TRAVAIL DE FIN D'ETUDES REALISE EN VUE DE L'OBTENTION DU GRADE ACADEMIQUE DE BACHELIER EN

INFORMATIQUE DE GESTION

Utilisation / exploitation de données ouvertes dans le domaine de l'accessibilité en Wallonie

Etudiant(e) :

Alexandre ROSATI

Année académique 2014-2015

TRAVAIL DE FIN D'ETUDES REALISE EN VUE DE L'OBTENTION DU GRADE ACADEMIQUE DE BACHELIER EN

INFORMATIQUE DE GESTION

Utilisation / exploitation de données ouvertes dans le domaine de l'accessibilité en Wallonie

Etudiant(e) :

Alexandre ROSATI

Année académique 2014-2015

Tout d'abord, j'adresse mes remerciements au directeur du CETIC, Monsieur Hubaux, qui m'a permis d'effectuer un stage au cœur de son établissement.

Je tiens à remercier vivement mes deux maitres de stage, Monsieur Christophe Ponsard, responsable du département SSE et Fabrice Estiévenart, ingénieur de recherche senior au département SST, pour leurs encadrements et leurs conseils très avisés.

De même, je remercie particulièrement Sébastien Dupont pour son aide sur les sujets peu maitrisés, mais aussi pour tous ses bons conseils qui m'ont permis de finir à bien ce stage.

Je remercie également toute l'équipe du département SST pour leur accueil ainsi que toutes les discussions échangées et les ASBL Gamah, ANLH pour leur coopération autour du sujet.

Enfin, je tiens à remercier toutes les personnes qui m'ont épaulé lors de la rédaction de ce rapport et Madame De Luca pour tous ses encouragements.

Table des Matières

Chapitre 1

Introduction

Durant ma dernière année de bachelier en informatique de gestion à la Haute Ecole Provinciale du Hainaut Condorcet, j'ai pu participer à un stage d'immersion en entreprise pendant une durée de quinze semaines.

Challenge, autonomie, liberté, voici les critères que je m'étais fixés bien avant la première recherche de stage. Lors d'une discussion avec l'un de mes professeurs en rapport à mes attentes, celle-ci m'a orienté vers un centre de recherche : le CETIC . Après plusieurs contacts avec l'un des responsables du CETIC, j'ai été conquis par le type de sujet proposé, au contenu très varié, ainsi que par le contexte très innovant.

Partagé entre les départements SST et SSE, l'objectif du stage était d'explorer des scénarios ouverts pouvant être combinés aux réseaux actuels d'experts afin d'augmenter la quantité d'informations d'accessibilité disponible, tout en préservant le niveau de qualité.

Aucune base concrète n'était présente lors de mon arrivée. Collaborant avec 2 organismes externes au CETIC, des recherches ont pu être effectuées dans le domaine de l'accessibilité en Wallonie permettant d'améliorer le statut actuel des PMR .

À la suite de plusieurs réunions en interne, les objectifs finaux ont été rapidement décrits et le développement a pu prendre forme. J'ai notamment pu explorer des technologies telles que Django, AngularJS et bien d'autres ...

Au travers de ce travail écrit, vous constaterez qu'un simple fichier texte peut être produit en un projet mobile complet via des technologies et des langages inconnus lors de mon arrivée.

Chapitre 2

CETIC

2.1 Présentation de l'entreprise

Le CETIC est le centre belge de recherche appliquée au service des entreprises dans le domaine des Technologies de l'Information et de la Communication (TIC).

Présent depuis 2001 sur le site de l'Aéropole de Charleroi, le CETIC y a été créé a l'initiative de l'Université de Namur (UNamur), de l'Université catholique de Louvain (UCL) et de la Faculté Polytechnique de l'Université de Mons (UMons). Grâce au relations privilégiées avec les différents laboratoires universitaires, les équipes du CETIC sont au cœur des progrès de la recherche, en Belgique, en Europe et dans le monde.

2.2 Départements de recherche

1. **Software & System Engineering**
   Aider les entreprises à concevoir des produits et services de meilleure qualité, à en assurer la fiabilité, l'accessibilité, la sécurité, le respect des normes internationales, en leur apportant un soutien méthodologique.
2. **Software & Services Technologies**
   Aider les entreprises à exploiter plus rapidement les nouvelles architectures informatiques réparties, à accélérer le processus de transformation d'information en connaissance par les technologies sémantiques, l'ouverture aux Open Data, à exploiter les réelles opportunités du logiciel libre et en mettant a leur disposition une expertise technologique de pointe.
3. **Embedded & Communication Systems**
   Aider les entreprises à embarquer plus d'intelligence et plus de connectivité dans les systèmes qu'elles développent, mettre au point les démonstrateurs technologiques, les prototypes, en exploitant et intégrant les nouvelles technologies électroniques.

2.3 Coordonnées de l'entreprise

CETIC Aéropole, bâtiment EOLE
Rue des Frères Wright, 29 boite 3
B-6041 Charleroi
Belgique

Chapitre 3

Access-I

3.1 À propos d'Access-i

L'asbl Access-i a pour mission de promouvoir l'information sur l'accessibilité des biens et services aux personnes à mobilité réduite.
Sur access-i.be, vous trouverez des informations fiables et objectives concernant le niveau d'accessibilité des bâtiments, sites et événements.
Un Access-i contient 7 cases représentant chacune une catégorie de personnes à mobilité réduite.
Le niveau d'accessibilité d'un espace ouvert au public est déterminé pour chacune de ces catégories.
Lorsque la couleur de la case est :

- verte, le site est accessible en autonomie,
- orange, le site est accessible avec un coup de main ponctuel,
- grise, le site n'est pas accessible de manière satisfaisante ou rien de spécifique n'est prévu.

Des informations plus détaillées (parking, entrée, circulation, signalétique, etc.) pour chaque catégorie sont également disponibles.
Pour chaque fiche descriptive (d'un bâtiment ou d'un événement), un onglet spécifique pour chacune des catégories précise ces informations sous forme de points forts et de points faibles.

Personne a mobilité réduite

Un PMR, aussi décliné comme personne à mobilité réduite, est une situation de handicap suite a un accident, une maladie ou bien dû a la vieillesse. Une mobilité réduite entraine une réduction à l'autotomie de déplacement, limité ou inexistant dans la vie de tous les jours.

3.2 Les assosiations

Créée en avril 2010, l'asbl se compose de 11 associations membres :

- Acces-a ;
- Altéo ;
- AMT Concept ;
- Association Belge des Paralysés (ABP) ;
- Association Francophone d'Aide aux Handicapés Mentaux (AFrAHM) ;
- Association Nationale pour le Logement des personnes Handicapées (ANLH) ;
- Association Socialiste de la Personne Handicapée (ASPH) ;
- Groupe d'Action pour une Meilleure Accessibilité aux personnes Handicapées (Gamah) ;
- Groupe d'Entraide pour Hémiplégiques (GEH) ;
- Passe le message à ton voisin ;
- Plain-Pied.

dans la suite de ce chapitre, je détaillerai les 2 principales associations actives sur le terrain et avec lesquelles j'ai eu des contacts durant ce stage : Gamah et l'ANLH.

3.3 Gamah

3.3.1 Présentation de l'entreprise

Depuis plus de 30 ans, Gamah (Groupe d'Action pour une Meilleur Accessibilité au personnes Handicapées) améliore l'accessibilité des bâtiments, des cheminements et des transports. Leurs conseillers se basent sur les réglementations existantes mais également sur les bonnes pratiques que leur équipe a pu définir grâce à l'expérience et à l'expertise engrangées.

3.3.2 Les missions de Gamah

1. **le conseil** aux architectes et aux maîtres d'œuvre ;
2. **la formation** des acteurs du bâtiment, de la voirie et des transports ;
3. **l'évaluation** de l'accessibilité de l'environnement bâti ;
4. **l'interpellation** des constructeurs et décideurs politiques au sujet des besoins des PMR.

3.4 ANLH

3.4.1 Présentation de l'entreprise

Depuis plus de 30 ans, l'ANLH avec l'aide de professionnels handicapés et valides assure la promotion de l'intégration sociale des personnes handicapées physiques. Dans ce but, elle milite en faveur de l'accessibilité et de l'adaptation des logements.

Leurs activités se concentrent principalement autour de 5 grands axes :

- les logements pour personnes handicapées
- les services d'Aide à la Vie Journalière
- l'accessibilité
- les transports
- la diffusion

3.4.2 Activités de l'ANLH

Les logements adaptés

Pionnère en matière de logements adaptés pour personnes handicapées, l'ANLH avec les Sociétés de Logements Sociaux est à l'origine de la construction, dans toute la Belgique, de plusieurs centaines de logements sociaux adaptés, reliés ou non, à des services d'aide à la vie journalière (AVJ).

L'ANLH publie régulièrement des ouvrages reprenant les prescriptions et les normes en matière de logements adaptables, accessibles, adaptés ou adaptés avec AVJ.

Les services d'Aide à la Vie Journalière (AVJ)

L'ANLH participe activement à la création et au développement de services d'aide à la vie journalière à Bruxelles et en Wallonie. Ces services ont pour objectif de permettre à des personnes handicapées physiques graves de vivre de façon autonome dans un logement privé grâce à un service 24h/24. Ces services ne remplacent pas, mais complètent les services à domicile existants (infirmières, aides familiales, kinés,...).

L'accessibilité

L'ANLH a pour objectif que toutes les personnes handicapées physiques puissent circuler aisément partout. C'est pourquoi, elle s'investit dans différents projets, tant au niveau national qu'européen, dont le but est de rendre l'environnement plus accessible.

Pour l'aider dans sa mission, l'ANLH a également créé une Agence de Conseil, de Consultation et d'Expertise en matière d'Accessibilité : ACCES-A.

Par son expérience et son investissement dans le secteur de la personne handicapée, l'ANLH est devenue une référence en matière d'accessibilité et a publié de nombreux ouvrages traitant de ce sujet.

Les transports

L'ANLH s'est positionnée comme conseillère en matière d'adaptation des moyens de transport : train, tram, bus, métro, taxis, minibus,... Elle suscite et s'investit dans différents projets visant cet objectif.

La diffusion

Pour répondre aux besoins d'information des personnes handicapées, des professionnels du secteur social ou architectural,... l'ANLH a mis en place différents outils.

Un site internet qui reprend les différentes activités de l'association, les législations en vigueur, les informations utiles, les publications parues,... www.anlh.be

Un ensemble de banques de données innovantes "accesatout" qui permet de trouver sur internet de nombreuses informations en matière d'aides techniques, de logement, de services,...

Chapitre 4

AccessI-Mobile

4.1 Présentation

AccessI-Mobile est une application mobile qui a pour objectif d'accompagner les personnes à mobilité réduite au travers de leurs smartphones. L'application sera multiplateforme, c'est à dire porté par les différents systèmes comme IOS et Android, afin d'atteindre un large public et basé sur le même principe que le site parent : AccessI.

Actuellement les personnes à mobilité réduites (PMR) disposent de peu de données fiables et intégrées relatives à des infrastructures dans lesquelles elles souhaitent se rendre (ex. musée, centre commercial,...) et aux moyens de transports accessibles à proximité (trains, bus,...).

L'objectif du travail a été d'abord d'inventorier des données disponible dans ces domaines, ensuite de les consolider et de les rendre disponible via une interface ouverte et enfin de bâtir une application mobile permettant aux PMR d'y accéder le plus aisément possible lors de leurs déplacements.

4.2 Etat de l'art

Pour préparer l'application, nous devons effectuer certaines recherches structurées concernant le domaine ciblé et fixer les limites de recherche.

Le but d'un état de l'art nous permet de diviser un projet en plusieurs sections et de classifier les outils ou méthodes qui seront utilisés durant la phase de développement.

Après avoir décomposé le sujet en différentes parties, j'ai pu regrouper les éléments par thèmes :

- Open Data
  - ETL
  - Moteur de stockage
  - Web sémantique & Linked Open Data
- Inventaire des sources de données
  - données public
  - données privé (asbl, entreprise, indépendant)

- Technologie
  - Framework de publication
  - Framework mobile
- Spécification
  - public ciblé
  - zone géographique pour le prototype
- Développement

4.3 Réunions CETIC, Gamah & ANLH

Lors de plusieurs réunions, nous avons procédé à des discussions sur les données ouvertes disponibles et privées. Le principal problème est que les données sur l'accessibilité sont majoritairement produites par des secteurs privés comme Gamah et l'ANLH qui restent isolés du monde extérieur.

Pour combler ce problème nous avons proposé la mise en place d'une application qui permettrait d'avoir accès à ces données depuis n'importe quel endroit du monde.

L'objectif aussi est de pouvoir agrémenter les données proposées par Gamah avec d'autres organismes comme : "jaccede.com", qui possèdent des données sur l'accessibilité en Belgique et en France. Le problème de ce genre d'organisme est que leurs données sont eux aussi privées donc inaccessible directement.

Après un contact avec le département informatique de "jaccede.com", nous avons reçu l'autorisation d'accéder à leurs données et de pouvoir "mixer" leurs informations avec ceux de chez Gamah.

Les réunions avec les associations ont eu lieu tous les mois environ, ce rythme correspondait à la durée des phases de travail (sprints agiles). La première réunion (en février) m'a permis de faire connaissance avec les associations et fixer les objectifs, la seconde (mi-mars) à permis de présenter mon inventaire de données, de définir un jeu de données de validation et présenter mon idée d'application mobile, la quatrième (mi-mai) a permis de présenter mon application.

En interne nous avons établi un cahier de charge suivant les recherches effectuées auparavant et sur les technologies à utiliser. Aussi nous avons fixé les limites qu'un prototype doit remplir et de son back-end.

