

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement 2016-2017

Routage multimodal des personnes

Tuteur académique

Ismâïla Abderhamane NDIAYE

Étudiant

Younes JAZOULI BENLAHBOUB (DI5)

3 avril 2017



Liste des intervenants

Nom	Email	Qualité
Younes JAZOULI BENLAHBOUB	younes.jazoulibenlahboub@etu.univ-tours.fr	Étudiant DI5
Ismâïla Abderhamane NDIAYE	Ismaila.NDIAYE@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Younes JAZOULI BENLAHBOUB susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Ismaïla Abderhamane NDIAYE susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Younes JAZOULI BENLAHBOUB, *Routage multimodal des personnes*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2016-2017.

```
@mastersthesis{
  author={JAZOULI BENLAHBOUB, Younes},
  title={Routage multimodal des personnes},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2016-2017}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	v
Liste des tableaux	vii
Introduction	1
I Partie 1 : Recherche et État de l’art	2
1 Contexte de la réalisation	3
1 Contexte	3
2 Objectifs	3
3 Bases méthodologiques	3
2 Description générale	4
1 Environnement du projet	4
2 Caractéristiques des utilisateurs	5
3 Fonctionnalités du système	5
4 Structure générale du système	6

3	État de l'art/veille	7
1	Calculateurs d'itinéraire.....	7
1.1	Open Trip Planner OTP	7
1.2	Prise en main d'OTP.....	9
1.2.1	Récupération du JAR	9
1.2.2	Récupération des données de transport.....	9
1.2.3	Démarrage d'OTP	9
1.3	NAVITIA	10
2	Fournisseurs des cartes.....	10
2.1	Open Street Map "OSM"	10
2.2	Google Maps API	11
2.3	Comparatif d'OSM et GoogleMAPS.....	11
3	Librairies JavaScript de carte interactive	11
3.1	LEAFLET.....	12
3.2	OPENLAYERS.....	12
3.3	GEOCOMPLETE.....	12
3.4	GMAPS.....	12
3.5	MAPLACE.....	12
3.6	Comparatif des différentes librairies	13
4	SPA Simple Page APP	13
4.1	Site web	14
4.2	SPA	14
5	FrameWork JS pour WebAPP	14
5.1	ANGULAR JS.....	14
5.2	REACT JS	15
5.3	EMBER JS.....	15
5.4	POLYMERJS	15
6	Comparatif des Licences	16
II	Partie 2 : Développement	17
4	Analyse et Mise en oeuvre	18
5	Phase de développement et d'intégration	26
1	Comprendre la structure de projet AngularJS.....	26
1.1	Pourquoi utiliser Angular JS 2	26
