

Meet us

Application de rendez-vous

Mémoire de projet de semestre présenté par

Marc Vachon

**Ingénierie des technologies de l'information avec orientation en
Ingénierie des technologies de l'information**

Mars 2019

Professeur-e HES responsable

Orestis Malaspinas

SOMMAIRE DÉTAILLÉ

Sommaire détaillé.....	2
Remerciements	4
Résumé	5
Table des illustrations.....	6
Annexes	7
1. Code de l'application	7
Liste des acronymes	8
Introduction	9
Chapitre 1 : Préparation.....	10
1. Fonctionnalités	10
1.1. Prioritaires	10
1.2. Secondaires.....	10
2. Technique	11
2.1. API	11
2.2. Langage.....	11
Chapitre 2 : Conception.....	12
1. Architecture	12
2. Classes	12
2.1. Utilisation des apis	12
2.2. Input/Output	12
2.3. Vérification.....	12
2.4. Meeting (Main)	13
Chapitre 3 : Fonctionnement	14
1. Utilisation	14
1.1. Couleur de l'interface	14
1.2. Connexion à internet	14
2. Commandes	14
2.1. Lancement	14
2.2. Exit	15
3. Exécution.....	15
3.1. Données utilisateurs	15

4. Résultat.....	16
4.1. Choix de la ville de rendez-vous	16
4.2. Trajets.....	17
Chapitre 5 : Futur de l'application	18
1. Front-end	18
1.1. Interface graphique.....	18
1.2. Développement.....	18
1.3. Que choisir ?.....	18
2. Fonctionnalités utilisateurs.....	19
3. Fonctionnement à améliorer	19
3.1. Choix apis.....	19
Conclusion.....	20
Références documentaires	21

REMERCIEMENTS

Je tiens à remercier mon encadrant de projet M. Malaspinas qui m'a suivi tout au long de ce projet. Sa disponibilité a été très appréciable ainsi que le suivi de mon projet. La communication était facile.

J'aimerais aussi remercier M. Perrot qui m'a permis de faire ce rapport de manière professionnelle et qui nous a fournis tous les outils pour y arriver.

Enfin j'aimerais remercier mes camarades de classe avec qui nous avons partagé certains problèmes et nous sommes soutenues.

RÉSUMÉ

Cette application permet la rencontre de plusieurs personnes habitant en suisse. Son principe est de trouver une ville dans laquelle se retrouver et de leur créer un rendez-vous ainsi que le trajet (en transport public) à réaliser.

Chaque personne entre son emplacement (ville) et ses dates possibles (date et heures). L'application va ensuite trouver le centre géographique par rapport à chacune de leur position et chercher la ville la plus proche. On va ensuite chercher le rendez-vous qui arrange le plus de personnes (date et heures).

Une fois la date et le lieu trouver, l'application va chercher le trajet pour chacun. Elle affichera pour chacun des membres le trajet le plus rapide entre le point entrer de chacun et le lieu de rendez-vous.



Candidat-e :

MARC VACHON

Filière d'études : ITI

Professeur-e(s) responsable(s) :

ORESTIS MALASPINAS

Travail soumis à une convention de stage en
entreprise : non

Travail soumis à un contrat de confidentialité : non

TABLE DES ILLUSTRATIONS

Figure 1: Nominatim.....	11
Figure 2: GeoDB Cities API.....	11
Figure 3: Transport API.....	11
Figure 4: Test connexion internet	14
Figure 5: make run	15
Figure 6: Exit programme	15
Figure 7: Renseigner ville	15
Figure 8: Choix de ville	16
Figure 9: Ajout date au calendrier.....	16
Figure 10: Ajouter utilisateur	16
Figure 11: Choix ville de rendez-vous.....	17
Figure 12: Résultat	17

ANNEXES

1. CODE DE L'APPLICATION

Le code étant trop long pour être intégré je mets ici le lien vers le git contenant celui-ci :

<https://gitedu.hesge.ch/marc.vachon/meet-us.git>

LISTE DES ACRONYMES

API : Application Programming Interface

JSON : Javascript Object Notation

Front-end : Interface web avec laquelle l'utilisateur peut directement interagir.