4.4 Fonctionnalité de l'application

4.4.1 Accueil

- Interface d'accueil avec les dernières informations

4.4.2 Menu

- Menu latéral
- Disponible sur toute les interfaces

- Redirection : accueil, carte, favoris, configuration, à propos

4.4.3 Carte

- Afficher tous les points d'intérêt comme les bâtiments, hôtels, restaurants, châteaux ...
- Centrer la carte sur la position actuelle
- Lors d'une interaction avec un point d'intérêt, redirection sur une interface détaillée
- Possibilité de zoomer et dézoomer la carte

4.4.4 Détails

- Détail du point d'intérêt dans son ensemble
- Accès aux commentaires postés par la communauté
- Accès aux moyens de mobilité à proximité
- Possibilité d'ajouter un suivi sur ce point d'intérêt

4.4.5 Commentaires

- **Lecture**
  - Affiche la liste de commentaires déjà présents
  - Possibilité de rédiger un commentaire
  - Possibilité de lire un commentaire
- **Rédaction**
  - Rédiger un commentaire détaillé
- **Détail**
  - Possibilité d'évaluer le commentaire
  - Possibilité de lire un commentaire dans son ensemble
  - Possibilité de signaler un abus

4.4.6 Mobilité

- Affiche les arrêts de bus et gare le plus proche de ce point d'intérêt
- Affiche la distance de vol entre votre position et le moyen de transport

4.5 Du coté back-end

En rapport à la publication de données, nous avons eu l'idée de mettre en place une application Web de type REST qui donnerait accès aux données depuis une base de données. La technologie qui sera utilisée n'a pas encore été définie et des tests seront effectués afin de choisir la bonne technologie adaptée au sujet.

Chapitre 5

Open Data

5.1 Introduction

Une donnée ouverte est une donnée numérique d'origine publique ou privée. Elle peut être notamment produite par une collectivité, un service public ou une entreprise. Elle est diffusée de manière structurée selon une méthodologie et une licence ouverte garantissant son libre accès et sa réutilisation par tous, sans restriction technique, juridique ou financière.

L'ouverture des données (en anglais Open Data) représente à la fois un mouvement, une philosophie d'accès à l'information et une pratique de publication de données librement accessibles et exploitables.

Elle s'inscrit dans une tendance qui considère l'information publique comme un bien commun, dont la diffusion est d'intérêt public et générale.

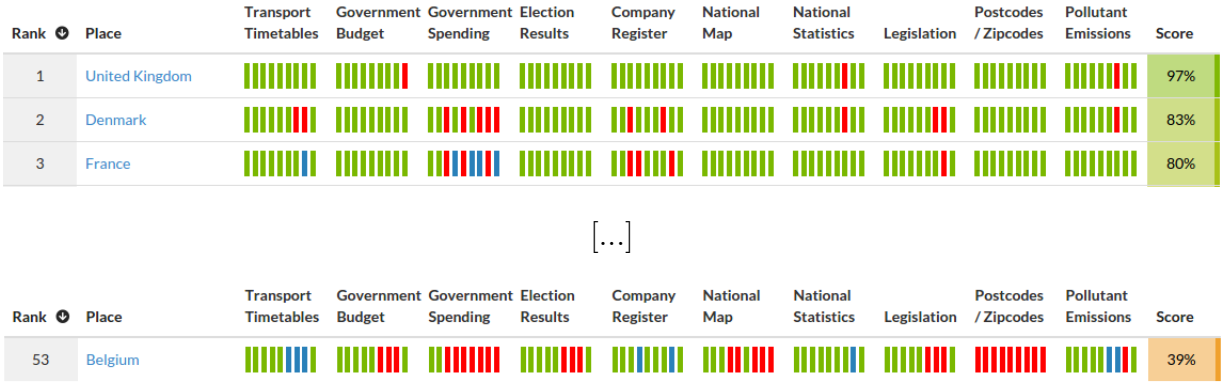
5.2 5 stars Open Data

Afin de rendre homogène les Open Data, Tim Berners-Lee a suggéré un schéma de déploiement pour les données ouvertes : 5 Stars Open Data.

⋆
Rendre disponible les données en ligne, aucun format est exigé, mais libre d'accès (open licence)
⋆⋆
Structurer les données (on favorisera des formats Excel que des documents PDF)
⋆ ⋆ ⋆
Utiliser un format non propriétaire, les formats comme Excel nécessitent des licences au contraire des fichiers CSV et JSON
⋆ ⋆ ⋆⋆
Utiliser des URI de sorte que l'on peut pointer vers les sources plus facilement
⋆ ⋆ ⋆ ⋆ ⋆
Possibilité de lier les données à d'autres, le format RDF est le plus utilisé

5.3 Au niveau Européen

D'après le "Global Open Data Index(2013)", la Belgique serait placée en 53éme places bien loin derrière nos pays frontaliers comme les Pays-Bas, l'Allemagne ou bien encore la France qui se positionne dans le top 3.



Rank	Place	Transport Timetables	Government Budget	Government Spending	Election Results	Company Register	National Map	National Statistics	Legislation	Postcodes / Zipcodes	Pollutant Emissions	Score
1	United Kingdom											97%
2	Denmark											83%
3	France											80%

[...]

Rank	Place	Transport Timetables	Government Budget	Government Spending	Election Results	Company Register	National Map	National Statistics	Legislation	Postcodes / Zipcodes	Pollutant Emissions	Score
53	Belgium											39%

Durant l'état de l'art, j'ai pu rencontrer deux problèmes majeurs:

Le premier étant que la Belgique est pauvre en jeu de données. Certains sites comme awt.be ne proposent pas assez de ressources utilisables. Par contre openknowledge.be a mis en place une API comme IRAIL qui permet de récupérer en temps réel les situations des voiries en Belgique et les emplacements de chaque gare ferroviaire. Ce type d'organisation est externe aux entreprises ferroviaires, comme la SNCB, et a pour but de promouvoir les Open Data en Belgique.

Le second est le type de format. Pour donner un exemple, la ville de Liège a rendu accessibles les places de parking pour handicapés au format PDF via des marqueurs rouges sur une carte ... Nul besoin de vous dire que ce genre de format est mis directement sur le côté.

5.4 Jeu de données pour l'accessibilité

Source	Description	Format	Categorie	Zone
data.irail.be	Station de la STIB	JSON/API	Transport	Bruxelles
data.irail.be	Station de Delijn	JSON/API	Transport	Flandre
data.irail.be	Données des voies ferroviaires	JSON/API	Transport	Belgique
opendata.awt.be	Données du TEC	GTFS	Transport	Wallonie
trafiroutes.wallonie.be	Embarras de circulation	Flux rss	Mobilité	Wallonie
data.irisnetlab.be	Embarras de circulation	JSON	Mobilité	Bruxelles
api.data.be	Information sur les entreprises belges	API	Entreprise	Belgique
access-i.be	Bâtiment & Evénements pour PMR	Excel	bâtiment	Wallonie
anlh.be	Bâtiment détaillé pour PMR	Excel	bâtiment	Wallonie
jaccede.com	Bâtiment détaillé pour PMR	Json/API	bâtiment	Belgique

Chapitre 6

mETL, Extract Transform Load

6.1 Introduction

Comme annoncé précédemment, pour arriver au plus haut niveau du schéma de Tim Berners-Lee, nous devons rendre les données accessibles dans un format structuré. C'est à ce moment que les ETL rentrent en jeu.

“ mETL ” permet de transformer un format en un autre d'après une structure donnée. Son fonctionnement reste théoriquement assez simple si nous n'avions pas besoin de modifier la structure du jeu de données.

Cet outil a été développé en Python par “Bence Falude” et ses collaborateurs tenant en compte au maximum de l'utilisation optimale de la mémoire de l'ordinateur ou du serveur qui exécutera l'outil.

6.2 Qu'est-ce qu'un ETL ?

Extract-Transform-Load est connu sous le terme ETL. Il s'agit d'une technologie informatique intergicielle permettant d'effectuer des synchronisations massives d'information d'une source de données vers une autre. Selon le contexte, on est amené à exploiter différentes fonctions, souvent combinées entre elles comme l'extraction, la transformation ou la conversion.

L'objectif est l'intégration ou la ré-exploitation de données d'un réservoir source dans un réservoir cible.

À l'origine, les solutions d'ETL sont apparues pour le chargement régulier de données agrégées dans les entrepôts de données (ou datawarehouse), avant de se diversifier vers les autres domaines logiciels. Ces solutions sont largement utilisées dans le monde bancaire et financier, ainsi que dans l'industrie, au vu de la multiplication des nombreuses interfaces.

6.3 Utilisation de mETL

L'utilisation de " mETL " se compose de quatre étapes :

1. Génération du fichier de configuration
2. Vérification, transformation et manipulation du mappage
3. Ajout d'une sortie
4. Application de la configuration avec l'outil

6.3.1 Etape 1 : metl-generate

Cette fonctionnalité permet de générer un fichier au format YAML qui a pour but de comprendre la structure de la ressource pour l'ETL.

La commande :

```
$: metl−generate csv config.yml
```

Par défaut la commande n'a besoin que de 2 paramètres: le type de fichier entrant et l'emplacement où se trouvera le fichier de configuration.

Pour rendre la configuration plus rapide, des paramètres optionnels sont prévus à cet effet :

```
Usage: metl−generate [options] CONFIG_FILE SOURCE_TYPE

Options:
−h, −−help Permet d'afficher ce message
−l LIMIT, −−limit=LIMIT
Place une limite d'enregistrement
−−delimiter=DELIMITER
−−quote=QUOTE
−−skipRows=SKIPROWS
−−headerRow=HEADERROW
−−resource=RESOURCE
−−encoding=ENCODING
−−username=USERNAME
−−password=PASSWORD
−−realm=REALM
−−host=HOST
```

À noter que la liste pour les fichiers d'entrée et de sortie est assez vaste :

Type de source (INPUT):

- CSV, TSV, XLS, XLSX, Google SpreadSheet
- PostgreSQL, MySQL, Oracle, SQLite, Microsoft SQL Server
- JSON, XML, YAML

Type de sortie (OUTPUT):

- CSV, TSV, XLS, XLSX, Google SpreadSheet
- PostgreSQL, MySQL, Oracle, SQLite, Microsoft SQL Server
- JSON, XML, YAML
- Neo4j

Exemple d'utilisation

```
$: metl−generate −−resource parking.csv −−headerRow 0 −−skipRows 1 csv config.yml
```

Un fichier de configuration "config.yml" sera présent dans le dossier actuel lors de léxécution de cette commande

```
source:
fields:
− map: ''
name: ''
type: Integer
− map: adresse_fr
name: adresse_fr
type: String
− map: place
name: place
type: String
− map: geo_y
name: geo_y
type: String
− map: geo_x
name: geo_x
type: Float
− map: adresse_nl
name: adresse_nl
type: String
headerRow: '0'
resource: parking.csv
skipRows: '1'
source: CSV
target:
silence: false
type: Static
```

Ce qu'on peut remarquer, c'est que l'ETL ne possède pas une intelligence artificielle.

Il donnera cette configuration par rapport au fichier source sans interprétation. Si un délimiter est mal positionné dans ce fichier(parking.csv), mETL prendra en compte qu'aucun problème n'est présent.

6.3.2 Etape 2 : Configuration

Dans le cas de parking.csv, des erreurs d'interprétation sont apparues. Pour régler ce problème, l'édition du fichier "config.yml" sera nécessaire et manuelle.
Via un éditeur de texte, les lignes 3 à 5 seront supprimées et le type en ligne 14, laissera place a un Float.

Exemple d'utilisation

```
1  source:
2  fields:
3  − map: adresse_fr
4  name: adresse_fr
5  type: String
6  − map: place
7  name: place
8  type: String
9  − map: geo_y
10 name: geo_y
11 type: Float
12 − map: geo_x
13 name: geo_x
14 type: Float
15 − map: adresse_nl
16 name: adresse_nl
17 type: String
18 headerRow: '0'
19 resource: parking.csv
20 skipRows: '1'
21 source: CSV
22 target:
23 silence: false
24 type: Static
```

Pour tester cette nouvelle configuration, une commande est disponible:

```
1  $: metl config.yml
```

Cette fonctionnalité est normalement prévue pour la 3e étape du processus. Pour le moment, il est utilisé pour "tester" la configuration car le type de sortie est en mode de débogage.

Le contenu du fichier "parking.csv" sera affiché dans la console sous la forme d'une table de données.

Exemple d'utilisation avancée

Dans d'autres cas, certains fichiers sont mal préparés pour être lus par un logiciel. Des listes dans un objet, qui est lui-même contenu dans un autre objet, enfin soit, des structures mal étudiées.

Pour remédier à ce problème, mETL propose avec beaucoup de liberté, des méthodes de transformation et de manipulation : **Transform**.
En passant par un simple Upper/Lower-Case a du Split de String, cet ETL n'a pas encore fini de proposer du contenu et de la flexibilité de manipulation.

Pour illustrer sa complexité, voici un exemple de configuration:

```
source:
fields:
− map: features/∗
name: features
type: List
− name: elementList
type: Complex
− name: geo_x
type: String
− name: geo_y
type: String
− name: adresse
type: String
− name: cause
type: String
resource: travaux_bruxelles.json
source: JSON
manipulations:
− expand: ListExpander
listFieldName: features
expandedFieldName: elementList
− modifier: SetWithMap
fieldNamesWithMap:
geo_x: geometry/coordinates/0
geo_y: geometry/coordinates/1
adresse: properties/street_name
cause: properties/cause
complexFieldName: elementList
− filter: DropField
fieldNames:
− features
− elementList
target:
silence: false
type: Static
```

Dans cet exemple, le fichier source contient un objet de type List, mais illisible sans transformation.
Les manipulations permettent d'extraire chaque élément dans des objets qui sont eux-mêmes mappés avec des règles de typage.

Remarque : Aucune limite n'est possible, dans le cas où aucune fonctionnalité proposée par mETL ne répond à vos besoins, il est possible d'inclure ses propres fonctions développées en Python.

6.3.3 Etape 3 : Ajout d'une sortie

Après que les données soient lisibles depuis le fichier source, et que les manipulations et transformations sont prêtes. L'ETL doit être préparé à charger (**Load**) toutes les données vers une cible.

Dans la plupart des cas, les données seront envoyées vers une base de données, le plus souvent vers PostGreSQL. Rien ne vous empêche d'écrire les données dans un fichier JSON ou XML.

Dans le cadre du stage, j'ai utilisé PostGreSQL pour faciliter la mise en place de la publication de données.

Exemple de Target

```
source:
[...]
target:
createTable: true
table: accessi_batiment
truncateTable: true
type: Database
url: postgresql://postgres:azerty@localhost:5432/api
```

Dans le cas présent, nous établirons une connexion PostGreSQL vers la base de données qui se nomme "api".