1.2	Environnement du projet	26
1.3	Prise en main du projet	28
	Conclusion	33

III	Annexe	34
6	Interfaces et Spécifications	35
1	Description des interfaces externes du logiciel	35
1.1	Interfaces matérielles/logiciel	35
1.2	Interfaces homme/machine	35
1.3	Interfaces logiciel/logiciel	36
2	Spécifications fonctionnelles	37
2.1	Définition de la fonction de recherche d'itinéraire	37
2.2	Définition de la fonction de récupération des données météo	37
2.3	Définition de la fonction d'affichage des détails d'un itinéraire	37
2.4	Définition de la fonction de Zoom sur la carte	37
2.5	Définition de la fonction d'auto complétion pour les champs de recherche des lieux de départ/arrivée	38
2.6	Définition de la fonction de sélection de la date et l'heure	38
2.7	Définition de la fonction de géolocalisation	38
3	Spécifications non fonctionnelles	38
3.1	Contraintes de développement et de conception	38
3.2	Contraintes de fonctionnement et d'exploitation	38
3.2.1	Performances	38
3.2.2	Capacités	39
3.2.3	Sécurité	39
3.2.4	Maintenance et évolution du système	39
7	Gestion de projet	40
1	Planning prévisionnel	40
2	Répartition des tâches	41
2.1	Documentation, lecture, compréhension	41
2.2	Outils liés au développement	42
2.3	Rédaction de l'état de l'art	42
2.4	Rédaction du CDS	42
2.5	Développement	42
2.5.1	Génération des données GTFS	43
2.5.2	Développement de la SAP	43
2.6	Préparation soutenance	45
3	Planning final	45
3.1	Méthodologie de gestion de projet	45
3.2	Planning PRD 2	46
4	Répartition des tâches	47
4.1	Développement de l'application :	47

8	Guide d'installation et d'utilisation	50
1	Guide de lancement de l'application	50
2	Guide de lancement d'un serveur OTP.....	51
3	Guide d'utilisation de l'application	52
3.1	Point d'entrée de l'application.....	53
3.1.1	Planificateur 1	54
3.1.2	Planificateur 2	58
3.1.3	Météo de ma ville	60
IV	Bibliographie	61
V	Webographie	62
	Comptes rendus hebdomadaires	64

Table des figures

2 Description générale

1	Schéma de l'architecture du projet	4
2	Diagramme des Cas d'utilisations	5
3	Structure General du système	6

3 État de l'art/veille

1	Architecture Open Trip Planner	8
2	Support de téléchargement des données OSM pour la ville de paris	9
3	Démarrage d'OTP.....	9
