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 3](#_Toc4405764)

[Remerciements 5](#_Toc4405765)

[Résumé 7](#_Toc4405766)

[Table des illustrations 9](#_Toc4405767)

[Annexes 11](#_Toc4405768)

[1. Code de l’application 11](#_Toc4405769)

[Liste des acronymes 13](#_Toc4405770)

[Introduction 15](#_Toc4405771)

[Chapitre 1 : Préparation 17](#_Toc4405772)

[1. Fonctionnalités 17](#_Toc4405773)

[1.1. Prioritaires 17](#_Toc4405774)

[1.2. Secondaires 17](#_Toc4405775)

[2. Technique 18](#_Toc4405776)

[2.1. API 18](#_Toc4405777)

[2.2. Langage 18](#_Toc4405778)

[Chapitre 2 : Conception 19](#_Toc4405779)

[1. Architecture 19](#_Toc4405780)

[2. Classes 19](#_Toc4405781)

[2.1. Utilisation des apis 19](#_Toc4405782)

[2.2. Input/Output 19](#_Toc4405783)

[2.3. Vérification 19](#_Toc4405784)

[2.4. Meeting (Main) 20](#_Toc4405785)

[Chapitre 3 : Fonctionnement 21](#_Toc4405786)

[1. Utilisation 21](#_Toc4405787)

[1.1. Couleur de l’interface 21](#_Toc4405788)

[1.2. Connexion à internet 21](#_Toc4405789)

[2. Commandes 21](#_Toc4405790)

[2.1. Lancement 21](#_Toc4405791)

[2.2. Exit 22](#_Toc4405792)

[3. Exécution 22](#_Toc4405793)

[3.1. Données utilisateurs 22](#_Toc4405794)

[4. Résultat 23](#_Toc4405795)

[4.1. Choix de la ville de rendez-vous 23](#_Toc4405796)

[4.2. Trajets 24](#_Toc4405797)

[Chapitre 5 : Futur de l’application 25](#_Toc4405798)

[1. Front-end 25](#_Toc4405799)

[1.1. Interface graphique 25](#_Toc4405800)

[1.2. Développement 25](#_Toc4405801)

[1.3. Que choisir ? 25](#_Toc4405802)

[2. Fonctionnalités utilisateurs 26](#_Toc4405803)

[3. Fonctionnement à améliorer 26](#_Toc4405804)

[3.1. Choix apis 26](#_Toc4405805)

[Conclusion 27](#_Toc4405806)

[Références documentaires 29](#_Toc4405807)

# Remerciements

Je tiens a remercié mon encadrant de projet M. Malaspinas qui ma suivit tout au long de ce projet. Sa disponibilité a été très appréciable ainsi que le suivit de mon projet. La communication était facile.

J’aimerai 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’aimerai 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](#_Toc4403705)

[Figure 2: GeoDB Cities API 11](#_Toc4403706)

[Figure 3: Transport API 11](#_Toc4403707)

[Figure 4: Test connexion internet 14](#_Toc4403708)

[Figure 5: make run 15](#_Toc4403709)

[Figure 6: Exit programme 15](#_Toc4403710)

[Figure 7: Renseigner ville 15](#_Toc4403711)

[Figure 8: Choix de ville 16](#_Toc4403712)

[Figure 9: Ajout date au calendrier 16](#_Toc4403713)

[Figure 10: Ajouter utilisateur 16](#_Toc4403714)

[Figure 11: Choix ville de rendez-vous 17](#_Toc4403715)

[Figure 12: Résultat 17](file:////Users/marcbook/Documents/School/Hepia_3eme/meet-us/rapport.docx#_Toc4403716)

# Annexes

## 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 intéragir. |
|  |  |

# 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 trasnport 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.

## Fonctionnalités

### 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 laiser 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.

### 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.

## Technique

### 2.1. API

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

|  |  |
| --- | --- |
| A close up of a logo  Description automatically generated  Figure : Nominatim | **Nominatim (OpenStreetMap) :** Elle est utilisée pour la recherche des villes avec le nom entrer par un utilisateur. Si on entre Genève elle va chercher les références à Genève et nous afficher les villes qui ont ce nom. |
| Figure : GeoDB Cities API | **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 donnes toutes les villes correspondantes. |
| Figure : Transport API | **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 libraire a été nécessaire afin de travailler avec Json pour la réception des données des apis.

# Chapitre 2 : Conception

## Architecture

L’architecture a été faites pour être adapter plus tard. Toutes les parties du code ont été séparées de la classe principale où se déroule le programme. Le fait de séparé le code en différente classe permet d’ajouter des éléments à l’application plus facilement.

## Classes

### Utilisation des apis

L’utilisation des apis ce 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 pa duppliquer 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.

### Input/Output

L’écriture et l’affichage on été séparé du code général. Les deux ont une classes propre à leur fonctions.

Pour l’input, ça permet de sauvegarder le contenu entrer par utilisateur dans un buffer et l’utiliser plus tard. Ca 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 facilite l’affichage.