Paramètre du Target

- url : connexion au SGBD
- table : nom de la table qui va recevoir les données
- createTable: (true/false) permets de créer la table si inexistante
- replaceTable: (true/false) permets de remplacer la table si existante ainsi que sa structure
- truncateTable: (true/false) vide au complet la table avant l'insertion
- addIDKey: (true/false) génère directement une clé primaire sous le nom de 'id'

6.3.4 Etape 4 : Exécution de la configuration

La dernière étape du processus consiste à appliquer les règles énumérées dans le fichier de configuration.
La commande :

```
1  $: metl config.yml
```

Le temps que prendra l'outil sera en rapport au traitement des données, mais généralement quelques secondes suffisent réellement.

6.3.5 Conclusion

Dans l'ensemble mETL est un outil très utile pour le traitement de données. Sa facilité qui permet de transformer un fichier structuré vers une base de données offre un gain de temps non négligeable pour un développeur.

Dans le cadre du stage, j'ai dû utiliser les données de AccessI au format Excel et mETL m'a permis d'éviter de développer une application permettant d'extraire ces données pour les charger dans une base de données.

Le seul point négatif est que mETL n'est pas fait pour des novices. D'un côté il faut savoir naviguer dans l'environnement Linux pour mettre en place l'outil et ses dépendances, mais aussi de savoir coder en Python pour répondre à vos besoins si l'outil ne propose pas de méthode à vos manipulations.

Pour conclure, mETL est puissant, rapide et facile d'utilisation. Cet outil fait gagner un temps précieux pour le chargement de données, mais si vous n'êtes pas familier avec l'environnement Linux, d'autres ETL au format GUI sont proposés sous le système Windows.

6.4 Autres ETL

Voici une liste non exhaustive d'ETL sous licence gratuite et payante :

- Pentaho data integration (kettle)
- SQl Server (SSIS)
- Jasper Soft ETL, Talend
- Talend Open Studio
- Clover ETL
- Essbase Integration Services (EIS)
- bubbles

Chapitre 7

Vagrant, VirtualBox et Ansible

7.1 Introduction è la virtualisation automatisée

Pour un analyse développeur, la virtualisation est essentielle afin d'avoir des environnements isolés. Le plus populaire logiciel de virtualisation, **VirtualBox**, permet la mise en place de machine virtuelle avec n'importe quel OS en utilisant les ressources matériels de l'ordinateur. Ce qu'on peut reprocher à ce genre de logiciel est le manque d'automatisation. Dans le cas où vous avez besoin d'installer un serveur Debian avec un environnement Python pour tester un outil par exemple, vous serez confronté à rester devant le poste afin d'installer le tout manuellement.

Marier avec **Ansible**, **Vagrant** permet de mettre en place une machine virtuelle via une seule ligne de commande et un dossier de configuration. Il permet d'avoir un environnement souhaité et redéployable rapidement, au cas où votre machine ne répond plus à vos attentes, après des opérations manuelles.

Un autre avantage se positionne autour de la communauté **Ansible**. Sur des sites de partage, comme **github.com**, des dossiers de configuration se trouvent en libre accès et utilisation. Dites-vous que si vous devez déployer un serveur web avec Django et une base de données MySQL, une autre personne l'a déjà fait pour vous.

VirtualBox Provider de virtualisation (crée des VM) - alternatives: VMWare, AWS, OpenStack, ...

Ansible Provisioning (organise l'environnement) - alternatives: chef, puppet, scripts shell

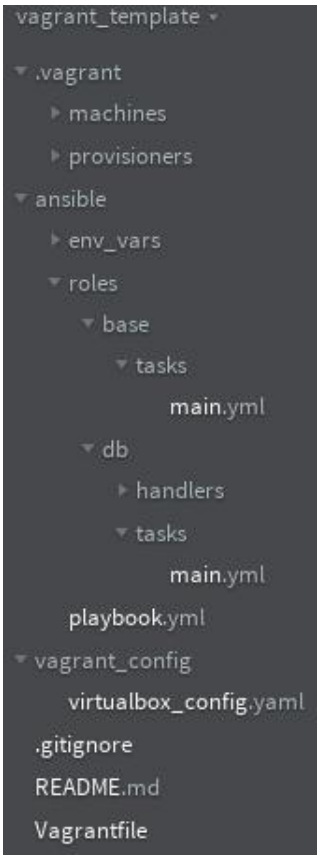
Vagrant Orchestre les providers et les provisionneurs

7.2 Fonctionnement et déploiement

Afin de déployer une machine virtuelle, Vagrant nécessite des fichiers de configuration. Ils sont regroupés en 3 grandes sections :

- Variables d'environnement Ansible
- "playbook" Ansible
- Configuration système, matériel et réseau

Structure d'un projet Vagrant



Cette figure montre comment organiser un dossier Vagrant. Le fichier principal se nomme "Vagrantfile" et permettra par la suite d'appliquer toutes les configurations définies par son utilisateur via les rôles Ansible.

7.2.1 Section 1 : Variables d'environnement Ansible

Les variables d'environnement Ansible permettent de stocker des informations dans des fichiers isolés.

Un exemple classique d'utilisation est la gestion des comptes de bases de données. Pour éviter d'écrire les noms d'utilisateur et mots de passe dans les rôles, les fichiers d'environnement permettent d'isoler ces informations afin d'avoir une meilleure lisibilité lors d'une édition. De même, ça permet de ne pas stocker les informations sensibles dans un VCS (ex : git ignore).

exemple de fichier d'environnement : *ansible/env_vars/base.yml*

```
1  #Configuration generale du projet
2
3  project_name: 'API REST Dango Framework + PgSQL'
4  application_name: 'REST_DJANGO'
```

Cette configuration annonce à Ansible que le nom du projet se nommera "API REST Django Framework + PGSQL". Si par la suite le nom venait à changer, il suffirait de changer cette variable avec un éditeur de texte.

De même pour la configuration des bases de données, le fichier "*ansible/env_vars/local.yml*" se présente de cette manière :

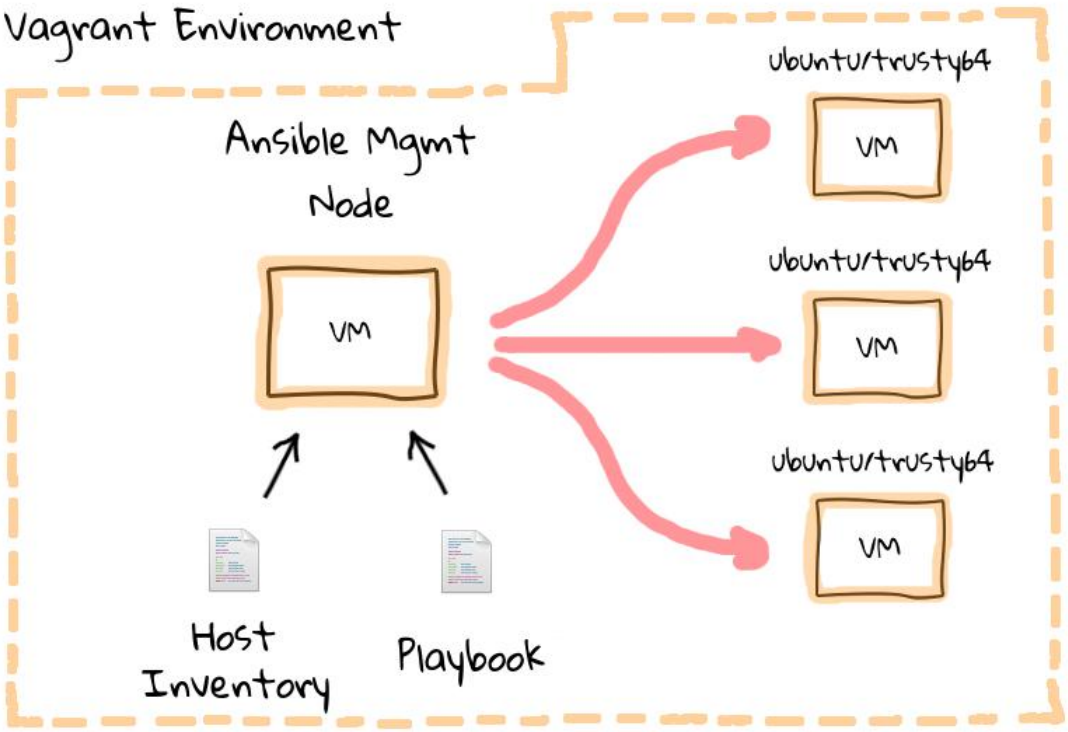
```
1  ---
2
3  # Configuration DB PgSQL
4  db_user: "postgres"
5  db_name: "api"
6  db_password: "C8mR397aVFu5vb"
```

Ce qu'il faut retenir des variables d'environnement Ansible est qu'il est possible de regrouper les informations pertinentes, comme les mots de passe, a l'extérieur des configurations. Ceci permet une meilleure lisibilité et organisation en cas d'édition.

7.2.2 Section 2 :"Playbook" Ansible

Les rôles Ansible, aussi appelés Playbook en Anglais, contiennent toutes les étapes qui doivent être exécutées par Vagrant afin d'avoir l'environnement de travail souhaité. Beaucoup de Playbook sont disponibles à travers des sites de partage, mais la documentation sur "Ansible" est assez complète.

Positionnement des Playbook dans un projet Ansible



Pour faire simple, une machine virtuelle déployée avec Vagrant contient une configuration Ansible. Cette dernière se compose de Playbook dans le but de compléter la configuration de base du système choisi. Chaque Playbook englobe des tâches à exécuter, par exemple l'installation de package ou encore une configuration réseau avancée.

Généralement, les tâches sont classées par catégorie (service web, base de données, framework) afin de mieux se retrouver lors d'un partage ou d'une édition.

exemple de rôle : ansible/roles/db/tasks/main.yml

```
1  ---
2
3  - name: Install PostgreSQL
4  apt: name={{ item }} update_cache={{ update_apt_cache }} state=installed
5  with_items:
6  - postgresql
7  - postgresql-contrib
8  - libpq-dev
9  - python-psycopg2
10 - phppgadmin
11 tags: packages
12
13 - name: Ensure the PostgreSQL service is running
14 service: name=postgresql state=started enabled=yes
15
16 - name: Ensure database is created
17 sudo_user: postgres
18 postgresql_db: name={{ db_name }}
19 encoding='UTF-8'
20 lc_collate='en_US.UTF-8'
21 lc_ctype='en_US.UTF-8'
22 template='template0'
23 state=present
24
25 - name: Ensure user has access to the database
26 sudo_user: postgres
27 name={{ db_user }}
28 postgresql_user: db={{ db_name }}
29 password={{ db_password }}
30 priv=ALL
31 state=present
32
33 - name: Ensure user does not have unnecessary privileges
34 sudo_user: postgres
35 postgresql_user: name={{ db_user }}
36 role_attr_flags=NOSUPERUSER,NOCREATEDB
37 state=present
```

Cette configuration met en place un environnement PostgreSQL dès le premier lancement de la machine virtuelle. Aucune commande manuelle ne devra donc être appliquée pour son installation.

A noter que les variables d'environnement ont un impact pour ce genre de rôles. À la ligne 27, l'utilisation de la variable "db_name" mentionnera le nom de la base de données définit dans le fichier "*ansible/env_vars/local.yml*"

7.2.3 Section 3 : Configuration système, matériel et réseau

Après avoir créée le squelette du projet Vagrant, nous devons préparer le fichier principal "Vagrantfile" à appliquer toutes ces règles et de définir son type d'environnement.

Pour le choix de l'OS, les machines virtuelles sont principalement basées sur l'architecture Linux dont il existe un nombre assez conséquent de distributions possibles telles qu'Ubuntu, Debian, OpenSUSE, Fedora ...

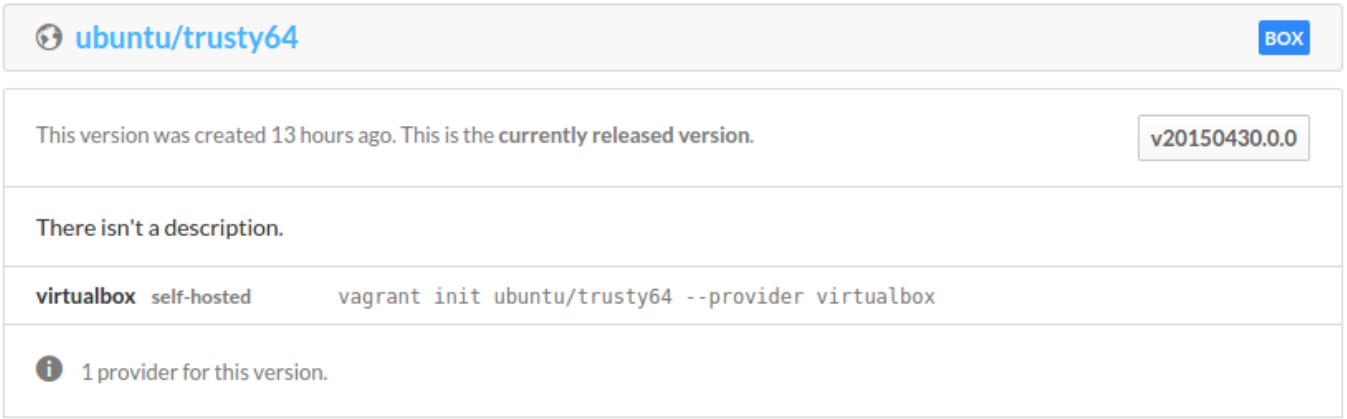
Dans le cadre du projet, mon poste de travail étant sous Ubuntu Desktop, je n'ai pas cherché très longtemps pour me diriger sur une box Ubuntu Server vu que les commandes d'environnement sont identiques.

Configuration du serveur : "vagrant_config/virtualbox_config.yml"

```
1  #configuration de la machine
2
3  box: "ubuntu/trusty64"
4  box_url: "https://atlas.hashicorp.com/ubuntu/boxes/trusty64"
5
6  ip: "192.168.33.15"
7  memory: 1024
8  cpus: 2
```

Cette configuration utilisera une box préconfigurée sous Ubuntu trusty64 depuis le Cloud de "hashicorp.com". Par la même occasion, le fichier définira l'ipv4 de base et la configuration matérielle (la mémoire ram et le nombre de cœurs à utiliser) pour son déploiement.