4	Exécution d'un calcul d'itinéraire	10

4 Analyse et Mise en oeuvre

1	Structure générale du système	18
2	Seed Angular Yeoman.....	19
3	Hello World d'une seed Angular 2 YeoMan.....	20
4	Interface du template SB Admin	20
5	Interface du module Météo de ma ville	21
6	Interface du planificateur 1.....	22
7	Interface du planificateur 2.....	24

5 Phase de développement et d'intégration

1	fichiers de configuration du projet	26
2	Diagramme de structure d'un module.....	28
3	Exemple d'utilisation de la convention de nommage.....	28

4	Les différents modules du projet	29
5	Zones sur lesquelles interagissent les modules	29
6	Structure du module Sidebar	30
7	Structure du module Entry-prog.....	30
8	Structure du module Secondary-planner.....	31
9	Structure du module Weather-Page	32
6	Interfaces et Spécifications	
1	Structure d'une SAP	35
2	Design de l'application.....	36
3	Captcha	39
7	Gestion de projet	
1	Répartition des tâches	40
2	Diagramme de Gant prévisionnel	41
3	Diagramme de Gant final : S10	46
4	Diagramme de Gant final : S10	46
5	Représentation du repo git.....	49
8	Guide d'installation et d'utilisation	
1	Commandes du lancement de l'application	50
2	Structure des fichiers du serveur OTP	51
3	Exemple d'une instance OTP	52
4	Image représentant le point d'entrée de l'application	53
5	Image représentant les éléments du formulaire du planificateur 1	54
6	Image représentant la carte du planificateur 1.....	56
7	Image représentant les layers de la carte	57
8	Image représentant un itinéraire affiché la carte	57
9	Image représentant l'interface du planificateur 2.....	58
10	Image représentant les éléments du formulaire du planificateur 2.....	58
11	Image représentant l'affichage d'un itinéraire par le planificateur 2	59
12	Image représentant l'affichage du module météo	60
13	Image représentant les prévisions MinuteCast.....	60



Liste des tableaux

3 État de l'art/veille

1	récapitulatif des caractéristiques de chaque librairie.....	13
2	Récapitulatif des options disponibles au niveau des UI pour chaque librairie.	13
3	Récapitulatif des options disponibles au niveau des couches vectorielles pour chaque librairie.....	13
4	Tableau récapitulatif des descriptifs pour chaque licence.	16