INTRODUCTION

Dans le cadre du projet de semestre j'ai choisi le projet proposé par M. Malaspinas « Application de rendez-vous ».

Les objectifs définis pour ce projet était la création d'une application créant un rendez-vous pour deux ou plusieurs personnes habitant en Suisse. Chaque personne entre sa ville de départ ainsi qu'un calendrier de disponibilité. L'application va ensuite trouver un centre géographique entre chacun d'eux et choisir une date de rendez-vous.

Le résultat est le rendu d'une date et heure de rendez-vous avec le trajet en transport public pour chaque membre entre leur ville et celle de rendez-vous.

CHAPITRE 1 : PRÉPARATION

Avant de commencer la création de l'application il a fallu faire des choix de fonctionnalités à implémenter et technique.

1. FONCTIONNALITÉS

1.1. Prioritaires

Dans les fonctionnalités prioritaires on retrouve l'entrée des données utilisateurs. Chaque utilisateur doit pouvoir entrer son adresse ainsi que ses dates de disponibilité. On doit aussi lui laisser le choix d'en avoir plusieurs.

Calculer le centre géographique par rapport aux adresses des utilisateurs. Il faut aussi trouver une date de rendez-vous suivant le calendrier des utilisateurs.

La dernière est le fait d'avoir le trajet en transport public pour chacun des utilisateurs.

1.2. Secondaires

Les fonctionnalités secondaires sont celles qui sont implémenté un fois que les prioritaires sont fonctionnelles. Elle ne sont pas vital au programme mais permette une utilisation plus aggréable et plus de contrôle pour les utilisateurs.

Dans les fonctionnalités secondaires on retrouve le fait de pouvoir choisir la ville par rapport à ce qu'on a entré ainsi que la possibilité de pouvoir entrer.

Le choix de la ville de rendez-vous par rapport au centre.

Enfin, le fait de pouvoir quitter le programme à tout moment est aussi disponible.

2. TECHNIQUE

2.1. API

Cette application utilise 3 apis différentes. Chacune a un rôle bien spécifique.



Figure 1: Nominatim



Figure 2: GeoDB Cities API



Figure 3: Transport API

Nominatim (OpenStreetMap) : Elle est utilisée pour la recherche des villes avec le nom entré par un utilisateur. Si on entre Genève elle va chercher les références à Genève et nous afficher les villes qui ont ce nom.

GeoDB Cities API : Elle est utilisée pour trouver la ville la plus proche d'un point géographique. On lui renseigne les coordonnées géographiques de notre point, le rayon de recherche, la taille minimale de la ville et elle nous donne toutes les villes correspondantes.

Transport API : Elle est utilisée pour trouver le trajet entre une ville A et une ville B. Elle nous donne toutes les informations du trajet. Quel moyen de transport, le lieu où le prendre, le temps de trajet total, etc.

2.2. Langage

L'application a été entièrement codée en Java. L'ajout d'une librairie a été nécessaire afin de travailler avec Json pour la réception des données des apis.

CHAPITRE 2 : CONCEPTION

1. ARCHITECTURE

L'architecture a été faite pour être adaptée plus tard. Toutes les parties du code ont été séparées de la classe principale où se déroule le programme. Le fait de séparer le code en différentes classes permet d'ajouter des éléments à l'application plus facilement.

2. CLASSES

2.1. Utilisation des apis

L'utilisation des APIs se fait dans la classe "Request". Dans cette classe on y retrouve toutes les requêtes faites aux différentes APIs. Le contenu des requêtes est rendu en String.

Afin de ne pas dupliquer du code inutilement une fonction interne à la classe s'occupe de faire les requêtes. Son seul argument est l'URL sur lequel on va faire la requête.

2.2. Input/Output

L'écriture et l'affichage ont été séparés du code général. Les deux ont une classe propre à leurs fonctions.

Pour l'input, ça permet de sauvegarder le contenu entré par l'utilisateur dans un buffer et l'utiliser plus tard. Ça permet aussi d'ajouter des commandes comme "exit" par exemple.