Versioning de la box ubuntu/trusty64 depuis le Cloud hashicorp.com



Configuration du projet : “Vagrantfile”

```
1  # −∗− mode: ruby −∗−
2  # vi: set ft=ruby
3  require 'yaml'
4
5  project_name = "Accessi API Django Rest"
6
7  Vagrant.configure("2") do |config|
8  config.env.enable
9
10 config.vm.box = "ubuntu/trusty64"
11
12 config.vm.synced_folder ".", "/vagrant", type: "rsync",
13                                rsync__exclude: [".vagrant/", "Vagrantfile∗", "aws_config∗"]
14 config.vm.synced_folder "/vagrant/sync", "/vagrant/sync"
15
16 config.vm.define :local do |local|
17 # apt cache
18 if Vagrant.has_plugin?("vagrant−cachier")
19 config.cache.scope = :box
20 end
21
22 # Virtualbox settings
23 local.vm.provider :virtualbox do |vb, override|
24 conf = YAML.load_file('vagrant_config/virtualbox_config.yaml')
25 vb.name = project_name
26 local.vm.box = conf["box"]
27 local.vm.box_url = conf["box_url"]
28 local.vm.network "private_network", ip: conf["ip"]
29 vb.cpus = conf["cpus"]
30 vb.memory = conf["memory"]
31 #vb.gui = true
32 end
33
34 # Ansible provisioner.
35 local.vm.provision "ansible" do |ansible|
36 ansible.playbook = "ansible/playbook.yml"
37 ansible.host_key_checking = false
38 ansible.verbose = "v"
39 end
40 end
41 end
```

Le fichier “Vagrantfile” est le centre du projet Vagrant. Il appellera toutes les configurations du système, mais aussi tous les Playbook à installer.

Toujours dans ce même fichier, la configuration réseau se fera facilement via des variables Vagrant comme à la ligne 28. Lors de la création de la machine virtuelle, Vagrant s’occupe entièrement de mettre en place les interfaces réseau d’après des templates.

7.3 Utiliser la machine Vagrant

La commande pour générer et démarrer une machine virtuelle est : **vagrant up**

La première fois que cette commande sera exécutée, si la machine n'a pas déjà été créée, Vagrant mettra en place tout l'environnement comme décris dans le dossier de configuration. Aucune autre commande ne sera nécessaire pour l'installation de vos logiciels et outils.

À la suite de l'installation, il existe une commande qui permettra de faire une connexion entre votre host et votre guest via le protocole SSH: **vagrant ssh**

D'autres commandes sont disponibles afin d'administrer vos différents projets.

Commande principale de Vagrant :

```
1  #demarre la machine & la cree si inexistant
2  root$: vagrant up
3  # relance la machine
4  root$: vagrant reload
5  # stop la machine
6  root$: vagrant halt
7  # affiche l'etat actuel de la machine
8  root$: vagrant status
9  # se connecte la machine
10 root$: vagrant ssh
11 # supprime la machine
12 root$: vagrant destroy
```

Synchronisation de dossier

À noter qu'il est possible de synchroniser des dossiers entre votre host & votre guest. Ceci permet de modifier le code de l'application sur l'host, les modifications sont faites en même temps sur le guest où est installé le serveur d'application. Ex: changer index.html sur mon host va changer mon site web exposé par la VM.

Chapitre 8

Python et Django Rest Framework

8.1 Introduction au Python

Le Python est un langage de programmation orienté object similaire au Perl et Ruby. Sa force majeure est qu'il propose des milliers de modules qui vous feront gagner un temps précieux durant le développement de vos applications.

Pour ce qui est de l'apprentissage, le Python est très intuitif grâce à son typage dynamique fort et à sa communauté présente sur les forums. Entre autres, des sites sont disponibles pour apprendre rapidement ce langage avec des thématiques comme l'Open Data : **dataquest.io**.

8.2 Methode REST

Pour rendre accessible des données via un site, il existe les méthodes dites REST(representational state transfer). Il s'agit d'un ensemble de conventions et de bonnes pratiques à respecter et non d'une technologie entière.

L'information de base, dans une architecture REST, est appelée ressource. Toute information qui peut être nommée est une ressource : la description d'un bâtiment, la liste des arrêts de bus ou n'importe quel concept. Dans un système hypermédia, une ressource est tout ce qui peut être référencé par un lien.

L'interface entre les composants est simple et uniforme. En HTTP, cette interface est implantée par les verbs GET, PUT, POST, DELETE, ... qui permettent aux composants de manipuler les ressources de manière simple. Par exemple quand un agent voudra récupérer la liste des arrêts de bus depuis l'application, il passera par la méthode GET qui lui retournera les ressources voulues.

8.3 Django

Django est un framework web Python de haut niveau qui encourage le développement rapide et propre. Gratuit et open source, Django vous permet d'éviter de réinventer la roue grâce à toutes les libraires disponibles en Python, mais aussi de tout ce que ce framework offre dès son installation.

Le fonctionnement d'une application Django se divise en 2 parties.

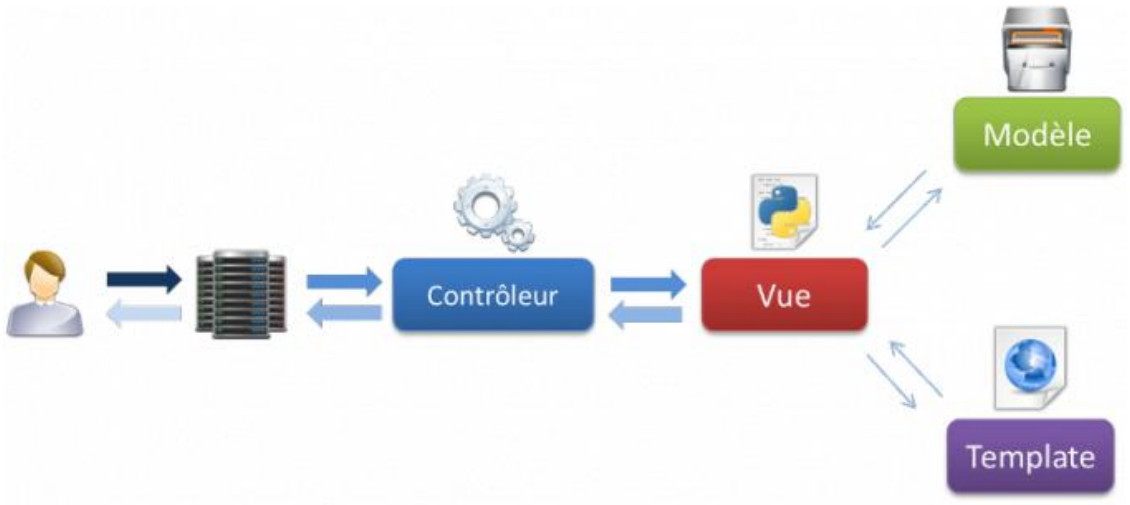
La première section a pour but de préparer l'étape entre l'utilisateur et l'application en elle même. Cette section gère la relation entre les bases de données et les applications Django, mais aussi s'occupe du routage via des règles URL.

La seconde section est l'application. Organisée en modèles, vues et templates, l'application se trouve au coeur du projet Django.

Utilisé par la NASA ou le Washington Post, Django prouve sa fiabilité et sa stabilité à travers ce type d'organisme.

8.4 Le modéle MVT

Django se base sur le modèle MVT, légèrement différent du modèle MVC, le framework gère lui même le contrôleur et laisse place au Template.

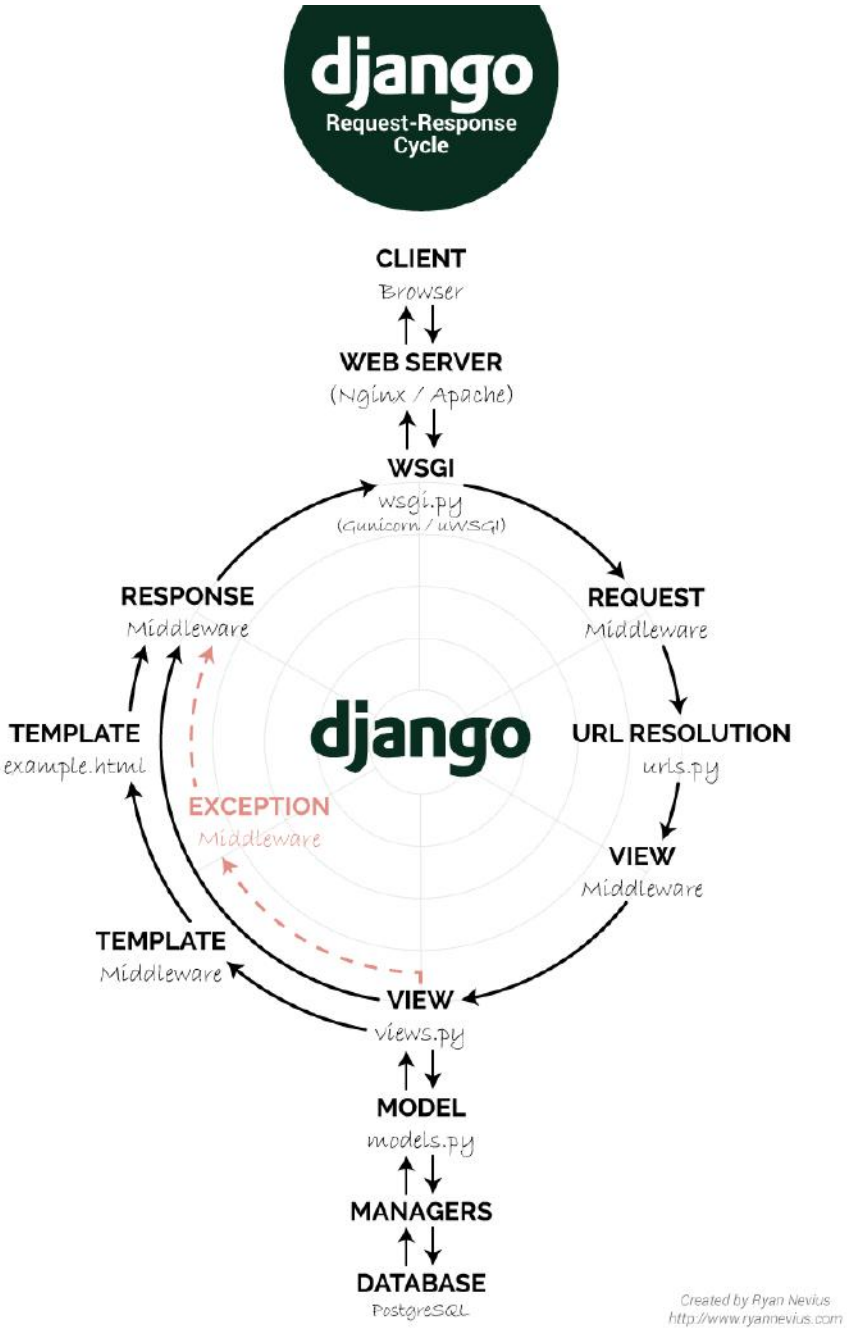


Lors d'une requête venant de l'utilisateur, le framework Django gère lui même, via les règles de routage défini par le développeur, de charger la bonne vue correspondante au résultat voulu.

Une **template** est un fichier HTML qui sera récupéré par la vue pour être envoyer a l'utilisateur, mais entre cette étape, Django va exécuter la template comme si c'était un fichier de code.

Inclus dans les templates, le framework propose l'utilisation des structures conditionnelles, des boucles, des variables... afin d'avoir une grande liberté de développement.

Présentation du cycle de Django



8.5 Rest Framework

Le framework **Django Rest** offre la possibilité de mettre en place une API qui a pour but d'autoriser l'accès aux ressources d'une ou plusieurs bases de données.Utiliser comme une application, il faudra lui décrire les modèles, vues et routage pour être fonctionnel.

Exemple d'utilisation

Pour expliquer au mieux le fonctionnement d'une application Rest, il est nécessaire de l'illustrer.

Dans le cas ou un client voudrait récupérer les données d'un article, celui-ci devra passer par une URI ressemblant à **http://api.yourcompany.com/articles/22**.

En décomposant l'URI, le framework interprète qu'il devra lire uniquement l'article 22 de la base de données.

Avec le framework Django Rest, seulement 4 fichiers devront être adaptés:

- url.py
- view.py
- serializers.py
- model.py

8.5.1 Routage - url.py

Le fichier **url.py** a pour but de gérer le routage de l'application. Pour faire simple, lorsqu'un client appel une ressource via un URI, ce fichier permet de faire la liaison entre l'extension de l'adresse et la vue adéquate.

Code source : url.py

```
1  from django.conf.urls import patterns, include, url
2  from rest_framework import routers
3  from rest import views
4
5  router = routers.DefaultRouter()
6  router.register(r'articles', views.ArticlesViewSet)
7
8  urlpatterns = patterns('',
9    url(r'^', include(router.urls)),
10 )
```

À ce niveau, l'application propose un URI : **http://api.yourcompany.com/articles/**

Lorsqu'un client accédera à cette adresse, Django ira lire le fichier url.py qui rédigera le client vers la vue : **ArticlesViewSet**.

8.5.2 Vue - views.py

La vue, dans son ensemble, affiche les données depuis une ressource. L'utilisation de "viewset" permet de passer par un template du framework et nécessitera que l'indication du modèle correspondant aux ressources de la vue.

Code source : views.py

```
1  from rest_framework import viewsets
2  from rest.models import Articles
3  from serializers import ArticlesSerializer
4
5  class ArticlesViewSet(viewsets.ModelViewSet):
6      queryset = Articles.objects.all()
7      serializer_class = ArticlesSerializer
```

Une vue sera alors créée et regroupera toutes les ressources d'article depuis le modèle **Articles**. Par ailleurs, un objet sera aussi mentionné pour la gestion de l'affichage.

8.5.3 Sérialisation : serializers.py

La sérialisation permet aux données, complexes telles que les modèles, d'être convertis en type de données natives pour Python. Ils peuvent ensuite être facilement rendus dans les formats JSON ou XML. Dans le cas présent, la sérialisation permet aussi de sélectionner les champs à afficher pour limiter la vue des données.