### Vérification

Une classe pour faire les testes “Check” contient les fonctions de testes. On y retrouve des testes de validitié 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.

### Meeting (Main)

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

Toutes les données entrer par un utilisateur sont enregistré dans des tableaux correspondant. On retrouve la table des utilisateurs et la ville et la date de rendez-vous.

# Chapitre 3 : Fonctionnement

## Utilisation

### 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.

### 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.

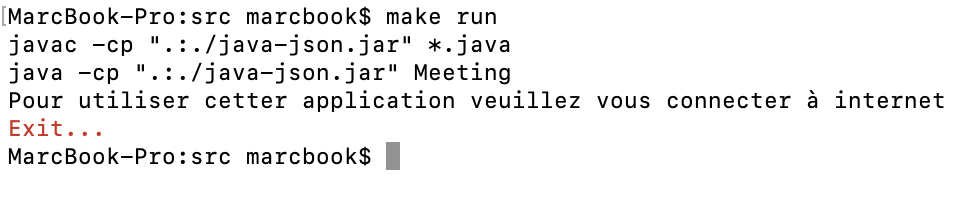


Figure : Test connexion internet

## Commandes

### 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.

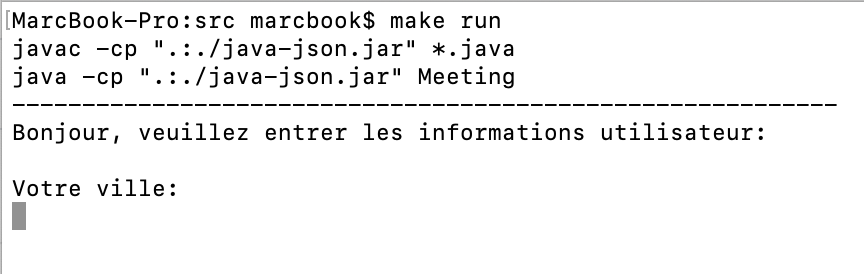


Figure : make run

### 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”.

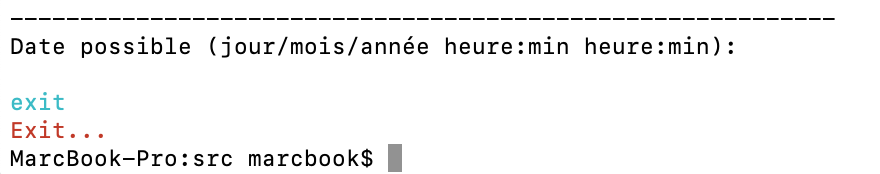


Figure : Exit programme

## 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 cohérant un exemple d’utilisation a été réalisé avec un utilisateur se trouvant à Genève, un autre à Berne et un dernier à Zürich.

### 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.

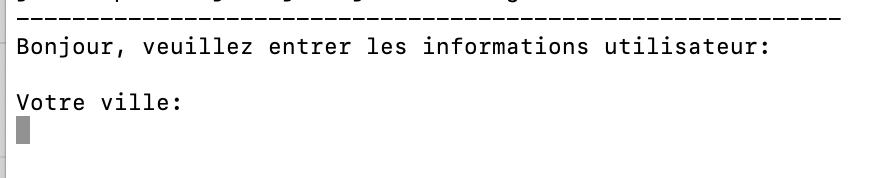


Figure : Renseigner ville

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.

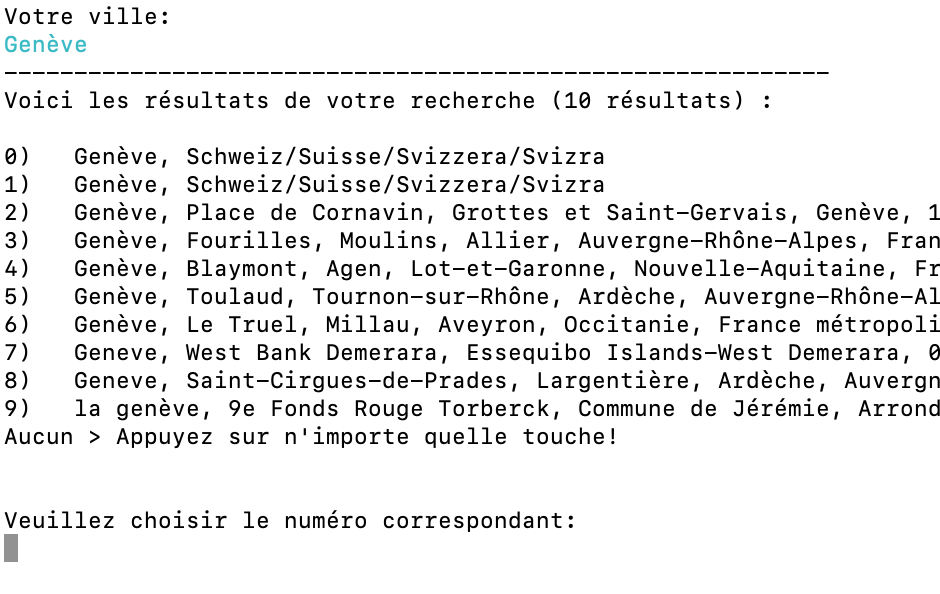


Figure : Choix de ville

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 est doit être strictement respecté.