5	Tableau comparatif des différentes licences	16

4 Analyse et Mise en oeuvre

1	Liste des fonctionnalités développées pour le formulaire.....	23
2	Liste des fonctionnalités développées pour la carte	23
3	Comparatif des éléments des planificateurs implémentés.....	25

6 Interfaces et Spécifications

1	Sketch prototype de l'application	36
---	---	----



Introduction

Ce document présente la conception d'une solution permettant la planification des trajets multimodaux avec le choix des moyens de transport, notre solution doit nous fournir le trajet le plus court tout en prenant compte de plusieurs éléments.

Ce projet a été proposé par M. Ismaïla Abderhamane NDIAYE qui représente le MOA, Docteur en Informatique et membre de l'équipe du laboratoire d'ordonnancement et conduite.

Le projet est réalisé par le MOE, M. Younes JAZOULI BENLAHBOUB, pendant la durée du PRD avec un temps de travail de 2 jours /semaine à temps plein.

Première partie

Partie 1 : Recherche et État de l'art

1

Contexte de la réalisation

1 Contexte

Lors d'un trajet un voyageur doit souvent emprunter un à plusieurs modes de transports (voiture, tram, métro ...) exploités par plusieurs transporteurs. Il est donc primordial d'avoir une solution optimisée qui va nous permettre de faciliter cette multimodalité au niveau des correspondances entre modes et réseaux de transports.

2 Objectifs

L'objectif du projet est de créer une application web dont le rôle est le routage des personnes, l'utilisateur pourra renseigner sur l'interface son lieu de départ et d'arrivée ainsi que le choix des modes de transports. Grâce à un calculateur d'itinéraire existant (OTP), l'application va nous retourner le trajet le plus optimisé au niveau du temps tout en prenant en compte les critères de l'utilisateur (tram, bus, voiture...). Au niveau de l'affichage de notre solution, on utilise un Framework JS "OpenLayer" permettant de gérer l'affichage des cartes interactives avec des marqueurs.

3 Bases méthodologiques

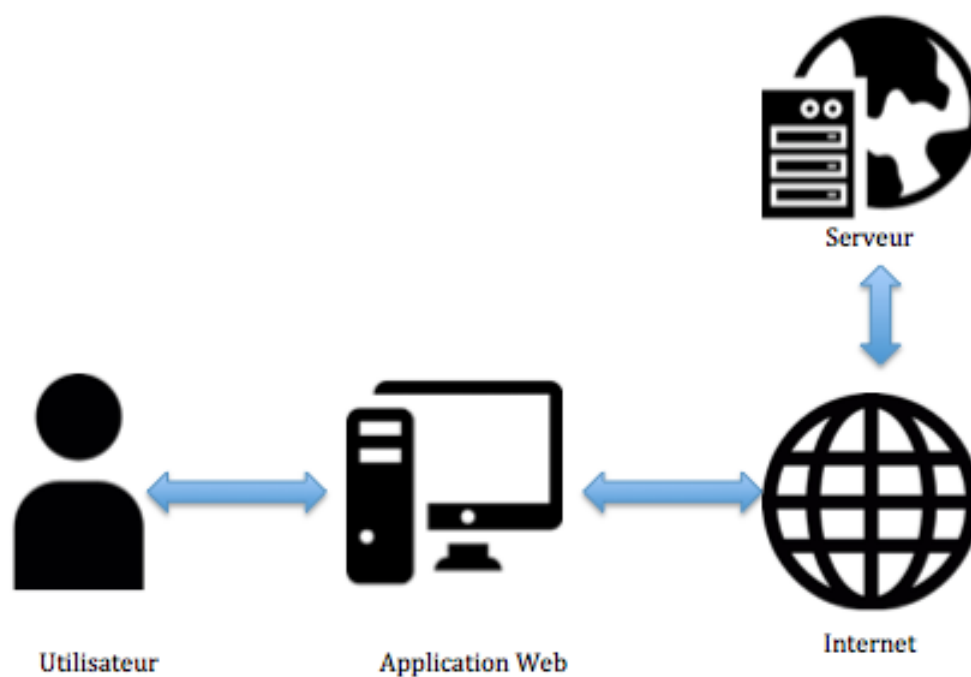