Code source : serializers.py

```
1  from rest_framework import serializers
2  from rest.models import Articles
3
4  class ArticlesSerializer(serializers.HyperlinkedModelSerializer):
5      class Meta:
6          model = Articles
7          fields = ('url', 'id_article' ,'author', 'category', 'title', 'content', 'timestamp')
```

La variable "fields" délimitera les champs à sélectionner. On notera plus tard que des champs seront cachés comme le statut ou bien encore le nombre de vues de l'article.

8.5.4 Modèle : models.py

Le modèle représente la structure de la table. Lié à une base de données, le modèle doit permettre à Django d'interpréter la structure des ressources.

Code source : models.py

```
1  from django.db import models
2
3  class Articles(models.Model):
4      id_article = models.AutoField(primary_key=True)
5      author = models.TextField(blank=True)
6      category = models.TextField(blank=True)
7      title = models.TextField(blank=True)
8      content = models.TextField(blank=True)
9      timestamp = models.DateField(format='%d/%m/%Y', input_formats='%d/%m/%Y')
10     status = models.IntegerField(blank=True, null=True)
11     count_views = models.IntegerField(blank=True, null=True)
12
13     class Meta:
14         managed = False
15         db_table = 'blog_article'
```

Au contraire du fichier de sérialisations, tous les champs doivent être présents avec le type défini dans la base de données.

8.5.5 Utilisation du framework

Via l'URI : **http://api.yourcompany.com/articles/?format=json**, Django retournera, sous forme de données JSON, la liste de tous les articles provenant de la base de données. Pour limiter la recherche, le framework intègre la fonctionnalité de lire élément par élément. Identifier par la clef primaire, l'ajout de l'identifiant dans l'URI permet d'isoler l'objet de la liste d'articles.

http://api.yourcompany.com/articles/511/?format=json retournera par exemple :

```
1  {"url":"http://api.yourcompany.com/articles/511/","id_article":511,"author":"Rosati", [...]}
```

8.6 Conclusion

L'utilisation du framework Django Rest n'est que bénéfique. Utilisée au courant du stage, sa facilité d'adaptation a été stupéfiante. Le chapitre ne démontre pas la puissance que le framework propose. Il est possible d'intégrer ses propres fonctions aux modèles déjà existants par exemple.

Ce qu'il faut retenir c'est que **Django Rest Framework** permet la mise en place de données dites fermées depuis une application Web ce qui rend les données ouvertes.

Chapitre 9

AngularJS

9.1 Introduction

AngularJS est un framework JavaScript porté par Google. Devenu la référence des frameworks JavaScript, AngularJS est accompagné de Node.js en ce qui concerne la partie serveur. À rappeler que le langage JavaScript se positionne du côté client, aucune information n'est tenue sur le serveur pour chaque utilisateur.

Son infrastructure se base sur le modèle MVVM, c'est à dire Modèle Vue Vue-Modèle, qui a été conçu pour le développement de site web. La base du MVVM est un travail en cascade en temps réel. Pour être plus simple, lors de la saisie côté client, la vue appel à la mise à jour du contrôleur qui met à jour lui-même cette vue.

9.2 Comparaison avec Jquery

Jquery est une librairie, ce qui veut dire que votre code appelle des fonctions de la librairie pour produire un résultat.
AngularJS, lui, est un framework basé sur l'architecture MVVM imposant sa philosophie. Vous ne pouvez pas marier un projet AngularJS dans un projet utilisant Jquery, mais l'inverse lui est possible.

Ci-joint, un tableau comparatif entre les fonctionnalités qu'offre jQuery et AngularJS.

	jQuery	AngularJS
Abstracts the DOM	✓	✓
Unit Test Runner	✓	✓
Deferred Promises	✓	✓
Cross-Module Communication	✓	✓
Animation Support	✓	✓
AJAX / JSONP	✓	✓
RESTful API	✗	✓
Integration Test Runner	✗	✓
MVC Pattern Support	✗	✓
Templating	✗	✓
Two-way Data Binding	✗	✓
Dependency Management	✗	✓
Deep-Link Routing	✗	✓
Form Validation	✗	✓
Localization	✗	✓
File Size	32KB	38KB

9.3 La philosophie AngularJS

9.3.1 Les directives

Les directives sont une propriété d'AngularJS qui permet d'injecter directement des comportements dans le code HTML. On les compare généralement aux attributs des éléments HTML, mais les propriétés venant du framework ont plus un impact fonctionnel que descriptif.

Il existe des dizaines de directives, mais on restera sur les principaux et les plus utilisés.

Pour commencer il existe le **ng-model**. Cette directive a pour but de lier l'élément avec l'environnement AngularJS. Il est utilisé pour les éléments tels que **input**, **textarea** & **select** qui permettront par la suite d'accéder aux valeurs entrées côté client depuis le contrôleur et même sa propre vue.
exemple d'utilisation

```
1  <body ng-app>
2  <form>
3  <label>Votre prenom:
4  <input type="text" name="prenom"
5  ng-model="user" />
6  </label>
7  </form>
8  Bonjour "{{user}}" !
9  <script src="https://ajax.googleapis.com/[...]/angular.min.js"></script>
10 </body>
```

résultat

Votre prenom: Alexandre
Bonjour "Alexandre" !

Lors de l'insertion dans l'élément "prenom", la vue se chargera elle même de mettre à jour la valeur en ligne 8. Aucun rafraîchissement ne sera nécessaire à contraire d'un POST en PHP et on notera qu'aucun code JavaScript ne sera nécessaire.

Dans le même style de directives, le second le plus utilisé est **ng-repeat**. Accompagnée des filtres, cette directive est utilisée pour la création de liste dynamique et autogérée.

exemple d'utilisation

```
<div ng-init="amis = [
{name:'Anthony', age:19, location:'Houdeng'},
{name:'Etienne', age:22, location:'Mons'},
{name:'Nicodemo', age:23, location:'Manage'}
]">
J'ai {{amis.length}} amis. Ils sont :
<input type="search" ng-model="q" placeholder="Name, age or location" aria-label="filter friends" /
<ul>
<li ng-repeat="ami in amis | filter:q as count">
[{{$index + 1}}] {{ami.name}} est agee de {{ami.age}} ans et viens de {{ami.location}}.
</li>
<li ng-if="count == 0">Aucun ami trouvee !</li>
</ul>
</div>
<script src="https://ajax.googleapis.com/[...]/angular.min.js"></script>
```

résultat 1

J'ai 3 amis. Ils sont : Name, age or location

- [1] Anthony est agee de 19 ans et viens de Houdeng.
- [2] Etienne est agee de 22 ans et viens de Mons.
- [3] Nicodemo est agee de 23 ans et viens de Manage.

résultat 2

J'ai 3 amis. Ils sont : an ×

- [1] Anthony est agee de 19 ans et viens de Houdeng.
- [2] Nicodemo est agee de 23 ans et viens de Manage.

résultat 3

J'ai 3 amis. Ils sont : Marie ×

- Aucun ami trouvee !

Dans ces exemples nous pouvons remarquer l'apparition de nouveaux éléments.

- **ng-init** permet d'initialiser des variables, en l'occurrence une liste d'objet JSON qui sera accessible dans cette vue et du contrôleur si présent.
- **ng-repeat** l'Élément actuel se répétera x fois en rapport au nombre d'éléments dans la liste amis. On peut l'interpréter comme un "For Each" dans d'autre langage de programmation. On remarquera qu'un filtre est appliqué sur tous les éléments de chaque objet en question.
- **ng-if** offre la possibilité d'appliquer des conditions pour l'affichage. Dans l'exemple présent si le filtre ne trouve aucun résultat, la condition répond présente et donc affiche l'élément "li".

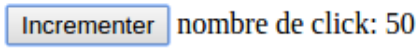
Ce que prouve ce code est que AngularJS est conçu pour développer des applications rapidement sans le moindre effort. En JavaScript, avec Jquery, nous aurions dû coder une dizaine de lignes pour avoir le même résultat alors qu'il n'aura fallu que 2-3 Directives avec ce framework.

Pour finir il reste une 3éme Directive utilisée dans chaque projet, il se nomme **ng-click**. Fortement utilisé pour l'interaction entre la vue et le contrôleur, **ng-click** permet aussi d'interagir dans sa propre vue sans passer par un tiers.

exemple d'utilisation

```
1  <button ng−click="count = count + 1" ng−init="count=0">
2  Incrementer
3  </button>
4  <span>
5  nombre de click: {{count}}
6  </span>
```

résultat



9.3.2 Les modules

AngulaJS permet d'organiser les fonctionnalités et les différentes libraires sous forme de module. En général les modules permettent d'isoler chaque partie de votre application sous forme de section réutilisable.

Par défaut AngularJS appelle les modules **ng** et **ngLocal** pour les fonctionnalités de base comme la création de vos propres modules et l'utilisation des directives comme "ng-repeat".

Un module est une collection de service, de directives, de contrôleur, de filtre et de configuration. Il peut être comparé au package en Java qui regroupe les parties comme des os sur un squelette.

9.3.3 Les controllers et le scope

Un contrôleur est le squelette d'une vue. Dans le modèle de AngularJS(MVVM), le contrôleur est caractérisé par Vue-Modèle. Il permettra d'interagir avec les données venant de l'extérieur comme une API REST, mais aussi de manager la vue en répondant a certaine condition.
Pour expliquer son utilité, voici un exemple :

```
function monSuperController($scope){
$scope.amis = [
{name:'Anthony', age:19, location:'Houdeng'},
{name:'Etienne', age:22, location:'Mons'},
{name:'Nicodemo', age:23, location:'Manage'}
];
}
```

Pour créer un contrôleur il suffit de lui donner un nom comme une fonction normale. Par convention on nommera un contrôleur avec comme suffixe "Controller" ou "Ctrl" pour pouvoir les différencier des fonctions classique.

On remarquera qu'une variable est en paramètre du contrôleur: **$scope**. En AngularJS on appelle cela comme une injection de dépendance qui est propre a ce framework.

Pour comprendre le **$scope** dites vous qu'il permet de lier les variables entre votre vue & le contrôleur. Grâce a ce module, nous pouvons déplacer les valeurs décrites dans **ng-init** et de les ajouter au contrôleur sous forme de **$scope**. L'utilité principale est d'alléger le code HTML et de laisser au contrôleur la gestion de données.

exemple d'utilisation

index.html

```
<body ng-app="myApp">
<div ng-controller="myCtrl">
J'ai {{amis.length}} amis. Ils sont :
<input type="search" ng-model="q" placeholder="Name, age or location"
    aria-label="filter friends" />
<button ng-click="reset()">reset</button>
<ul>
<li ng-repeat="ami in amis | filter:q as count">
[{{$index + 1}}] {{ami.name}} est agee de {{ami.age}} ans et viens de {{ami.location}}.
</li>
<li ng-if="count == 0">Aucun ami trouvee !</li>
</ul>

</div>
<script src="https://ajax.googleapis.com/[...]/angular.min.js"></script>
<script>\begin{center}
\includegraphics[scale=0.7]{C:/Users/Alexandre/Desktop/tfe_alex/angular_ng-click.png}
\end{center}
<script src="ctrl.js"></script>
</body>
```

ctrl.js

```
1  var app = angular.module('myApp', []);
2  app.controller('myCtrl', function($scope) {
3  $scope.amis = [
4  {name:'Anthony', age:19, location:'Houdeng'},
5  {name:'Etienne', age:22, location:'Mons'},
6  {name:'Nicodemo', age:23, location:'Manage'}
7  ];
8  $scope.reset = function(){
9  $scope.q = "";
10 }
11 });
```

Dans cet exemple, nous pouvons remarquer l'apparition d'un module **myApp** qui contiendra un contrôleur : **myCtrl**. Grâce à cette nouveauté, le **ng-init** d'auparavant se verra disparaître pour laisser place à un scope dans le contrôleur.

La fonction **$scope.reset** sera appelée lors d'une interaction sur le bouton en ligne 5 de **index.html** qui permettra la remise à défaut du filtre.

exemple de contrôleur

```
1  app.controller('favCtrl', function($scope, $rootScope, $state, localStorageService){
2  $rootScope.$on('$locationChangeSuccess', function () {
3  if($rootScope.favRefresh == 1){
4  $rootScope.favRefresh = 0;
5  loadData();
6  }
7  });
8  function loadData(){
9  $scope.objEmpty = false;
10 var obj = localStorageService.get('fav');
11 $scope.data = obj;
12 if ($scope.data.length == 0) $scope.objEmpty = true;
13 }
14 loadData();
15 $scope.goDetail = function(item){
16 var id = item.id;
17 var tmp = id.split('_');
18 if(tmp[0] == "ACCESSI"){
19 $state.go('app.detail', {'id': tmp[1]});
20 }else{
21 $state.go('app.detail2', {'id': tmp[1]});
22 }
23 }
24 })
```

9.3.4 Les routes

Le routage gère la transition entre toutes vos pages et permet de savoir quelles parties doivent être chargées au moment voulu. AngularJS propose un module prêt à l'emploi qui se nomme : **ngRoute**.
Le module **ngRoute** regroupe plusieurs éléments permettant la mise en place de votre routage :

2 services
- **$route** est le service central du module. Il lit l'URL demandée par le navigateur et la parse pour pouvoir la traiter ensuite en fonction des informations données au provider.
- **$routeParams** permet de gérer les paramètres et les arguments passés dans l'url.

1 provider
- **$routeProvider** est le fournisseur du module. Autrement dit, il initialise le routage grâce aux règles qu'on lui donne lors de la configuration de notre application.

1 directive
- **ngView** sert à insérer le template de votre page à l'intérieur de votre layout général.

exemple d'utilisation

```
1  routeApp.config(['$routeProvider',
2  function($routeProvider) {
3  $routeProvider
4  .when('/accueil', {
5  templateUrl: 'pages/accueil.html',
6  controller: 'acceuilCtrl'
7  })
8  .when('/membre/:id', {
9  templateUrl: 'pages/membre.html',
10 controller: 'membreCtrl'
11 })
12 .when('/contact', {
13 templateUrl: 'pages/contact.html',
14 controller: 'contactCtrl'
15 })
16 .otherwise({
17 redirectTo: '/accueil'
18 });
19 }
20 ]);
```

Cet exemple décrit la structure d'un fichier de routage. Il est possible de passer des données dans l'URL, mais elles doivent être déclarées dans ce fichier pour être utilisées via le contrôleur.

À note qu'aucune erreur 404 ne sera générée, car la route prend en compte l'exception qu'il peut y avoir une erreur et redirige l'utilisateur vers l'accueil.