Pour l'output, ça permet d'y ajouter des fonctions d'affichage propre au besoin du programme. On y retrouve aussi la possibilité de mettre le texte en couleur ou d'afficher des sections, plusieurs fonctions qui facilitent l'affichage.

2.3. Vérification

Une classe pour faire les tests "Check" contient les fonctions de tests. On y retrouve des tests de validité des dates par exemple. Il y a aussi la fonction qui vérifie la connexion Internet, la validité des heures ou des coordonnées.

Avoir cette classe permet de réutiliser ces fonctions partout dans le programme sans avoir à refaire les mêmes vérifications.

2.4. Meeting (Main)

C'est dans la classe Meeting qu'on retrouve le déroulement du programme. Cette classe fait le lien avec toutes les autres, on y retrouve la table des utilisateurs, l'input/output, les requêtes, etc.

Toutes les données entrées par un utilisateur sont enregistrées dans des tableaux correspondants. On retrouve la table des utilisateurs et la ville et la date de rendez-vous.

CHAPITRE 3 : FONCTIONNEMENT

1. UTILISATION

1.1. Couleur de l'interface

L'interface utilise un code couleur afin d'être le plus lisible possible.

- Rouge : les erreurs et lorsqu'on quitte le programme.
- Vert : les données enregistrées et les étapes réussies.
- Bleu-ciel : lorsque l'utilisateur écrit quelque chose.

1.2. Connexion à internet

La première vérification que le programme effectue est la connexion à internet. Comme l'application utilise des apis qui sont le coeur de celle-ci la connexion est obligatoire.

Si l'application n'arrive pas à se connecter à internet elle affiche un message et quitte le programme.

```
MarcBook-Pro:src marcbook$ make run
javac -cp "../java-json.jar" *.java
java -cp "../java-json.jar" Meeting
Pour utiliser cetter application veuillez vous connecter à internet
Exit...
MarcBook-Pro:src marcbook$ █
```

Figure 4: Test connexion internet

2. COMMANDES

2.1. Lancement

Afin de simplifier l'exécution de l'application un makefile a été réalisé. Par défaut la commande "make" va simplement compiler le programme. Pour lancer le programme on va utiliser "make run", cela va compiler et lancer l'application.

```
MarcBook-Pro:src marcbook$ make run
javac -cp "../java-json.jar" *.java
java -cp "../java-json.jar" Meeting
-----
Bonjour, veuillez entrer les informations utilisateur:
Votre ville:
█
```

Figure 5: make run

2.2. Exit

Lors de l'utilisation de l'application il est possible de quitter celle-ci à n'importe qu'elle moment avec la commande "exit".

```
-----
Date possible (jour/mois/année heure:min heure:min):
exit
Exit...
MarcBook-Pro:src marcbook$ █
```

Figure 6: Exit programme

3. EXÉCUTION

L'application fonctionne en deux phase. La première est le renseignement des données des utilisateurs et la seconde est le rendu des résultats.

Afin d'avoir des résultats cohérent un exemple d'utilisation a été réalisé avec un utilisateur se trouvant à Genève, un autre à Berne et un dernier à Zürich.

3.1. Données utilisateurs

La première étape est d'entrer la ville de laquelle on souhaite partir. Dans notre cas on va renseigner Genève, Berne et Zurich. L'exemple va se faire avec Genève.

```
-----
Bonjour, veuillez entrer les informations utilisateur:
Votre ville:
█
```

Figure 7: Renseigner ville

Une fois la ville renseigner l'application va nous proposer des choix de ville. Nous allons choisir la ville correspondante grâce à l'id ou appuyer sur n'importe quelle touche afin d'entrer une nouvelle fois le nom de la ville si celle-ci n'apparait pas ou qu'on s'est trompé.

```

Votre ville:
Genève
-----
Voici les résultats de votre recherche (10 résultats) :

0) Genève, Schweiz/Suisse/Svizzera/Svizra
1) Genève, Schweiz/Suisse/Svizzera/Svizra
2) Genève, Place de Cornavin, Grottes et Saint-Gervais, Genève, 1
3) Genève, Fourilles, Moulins, Allier, Auvergne-Rhône-Alpes, Fran
4) Genève, Blaymont, Agen, Lot-et-Garonne, Nouvelle-Aquitaine, Fr
5) Genève, Touloud, Tournon-sur-Rhône, Ardèche, Auvergne-Rhône-Al
6) Genève, Le Truel, Millau, Aveyron, Occitanie, France métropoli
7) Geneve, West Bank Demerara, Essequibo Islands-West Demerara, 0
8) Geneve, Saint-Cirgues-de-Prades, Largentière, Ardèche, Auvergn
9) la genève, 9e Fonds Rouge Torberck, Commune de Jérémie, Arrond
Aucun > Appuyez sur n'importe quelle touche!

Veuillez choisir le numéro correspondant:

```

Figure 8: Choix de ville

Dans notre cas l'id correspondant est le 0. Lorsqu'on fait notre choix, les coordonnées de la ville sont sauvegardées.

Maintenant nous allons renseigner notre calendrier. Chaque utilisateur peut entrer le nombre de date qu'il souhaite. Le format est indiqué et doit être strictement respecté.

```

-----
Date possible (jour/mois/année heure:min heure:min):
12/07/2019 12:00 22:00
SUCCESS
Vous avez saisi : 12/07/2019 de 12:00 à 22:00
-----
Avez-vous entrer toutes les t_date ? (y/n)

```

Figure 9: Ajout date au calendrier

Une fois qu'un utilisateur a terminé on nous propose d'en ajouter un autre si on le souhaite. A savoir qu'un minimum de 2 utilisateurs est demandé le choix est donc seulement disponible après le deuxième utilisateur terminé.

```

-----
Avez-vous fini d'entrer les utilisateurs ? (y/n)

```

Figure 10: Ajouter utilisateur

4. RÉSULTAT

4.1. Choix de la ville de rendez-vous

Après avoir calculé le centre géographique entre les utilisateurs plusieurs choix de ville proche de ce centre nous sont proposés. Comme pour les villes l'application nous permet de choisir

parmis ces propositions à l'aide des ids. A savoir que seules les villes en suisses nous sont proposées.

```

-----
Choix de la ville...
-----
Voici les villes proposées (10 résultats) :

0) Arrondissement administratif de Berne-Mittelland, Suisse
1) District de Lausanne, Suisse
2) district d'Arlesheim, Suisse
3) Bern, Suisse
4) Lausanne, Suisse
5) Arrondissement administratif de Thoune, Suisse
6) district de la Sarine, Suisse
7) Arrondissement électoral de Lucerne-campagne, Suisse
8) Arrondissement administratif de Bienne, Suisse
9) Arrondissement administratif de l'Emmental, Suisse
Aucun > Appuyez sur n'importe quelle touche!

Veuillez choisir le numéro correspondant:

```

Figure 11: Choix ville de rendez-vous

Nous allons prendre le premier choix qui est Berne. Le numéro 3.

4.2. Trajets

Une fois la ville de rendez-vous choisie l'application va nous rendre les résultats finaux. En premier temps la date et l'heure de rendez-vous et ensuite le trajet que chacun va emprunter.

On voit que le rendez-vous a été fixé le 12/07/2019 de 12h à 22h.

En dessous on voit le trajet pour chacun des utilisateurs ainsi que les informations de chaque étapes.

On peut aussi voir que si l'utilisateur habite sur le lieu de rendez-vous on ne lui donne pas de trajet.

```

-----
Rendus des résultats...
Le rendez-vous aura lieu le: 12/07/2019 de 12:0 à 22:0
-----
Parcours de Genève à Bern
Durée du trajet: 01:52

```

Heure	Voyage	Plateforme
10:15	Genève IC 5	4
11:24	Neuchâtel	5
11:32	Neuchâtel RE RE 3921	2
12:07	Bern	13

```

-----
Quelle chance vous habitez sur place !
-----
Parcours de Zürich HB à Bern
Durée du trajet: 00:56

```

Heure	Voyage	Plateforme
11:02	Zürich HB IC 8	31
11:58	Bern	5

```

-----
SUCCESS

```

Figure 12: Résultat

CHAPITRE 5 : FUTUR DE L'APPLICATION

Pour finir le tour de l'application on va maintenant voir les possibilités d'évolution de ce programme.