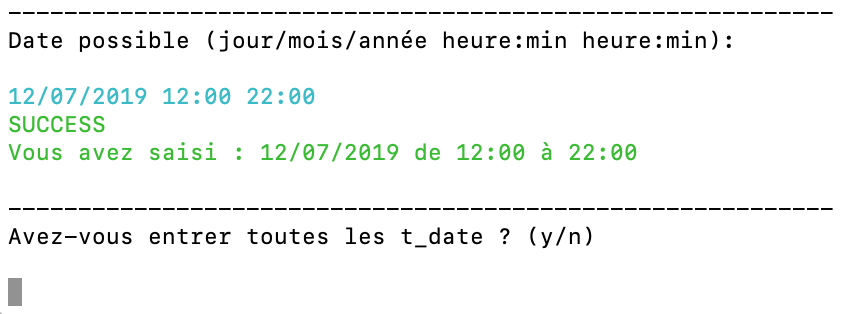


Figure : Ajout date au calendrier

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.

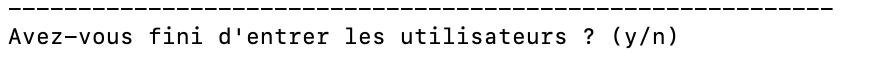


Figure : Ajouter utilisateur

## Résultat

### 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.

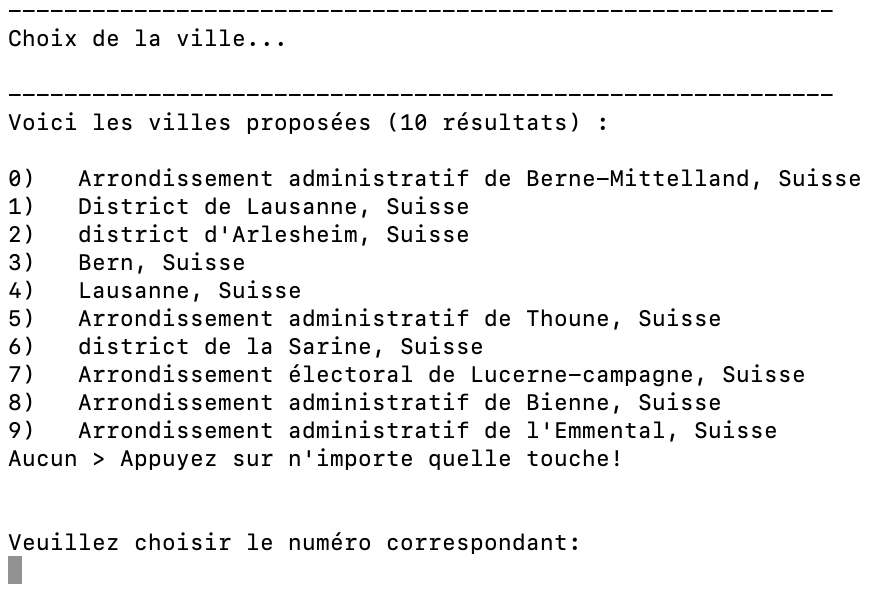


Figure : Choix ville de rendez-vous

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

### Trajets

Une fois fois la ville de rendez-vous choisi 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.



Figure : Résultat

On voit que le rendez-vous à 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.

# Chapitre 5 : Futur de l’application

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

## Front-end

### 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 permettrai aussi une prise en main plus rapide de l’application.

### 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).

### 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’accèssibilité, etc.

Le choix judicieux d’une web app permetrait 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 accéssible 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.

## 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 utilisé une adresse plus précise au lieu d’une ville comme point de départ.
* Localisé un utilisateur pour ne pas lui faire entré lui-même son adresse.
* La modification des données entrés pour un utilisateur (adresse, calendrier).
* Paramétrer le choix de la ville de rendez-vous (taille, batiment nécessaire).
* Choisir le moyen de transport (bus ou train).
* Ajouter des poids à chaque points lors du calcul du centre géographique.  
  (Exemple: si 5 personne habitent à berne contre une à Genève, on préférera déplacer cette dernière à Berne plutot 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.

## Fonctionnement à améliorer

### Choix apis

Comme il y a plusieurs apis qui fonctionne entre-elles il y a certains cas problématique de fonctionnement. Le nomages des villes est différent pour chacune d’elles et il arrive que l’api de transport ne trouve pas la ville données 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 intégré avec succès. L’application intègre certaines fonctions qui lui permette d’être utilisé 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 laisse 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 application avec cette objectif et 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