9.3.5 Les services

Les services sont la partie “modèle” du framework AngularJS, c’est ce qui va permettre de récupérer des données depuis différentes sources et de les envoyer vers les contrôleurs qui eux enverront un résultat à la vue.

Il existe cinq méthodes regroupées en 3 groupes qui permettent de créer un service: provider, factory, service, value et constant.

1. Le **provider** est la méthode la plus configurable pour créer un service, il est le seul service a pouvoir inclure des paramètres lors de son initialisation.

2. La **factory**, **service** et **value** sont eux des méthodes qui délèguent au **provider**. Elles sont des raccourcis pour le **provider** mais sont développées d’une tout autre manière.

3. Le service **constant** se comporte de manière plus spécifique puisque chaque service constant est son propre **provider**.

Pour généraliser le tout, les services **constant** et **value** sont utilisés pour contenir des valeurs ou des objets tandis que **factory** et **service** sont des raccourcis pour la création de **provider** qui permettent de créer des services sans se soucier de leur configuration.

Exemple de provider

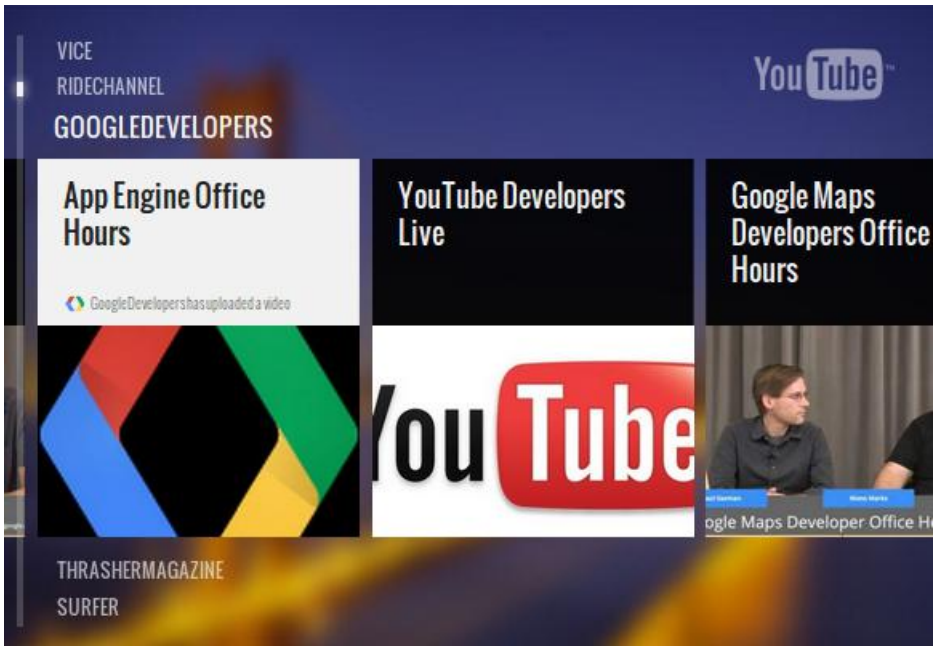
```
app.provider("profil", function(){
  this.$get = function() {
    return {
      "id": null,
      "name": "Anonyme",
      "login": function(){
        // fonction de connexion
      },
      "register": function(){
        //fonction d'inscription
      },
      "logout": function(){
        // fonction de deconnexion
      }
    }
  }
});
```

9.4 Retour sur AngularJS

Au contraire d'un développement dit "basique", ce framework propose beaucoup d'aide et facilite le déploiement d'une application en très peu de temps. Il est fortement utilisé pour des applications orientées vitrine.

Le seul inconvénient est que l'application soit du côté client. Le framework n'offre pas de fonctionnalité comme les variables de session en PHP. Il faudra développer des modules afin de protéger les données sensibles comme des identifiants ou des accès d'administration.

Exemple d'application en AngularJS



Sony, producteur des consoles PlayStation, a développé l'application YouTube en se basant sur le framework AngularJS. Ce framework est parfaitement adapté à ce type d'application, car il ne gère que l'affichage de données depuis des serveurs.

Chapitre 10

IONIC Framework

10.1 Introduction

IONIC est un framework qui a pour but de développer rapidement et facilement une application mobile hybride. Le terme hybride est pour décrire que l'application, développée dans un seul langage, sera déployée sur les différents supports comme Android, IOS ou WindowsPhone.

Basé sur l'architecture AngularJS pour le front-end, Ionic permet de développer des prototypes assez rapidement grâce aux nombreux outils que AngularJS propose, mais aussi a son framework CSS qui offre beaucoup d'élément préutilisable.

Du côté de la préparation mobile, Cordova gère la construction des applications native depuis le code source HTML et JavaScript.

10.2 Cordonva

Apache Cordova est un ensemble d'API mobile qui permettent au développeur d'une application d'accéder aux fonctions natives du périphérique comme la caméra, mais aussi au vibreur depuis le code JavaScript.

Grâce à Cordova, une application peut être construite sans code natif(Java, Objective-C, etc.) par le développeur de l'application. Au lieu de cela, les technologies web sont employées et hébergées localement dans l'application.

Cordova est disponible pour les plates-formes suivantes: iOS, Android, BlackBerry, Windows Phone, Palm WebOS, Bada et Symbian.

10.3 Les atouts de Ionic

L'une des raisons principales vient que Ionic propose beaucoup d'outils prêts à l'emploi. Dans un centre de recherche, le but recherché est de déployer un prototype sans prendre en compte l'aspect esthétique en priorité, l'application doit être fonctionnelle et facile d'utilisation.

10.3.1 Composant CSS

Dans le développement web, les frameworks CSS aident les développeurs à aménager leurs interfaces grâce à de nombreux code HTML et CSS prêt à l'usage, mais aussi à des philosophies de gestion d'espace visuel.

L'un des atouts d'utiliser les composants proposés par Ionic est son système de grille(**grid**). Ce système permet de rendre responsive les interfaces avec toutes les résolutions d'écran possible.

exemple d'utilisation

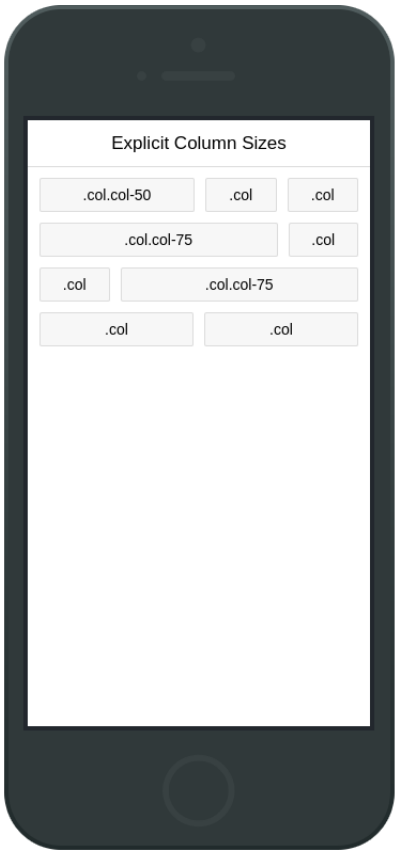
```
<div class="row">
<div class="col col-50">.col.col-50</div>
<div class="col">.col</div>
<div class="col">.col</div>
</div>

<div class="row">
<div class="col col-75">.col.col-75</div>
<div class="col">.col</div>
</div>

<div class="row">
<div class="col">.col</div>
<div class="col col-75">.col.col-75</div>
</div>

<div class="row">
<div class="col">.col</div>
<div class="col">.col</div>
</div>
```

résultat



Un autre exemple de composant est la **Cards**. Celui-ci permet d'afficher des données sous forme de billet avec la possibilité d'avoir un menu d'action en bas de contenu.

exemple d'utilisation

```
<div class="list card">

<div class="item item-avatar">
<img src="mcfly.jpg">
<h2>Marty McFly</h2>
<p>November 05, 1955</p>
</div>

<div class="item item-body">
<img class="full-image" src="delorean.jpg">
<p>
This is a "Facebook" styled Card. The header is created from a Thumbnail List item,
the content is from a card-body consisting of an image and paragraph text. The footer
consists of tabs, icons aligned left, within the card-footer.
</p>
<p>
<a href="#" class="subdued">1 Like</a>
<a href="#" class="subdued">5 Comments</a>
</p>
</div>
```

```
21
22  <div class="item tabs tabs-secondary tabs-icon-left">
23  <a class="tab-item" href="#">
24  <i class="icon ion-thumbsup"></i>
25  Like
26  </a>
27  <a class="tab-item" href="#">
28  <i class="icon ion-chatbox"></i>
29  Comment
30  </a>
31  <a class="tab-item" href="#">
32  <i class="icon ion-share"></i>
33  Share
34  </a>
35  </div>
36
37  </div>
```

résultat

10.3.2 Les plugins Cordova

Comme annoncé plus haut, Ionic est lié à Cordova. Il est possible d'installer des plug-ins depuis Cordova et de les utiliser en les injectant dans les modules Ionic. Sur le site officiel de Cordova, il existe une multitude de plug-ins en passant par la gestion de la caméra aux vibrations de l'appareil.

Exemple d'utilisation

puglins : cordova-plugin-geolocation

```
1  module.controller('GeoCtrl', function($cordovaGeolocation) {
2  var posOptions = {timeout: 10000, enableHighAccuracy: false};
3  $cordovaGeolocation
4  .getCurrentPosition(posOptions)
5  .then(function (position) {
6  var lat = position.coords.latitude
7  var long = position.coords.longitude
8  }, function(err) {
9  // error
10 });
11 });
```

Injecter dans les paramètres du contrôleur, **$cordovaGeolocation**, permet de retourner la position actuelle grâce au récepteur GPS vers le contrôleur.

puglins : cordova-plugin-statusbar

```
1  module.controller('MyCtrl', function($cordovaStatusbar) {
2  $cordovaStatusbar.overlaysWebView(true);
3
4  // styles: Default : 0, LightContent: 1, BlackTranslucent: 2, BlackOpaque: 3
5  $cordovaStatusbar.style(1);
6
7  // supported names: black, darkGray, lightGray, white, gray, red, green,
8  // blue, cyan, yellow, magenta, orange, purple, brown
9  $cordovaStatusbar.styleColor('black');
10
11 $cordovaStatusbar.styleHex('#000');
12
13 $cordovaStatusbar.hide();
14
15 $cordovaStatusbar.show();
16
17 var isVisible = $cordovaStatusbar.isVisible();
18
19 });
```

Grâce a ce plug-in, il sera donc possible de changer le rendu de la bar d'état à travers des méthodes proposé par **$cordovaStatusbar**.

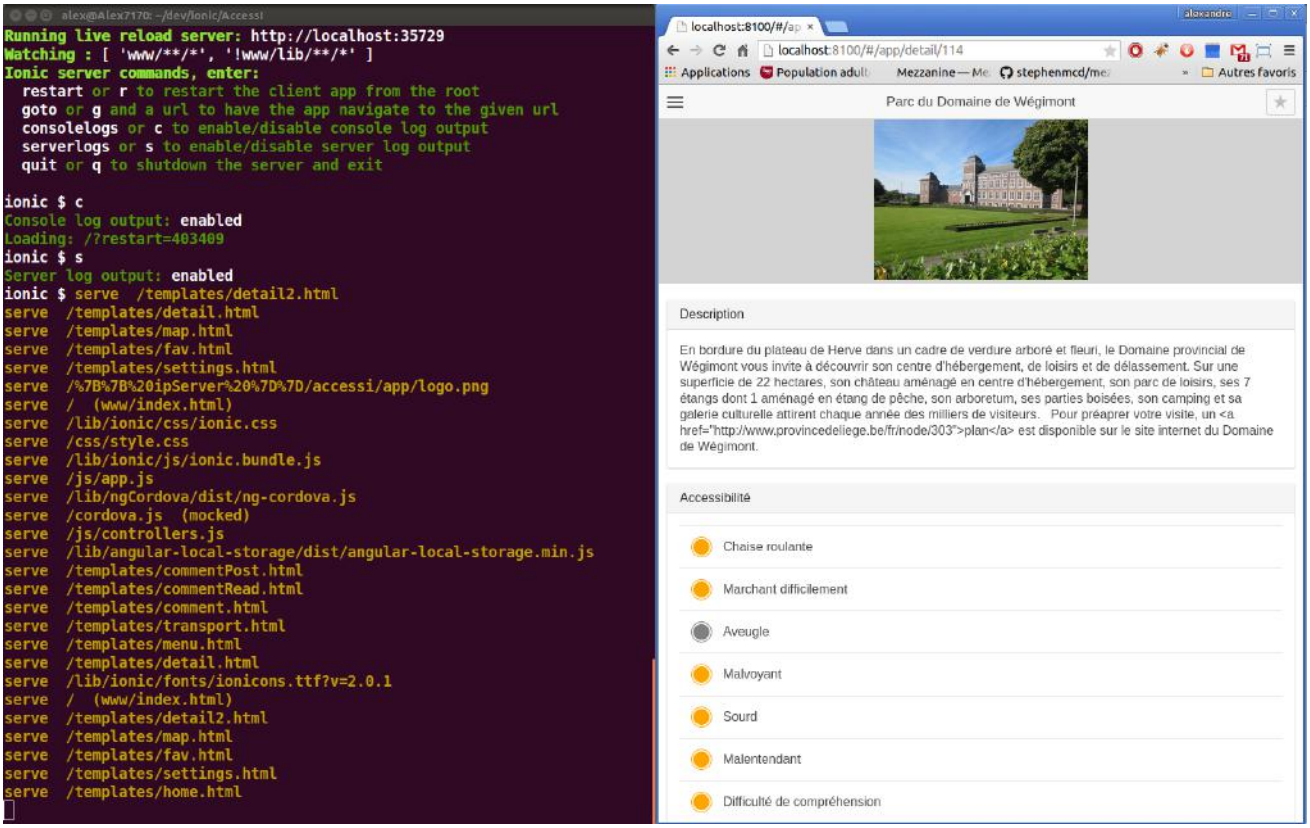
10.3.3 Le debug

Ionic propose beaucoup d'outils, l'un en particulier est de permettre la simulation de l'application dans le navigateur web. Il offre aussi la possibilité d'avoir un retour de log dans la console et indique les endroits où le projet n'atteint pas ses objectifs.