1. FRONT-END

1.1. Interface graphique

Une des premières améliorations qui pourrait être apporté est l'ajout d'une interface graphique bien plus pratique pour un utilisateur lambda. Il serait intéressant d'avoir accès à une carte et aux informations des utilisateurs entrés. Une interface graphique permettra aussi une prise en main plus rapide de l'application.

1.2. Développement

Etant donné que l'architecture de l'application différencie le code de l'affichage et de l'écriture, le développement d'une interface peut être indépendant du code.

Cette avantage permet d'avoir le choix et la possibilité de créer soit une web app soit une application native sur ordinateur ou mobile (dans certain cas la barrière du langage de programmation nécessiterai un portage du code).

1.3. Que choisir ?

Devant le choix de l'interface il faut se poser les bonnes questions. Le public qu'on cible, les coûts de développement ainsi que de maintien, l'accessibilité, etc.

Le choix judicieux d'une web app permettrait de toucher tout le monde d'un coup. Il n'y aurait pas besoin de savoir sur quelle hardware l'utilisateur se trouve et donnerai accès à tout le monde rapidement. Que ce soit à la création ou en maintenance, il est moins coûteux d'avoir une seule interface accessible sur n'importe quel appareil qu'en développer plusieurs.

Il est aussi plus facile d'accès d'aller sur internet que d'installer une application sur tout les supports où l'on désire l'utiliser.

2. FONCTIONNALITÉS UTILISATEURS

Parallèlement à l'ajout d'une interface graphique il serait intéressant d'ajouter certaines fonctionnalités utiles à l'utilisateur. Voici une liste non-exhaustive de quelques idées de fonctionnalités:

- Une interface graphique (comme expliquer ci-dessus au point 1).
- Pouvoir utiliser une adresse plus précise au lieu d'une ville comme point de départ.
- Localiser un utilisateur pour ne pas lui faire entrer lui-même son adresse.
- La modification des données entrées pour un utilisateur (adresse, calendrier).
- Paramétrer le choix de la ville de rendez-vous (taille, bâtiment nécessaire).
- Choisir le moyen de transport (bus ou train).
- Ajouter des poids à chaque point lors du calcul du centre géographique. (Exemple: si 5 personnes habitent à Berne contre une à Genève, on préférera déplacer cette dernière à Berne plutôt que déplacer tout le monde à Lausanne, qui est plus central).
- Avoir des comptes utilisateur afin de sauvegarder ses préférences (carnet d'adresses, calendrier, etc.).
- Calculer le retour de chacun à la fin du rendez-vous.

3. FONCTIONNEMENT À AMÉLIORER

3.1. Choix apis

Comme il y a plusieurs apis qui fonctionnent entre-elles il y a certains cas problématiques de fonctionnement. Le nomage des villes est différent pour chacune d'elles et il arrive que l'api de transport ne trouve pas la ville donnée par l'api de géolocalisation.

Pour le futur il serait important de trouver une solution à ce problème. Une solution serait d'utiliser moins d'apis différentes ou de vérifier chaque nom de ville utilisé et de l'adapter si nécessaire.

CONCLUSION

Pour conclure je suis content de l'application finale. Les fonctionnalités les plus importantes ont été intégrées avec succès. L'application intègre certaines fonctions qui lui permettent d'être utilisée par un utilisateur lambda.

Un autre point positif est la possibilité d'évolution. Les points évoqués plus haut à propos d'une interface graphique et autres fonctionnalités laissent penser que la base présentée ici est prometteuse.

J'ai pris beaucoup de plaisir à créer cette application car elle a un potentiel professionnel et pourrait un jour aboutir et être mise à disposition d'entreprise. Le fait de créer des applications avec cet objectif est très motivant.

RÉFÉRENCES DOCUMENTAIRES

<https://docs.oracle.com/javase/7/docs/api/overview-summary.html>

<http://geodb-cities-api.wirefreethought.com>

<https://nominatim.openstreetmap.org>

<https://transport.opendata.ch>

<https://www.programcreek.com/2014/01/compile-and-run-java-in-command-line-with-external-jars/>

<https://jar-download.com/artifacts/org.json>