitHub.
- Ce problème avec la détection du projet nous a alors amené à réfléchir à un moyen d'avoir une base de données en ligne pour ne pas avoir à la réinstaller à chaque

changement de poste, et surtout, pour éviter quelconque restrictions que l'on pourrait rencontrer avec les serveurs de l'université.
- Après étude et compréhension de la configuration à mettre en place, nous avons pu configurer le serveur d'écoute en ouvrant le port 3306, créer/modifier le user MySQL du projet pour lui permettre de se connecter depuis l'extérieur (autre que @localhost) afin d'exécuter les requêtes de notre code Java.
- Nous nous sommes renseignés pour ce qui est de la partie déploiement du code sur serveur web, cependant on nous a informé que les serveurs de l'IUT possédait avant des JVM intégrées, mais que ce n'est plus le cas. On nous a alors parlé d'une solution; GraalVM, un outil permettant de créer un exécutable pour le code Java. Il ne nous reste plus qu'à l'utiliser le moment venu.

Module L3/L3 Bis : Interface Utilisateur (Web)

Défis rencontrés

- Problème lors de l'affichage des planètes sur la map, les planètes ne s'affichent pas (erreur dans le JS) mais lorsque l'on entre un voyage, elles s'affichent, l'erreur provient du code JavaScript qui affiche la map sans charger les planètes.

- Problème sur le script d'insertion en base de donnée à cause de la taille du fichier non-supporté par les serveurs Wamp même en passant par une insertion via le site après modification des données depuis les fichiers du serveur wamp en local, le problème persistait.

- Lors de l'implémentation du chatbot, il était impossible pour le bot de comprendre ce que cherche l'utilisateur afin d'y répondre lorsque ce dernier faisait des fautes quelles qu'elles soient.

Le reste des fonctionnalités web sont en cours de développement par l'équipe.

Solutions envisagées

- Correction du bug via le code JavaScript dans le but de charger les planètes en amont avant de les afficher sur la carte dans le but d'éviter une erreur de donnée vide.

- Échange de dump de la base de donnée dans le but de pouvoir récupérer les données plus facilement afin de contrer les erreurs causées par les serveurs Wamp.

- Implémentation de l'algorithme de Levenshtein permettant ainsi au bot de comprendre l'utilisateur même avec quelques erreurs et de pouvoir le réorienter vers une question plus claire pour pouvoir y répondre.