Utilisation de la commande

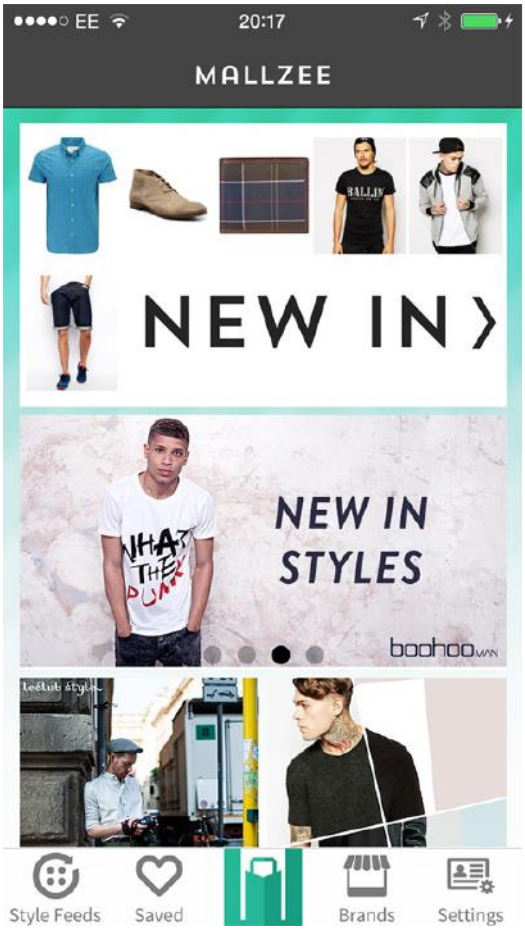
```
1 ionic serve
```

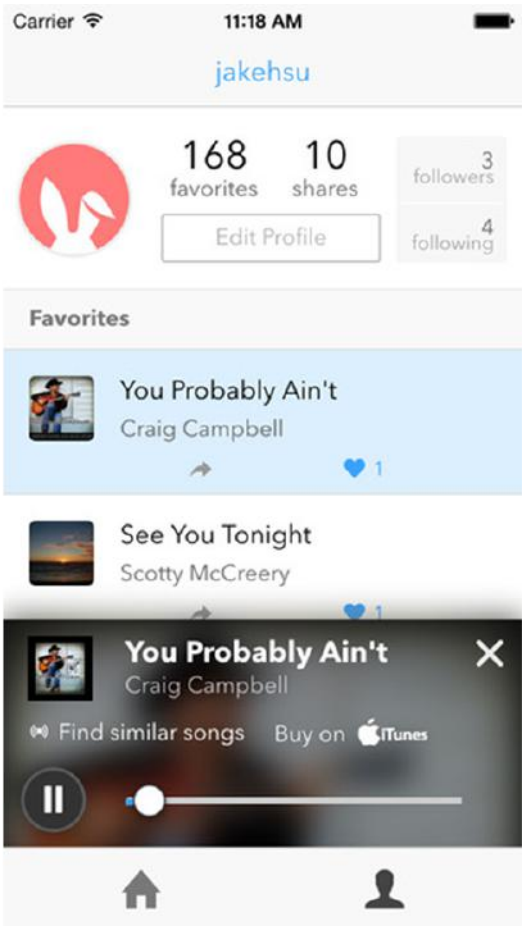
Cette commande, exécutée à la racine du projet, permet de rendre disponible la dernière version de l'application depuis un navigateur Web.



L'inconvénient de cette fonctionnalité est qu'il est impossible de tester les données comme l'identifiant de l'appareil (numéro de série) ou encore l'appareil photo.

10.4 Example d'application Ionic





Chapitre 11

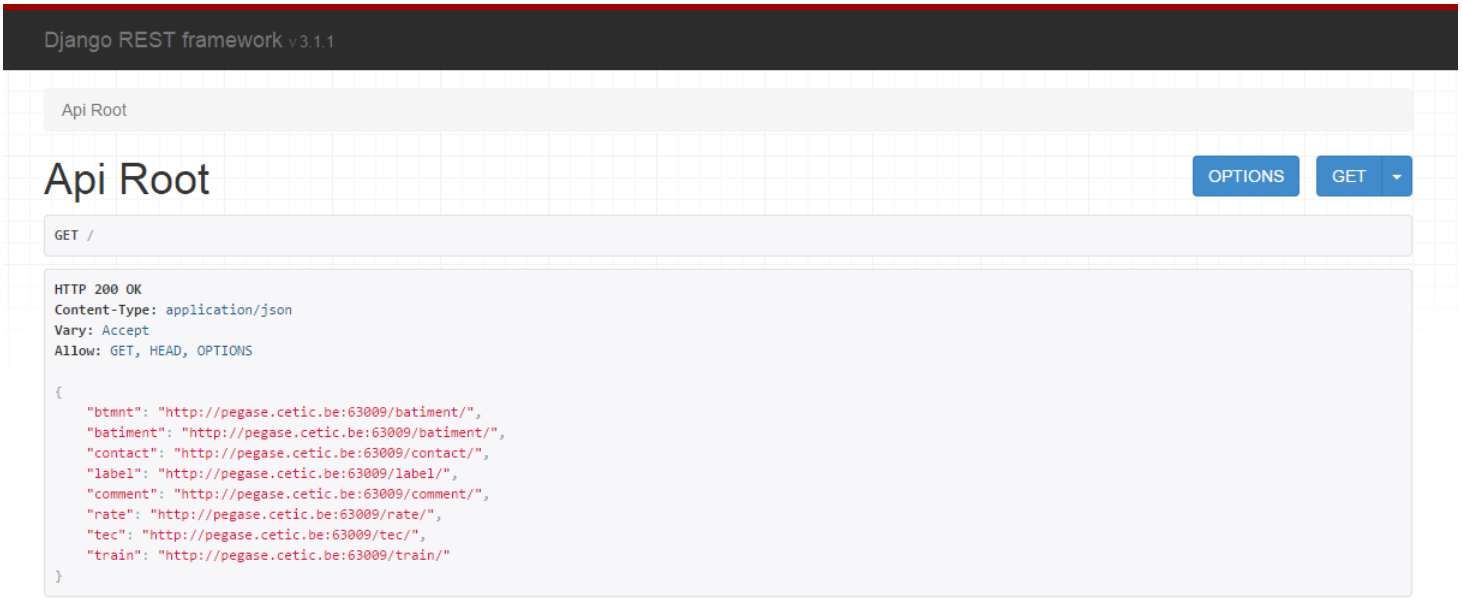
Projet Final

Après 12 semaines de recherche et développent, le projet AccessI-Mobile a vu le jour. Développé sous le framework Ionic pour le développement front-end, une API Rest Django était en back-end pour la gestion des données de access-i.be ainsi que les données de mobilité.

Lors de la recherche des jeux de données, j'ai pu joindre le département informatique de jaccede.com afin d'accéder à leurs ressources. Après plusieurs mails interposés, jaccede.com nous a invités à tester leur API et j'ai pu donc globaliser leurs données avec celles déjà présentes.

11.1 Back-end

Pour la publication de données, j'ai mis en place une application Rest portée par Django.



Au travers de cette application, le front-end pouvait recevoir les données des bâtiments ainsi que les moyens de transport et les commentaires au format JSON.

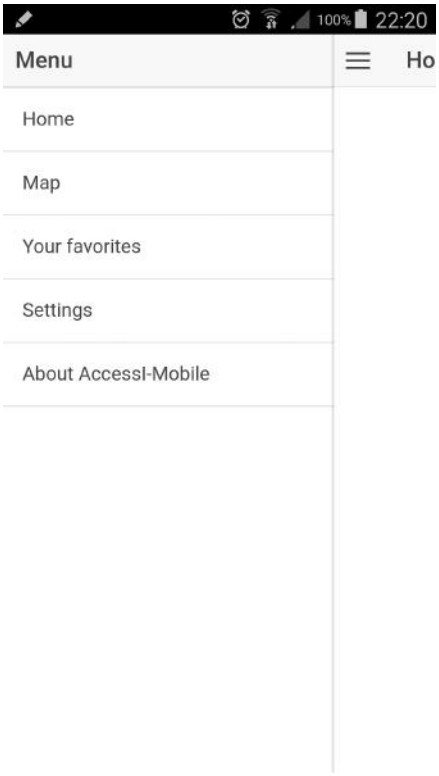
Dans le cadre du prototype, j'ai rendu l'application publique, mais il est possible de la rendre privé via des tokens d'authentification.

11.2 Front-end

Du côté client, l'application s'est basée sur le framework Ionic. Son avantage a été l'utilisation de AngularJS pour la liaison avec l'API Django Rest grâce à ces fonctions de lecture(GET) et d'envoi de données(POST).

De même, l'utilisation des composants CSS m'a fortement aidé pour la réalisation des interfaces. N'étant pas créatif dans l'âme, les éléments proposés ont permis d'avoir un prototype clair et rapide.

11.2.1 Module : Menu

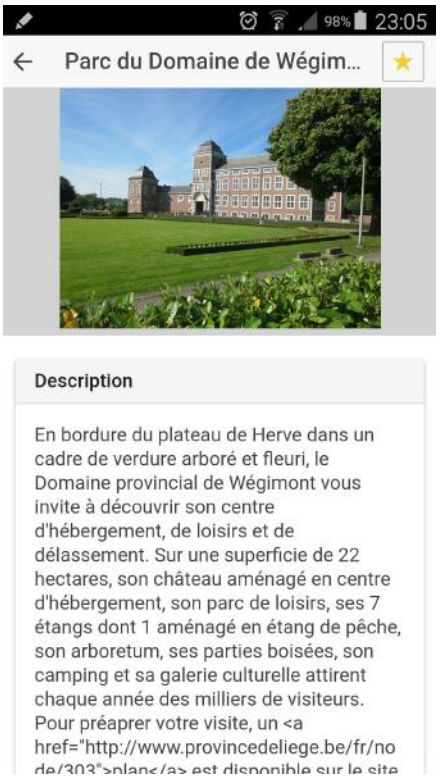


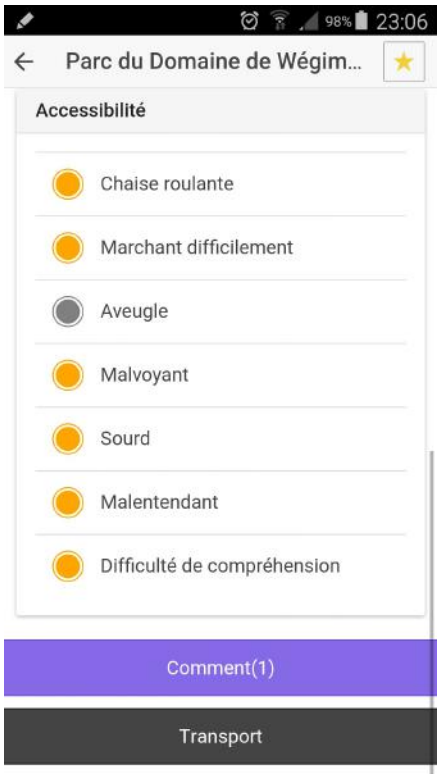
Le menu donne accès aux différentes fonctionnalités de l'application. Communément appelé **side menu**, ce composant est disponible depuis chaque interface en cliquant sur l'icône du header.

11.2.2 Module : Map

La map se trouve au coeur de l'application. Elle regroupe tous les bâtiments venant de chez access-i.be et de jaccede.com. À noter que les données venant de ces deux sources sont différentes et possèdent chacune sa structure de données. J'ai utilisé un code couleur pour pouvoir les identifier.

11.2.3 Module : Detail AccessI





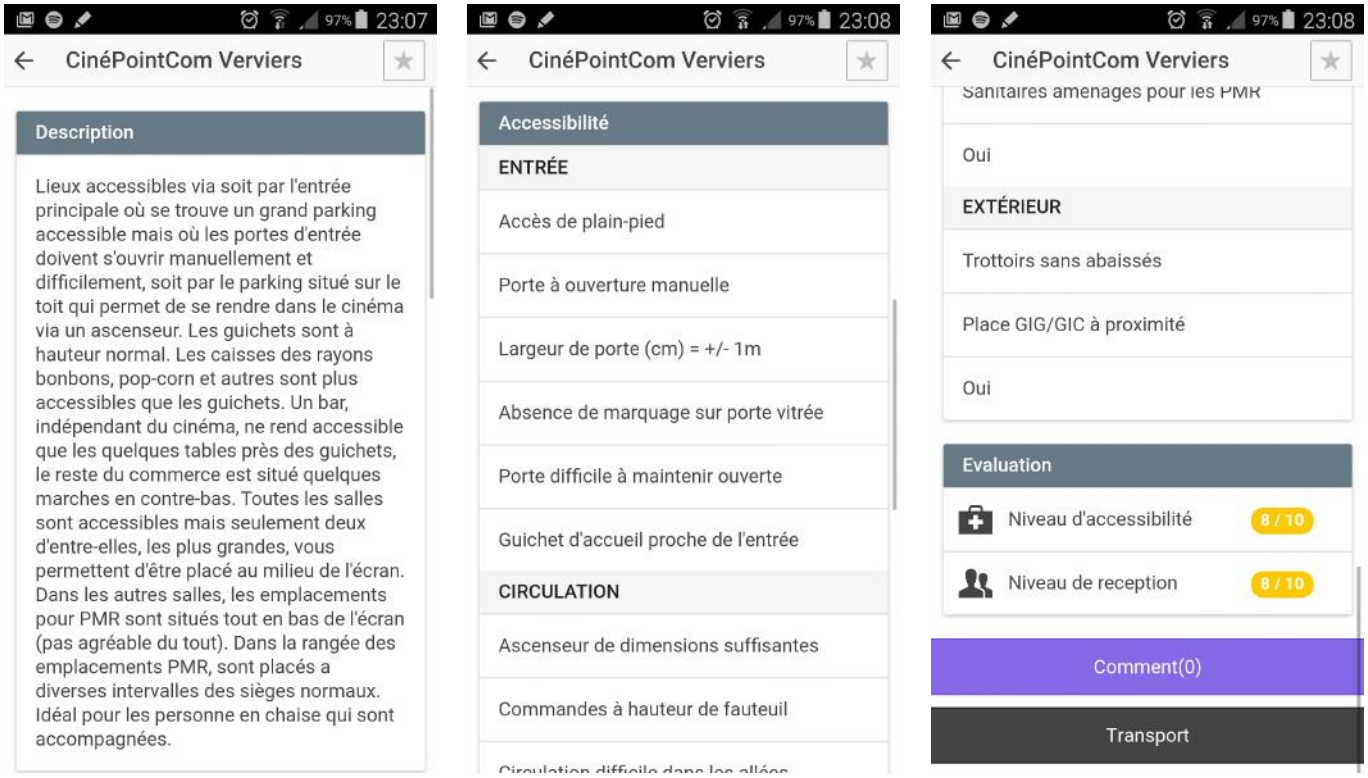
Une galerie d'images défile via un slider en début de fiche en rapport au bâtiment de celle-ci.

AccessI différencie ses handicaps par des codes couleur, représentés par le vert, orange et gris. Il est possible d'avoir accès au point fort et point faible individuellement via une interaction avec l'un des handicaps listés. Un popup s'affichera afin d'avoir les données correspondantes.

Trois options sont possibles via cette interface:

- **favoris** : possible d'ajouter ou de retirer cette fiche dans votre liste de suivis
- **comment** : Accède au commentaire lié à cette fiche
- **transport** : Affiche tous les arrêts de bus et de train à proximité de ce bâtiment

11.2.4 Module : Detail Jaccede

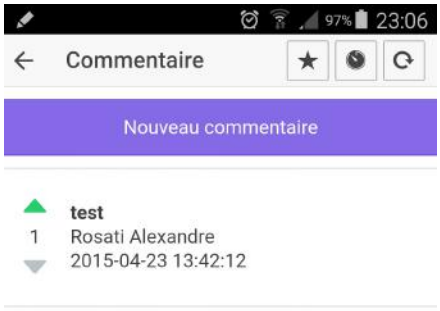


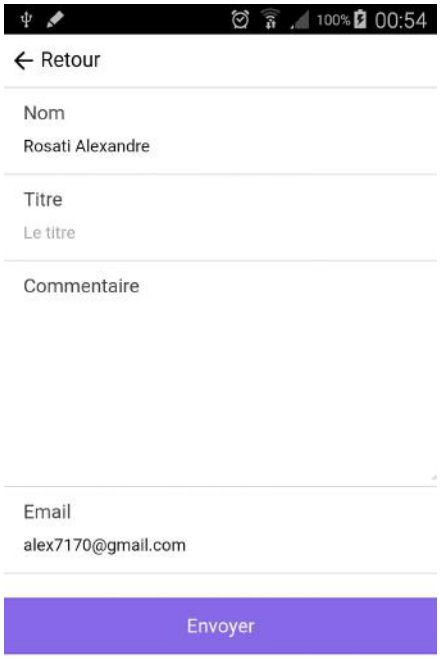
Ressemblant aux fiches d'AccessI, celle-ci est organisée différemment.

La grande différence est les informations sur l'accessibilité. Jaccede ne fait pas la différence pour chaque handicap et englobe les données. De même, l'évaluation reste générale, une personne à mobilité réduite devra elle même interpréter son niveau d'accessibilité en lisant toute la fiche.

Pour les options disponibles sur cette fiche, elles sont identiques à celles d'AccessI.

11.2.5 Module : Commentaire



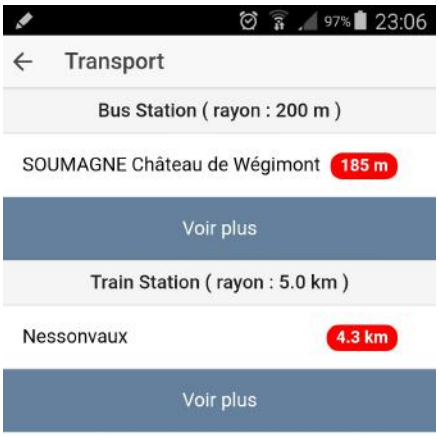


Chaque fiche a son ensemble de commentaires. Le but recherché étant d'aider à améliorer l'information grâce au crowdsourcing. Chaque commentaire peut être évalué par les autres utilisateurs afin que le rédacteur de cette fiche puisse prendre conscience que celui-ci est pertinent.

Lors d'un clic sur un commentaire, l'application retourne une interface plus détaillée avec le texte correspondant à ce commentaire.

Lors de la rédaction d'un commentaire, celui-ci sera envoyé à l'application REST en POST et sera donc stocké dans la base de données.

11.2.6 Module : Transport

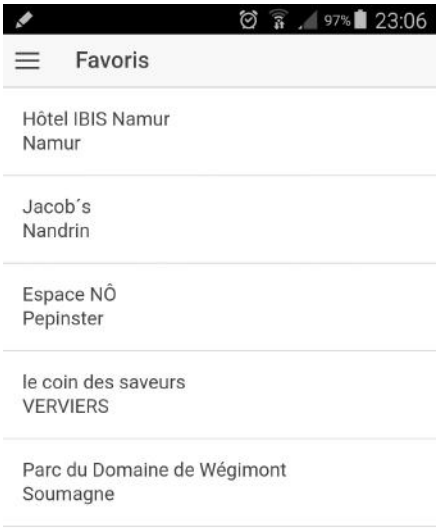


La mobilité est vraiment un problème pour les personnes à mobilité réduite. Pour faciliter l'information, les noms des rues possédant un arrêt de bus ou une gare sont affichés par rayon géographique.

Il est possible d'étendre la recherche géographique en cliquant sur les deux boutons : Voir plus.

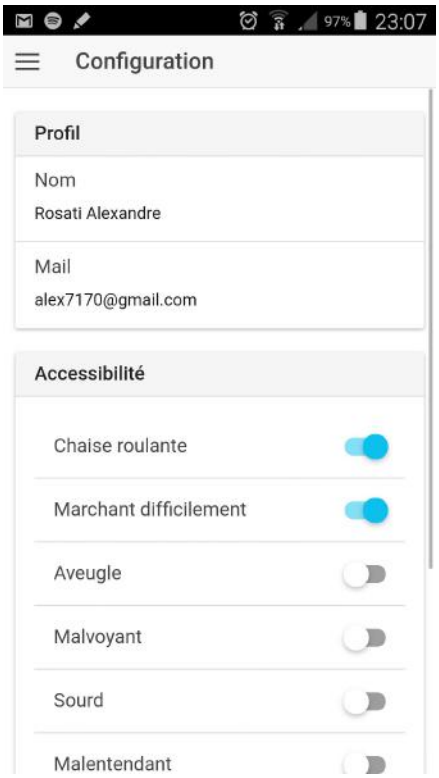
Par manque de temps, l'idée pour compléter cette fonctionnalité était d'afficher le POI sur une carte et d'y inclure un tracé de déplacement entre le bâtiment et le moyen de transport.

11.2.7 Module : Favoris



Pour faciliter la navigation, un système de suivis de fiche est disponible en cas de lecture reportée par son utilisateur.

11.2.8 Module : Configuration



L'avantage d'une configuration comme celle-ci est d'éviter d'écrire l'identifiant et l'adresse mail de l'utilisateur a chaque rédaction de commentaire.

Par ailleurs, pour chaque fiche provenant d'une source AccessI, il sera possible d'afficher les handicaps en rapport à l'utilisateur. Ceci allégera l'affichage de la liste des handicaps.

11.3 Retour Gamah et ANLH

Un retour a été collecté lors de la dernière réunion avec les associations qui a eu lieu mi-mai. Dans l'ensemble, le prototype a beaucoup plu à Gamah et l'ANLH. Connaissant les problèmes que pouvez engendrer le site access-i.be sur les mobiles, les partenaires ont été satisfaits du résultat.

Néanmoins, quelques problèmes et suggestions ont été relevés.

Pour les personnes malvoyante ou aveugle: l'application devrait respecter les standards d'accessibilité numérique, notamment en respectant un label tel que AnySurfer. Vu le public PMR, ses adaptations sont bien sûr indispensable si on envisage la diffusion de l'application mais sortaient de la portée du travail.

Ensuite, le système de commentaire devrait être focalisé sur chaque handicap individuellement. Actuellement, un commentaire est général, c'est à dire sur l'ensemble de la fiche. La parade serait de pouvoir commenter chaque handicap individuellement dans le but de regrouper les avis pour une meilleure lisibilité.

Pour conclure, l'application dans l'état actuel est assez prometteuse. À rappeler qu'il ne s'agit que d'un prototype donc des modifications doivent être obligatoirement faites avant un déploiement sur le marché public. Les associations ont été très intéressées à la fois par la méthode de consolidation (dans un cadre de certification des données d'accessibilité) et par l'application mobile (pour le niveau diffusion qu'elle rend possible).

11.4 Retour personnel

Globalement, peu de difficultés ont été présentes lors de la mise en place du prototype.

Cependant, un problème a freiné le développement. Dans le cas de jaccede.com, la structure des données n'étant pas la même que pour AccessI. J'ai dû développer un script permettant de réorganiser la structure. Après ça, les données provenant de jaccede.com étaient bien plus lisibles qu'auparavant.

Au niveau de la technologie, utiliser Ionic au lieu d'un développement natif Android m'a permis de déployer rapidement et avec une grande légèreté ce prototype. Ce que j'ai fait via ce framework m'aurait sans doute pris deux fois plus de temps en Android. Sans compter que le prototype est utilisable sur Android, IOS et Windows Phone directement.

Pour finir, je suis assez content du résultat. La bémol est que je ne me suis pas assez mis a la place d'une personne malvoyante donc inutilisable pour ce handicap.

Chapitre 12

Conclusion

Durant ces 15 semaines de stage au CETIC, j'ai pu mettre en place un environnement de données ouvertes autour de l'accessibilité en Wallonie et comprendre le monde du travail.

Au début de mon stage, j'ai consacré plusieurs jours à la recherche de données déjà existantes et le constat qui s'est répété est que les organismes, sociétés et services publics ne partagent pas ouvertement leurs données pouvant être bénéfiques aux personnes à mobilité réduite.

Un autre constat est que certains organismes veulent bien faire en partageant leurs données, mais ne pense pas à leurs utilisateurs. Quelques-uns présenteront les données sous forme de fichier classeur (Excel) d'autres sous forme d'application Web plus professionnelle.

L'utilisation des ETL m'ont permis de gagner un temps considérable par rapport à un développement plus manuel. Le fait d'extraire des données pour les transformer et les envoyer dans une base de données sans développer une seule ligne de code m'a vraiment convaincu de l'importance de ce type d'outil, en l'occurrence dans ce travail : "mETL".

À ce niveau les données sont encore dites fermées, stockées dans une base de données PgSQL, il a fallu les rendre disponibles depuis l'extérieur via une application Web.

Django REST Framework a été mon choix après avoir lu beaucoup de retour positif sur son utilité et sa liberté d'utilisation. Là encore, son déploiement a été légèrement problématique. Développé en Python, l'apprentissage de cette langue a quand même été rapide et très instructif grâce aux aides que j'ai pu bénéficier au sein de l'équipe du CETIC.

L'apprentissage du langage Python m'a permis d'être plus productif et plus rapide. La communauté qui s'est créée autour de ce language est tellement riche et social. Celle-ci propose une multitude de librairies qui m'ont permis d'éviter de réinventer la roue lors de mes développements.

Par la suite, un développement mobile basé sur le framework Ionic a vu le jour. Avec les acquis des années précédentes, apprendre AngularJS, l'architecture de base d'un projet Ionic, a été vraiment plaisant et facile pour la gestion du prototype. Là encore très peu de problémes sont apparu vu que je possédais une bonne connaissance du language JavaScript.

En parallèle du stage j'ai pu participer avec une équipe du CETIC et un étudiant de l'université de Louvain la Neuve à un Hackathon sur le thème du tourisme. Le travail d'équipe et le management de projet ont été la clef de la réussite. Ils m'ont permis de savoir mettre des limites, que

ce soit temporel ou moral et de travailler parfaitement en coopération avec des personnes dont les compétences sont différentes des miennes.

Pour conclure, ce stage m'a permis d'apprendre beaucoup de choses encore inconnues pour un passionné comme moi. Je ne regrette en rien d'avoir pu travailler avec le CETIC et ses employés qui m'ont toujours aidé lorsqu'un obstacle était sur mon chemin. Le fait de travailler avec un groupe externe à l'entreprise comme Gamah et l'ANLH, a permis d'avoir une vision externe et plus générale de ce qu'un client recherche d'un développeur.

Chapitre 13

Bibliographie et sources documentaires

13.1 Ouvrages

- SWINNEN G., Apprendre à programmer avec Python 3, France, Eyrolles, 2010

13.2 Articles

- PONSARD C. et SNOECK V., "Unlocking Physical World Accessibility through ICT: A SWOT Analysis", 2014

13.3 Sites Web

- CETIC, site officiel du CETIC, sur http://cetic.be, consulté le 02/02/2015
- GAMAH, site officiel de l'asbl Gamah, sur http://www.gamah.be/, consulté le 11/05/2015
- ANLH, site officiel de l'asbl ANLH, sur http://www.anlh.be/, consulté le 11/05/2015
- ACCESS-I, portail d'information sur l'accessibilité, http://www.access-i.be/, consulté le 11/05/2015
- ANONYME, plateforme d'e-Education, http://openclassrooms.com/, consulté le 13/05/2015
- FALUDI B., mETL, sur https://github.com/ceumicrodata/mETL, consulté le 06/02/2015
- ANONYME, Open Data, sur http://en.wikipedia.org/wiki/Open_data, consulté le 02/03/2015
- ANONYME, 5 stars Open Data, sur http://5stardata.info/, consulté le 05/03/2015
- ANONYME, index sur les données ouvertes, sur http://index.okfn.org/place/, consulté le 05/05/2015
- SALUAN A., Vagrant et la virtualisation pour faciliter le développement, sur http://www.synbioz.com/, consulté le 12/05/2015
- ANONYME, framework REST, sur http://www.django-rest-framework.org/, consulté le 12/03/2015
- ANONYME, documentation AngularJS, sur https://docs.angularjs.org/guide, consulté le 14/04/2015
- ANONYME, documentation sur Ionic Framework, sur http://ionicframework.com/docs/, consulté le 14/04/2015