L'application web est une SAP (Simple Page App) développée grâce aux Framework JavaScript Angular JS, ce qui nécessite des connaissances en : JavaScript, HTML et CSS. En ce qui concerne la modélisation des fonctionnalités du système, nous allons utiliser le langage UML. Le Versionning est géré en interne sur un serveur Git privé appartenant à l'encadrant du projet. Nous avons choisi de mettre en place une méthode agile pour le développement du projet, l'outil MsProject est utilisé pour réaliser la planification des tâches afin de mieux organiser les sprints.

2

Description générale

1 Environnement du projet

Le projet va se baser sur l'architecture suivante :



Projet : PRD DI5		Réalisé par : Younes JAZOULI
Document : Schéma de l'architecture		
2/11/2016	Création et élaboration du document	Révision 1.0

Figure 1 – Schéma de l'architecture du projet

Un utilisateur va pouvoir interagir avec l'application web via un navigateur web sur n'importe

quelle plateforme qui dispose d'une connexion internet, ainsi il pourra faire sa requête de planification de trajet.

2 Caractéristiques des utilisateurs

Généralement on peut identifier 2 types d'utilisateurs de l'application :

- Les utilisateurs normaux : ce sont des utilisateurs qui vont uniquement utiliser l'application pour faire une requête de calcul d'itinéraire. L'interface est ergonomique et intuitive, une expérience ou compétence particulière en informatique n'est donc pas nécessaire pour l'utilisateur pour pouvoir l'utiliser.
- Les administrateurs : Sont des personnes qualifiées au niveau des compétences informatiques. Ils auront pour rôle la gestion et la maintenance de l'application.

3 Fonctionnalités du système

Les fonctionnalités principales de l'application se retrouvent dans le diagramme suivant :

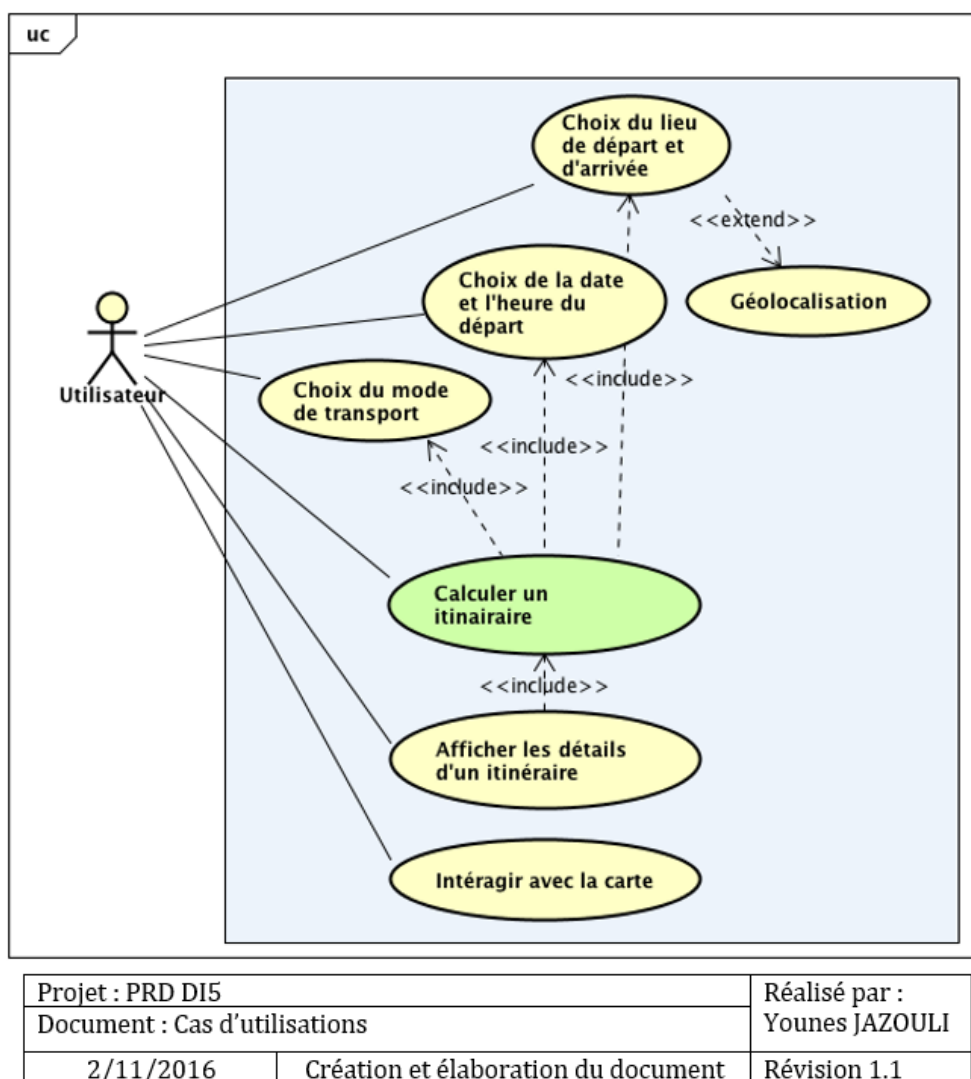


Figure 2 – Diagramme des Cas d'utilisations

4 Structure générale du système

La structure générale du système est représentée par le schéma suivant :

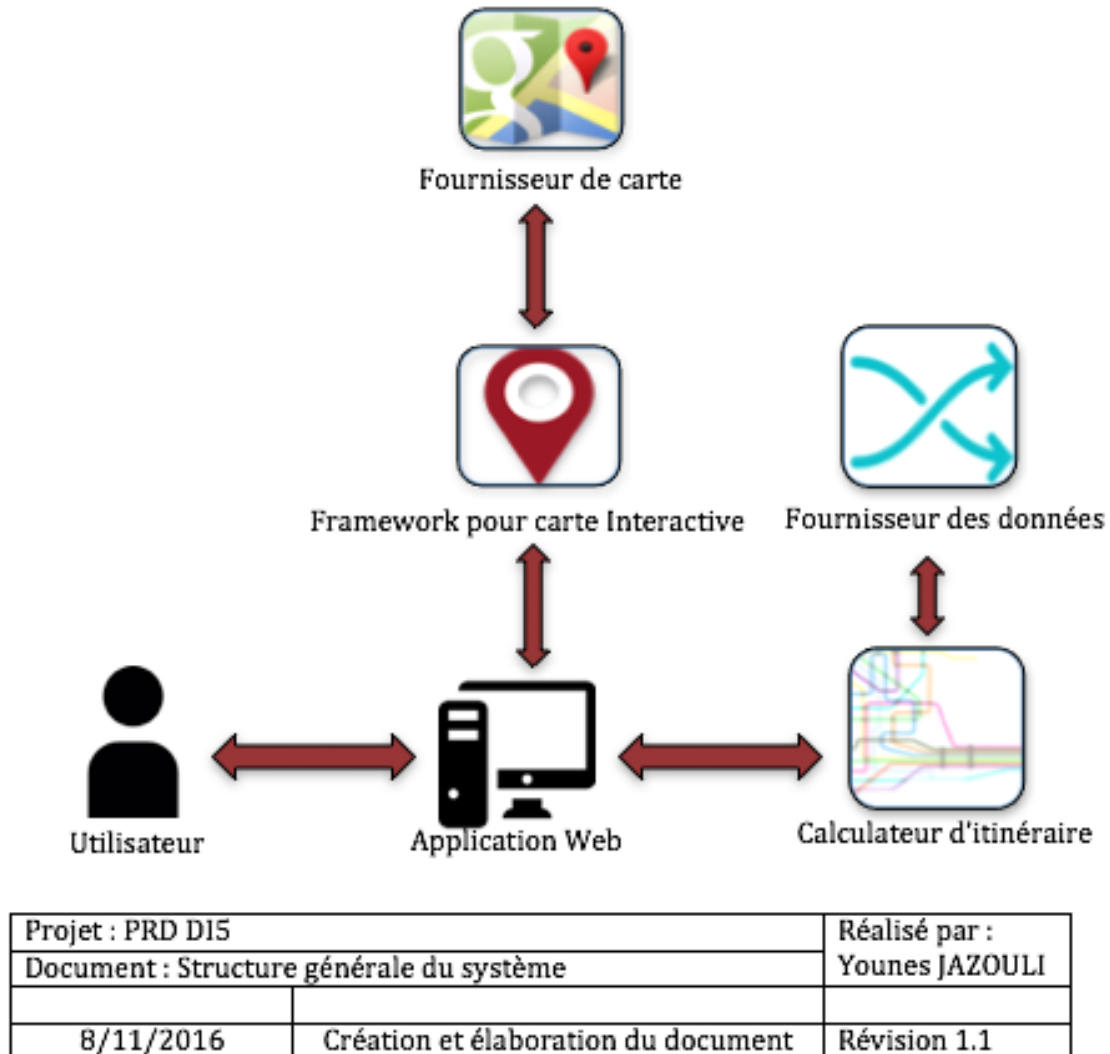


Figure 3 – Structure General du système

La structure se compose de plusieurs composants :

l'utilisateur interagit avec l'application web qui sera en relation avec le calculateur d'itinéraire qui reçoit les données d'un fournisseur afin de pouvoir planifier le trajet, et un Framework pour l'affichage des cartes interactives. Qui va nous permettre de mettre des marqueurs sur la carte pour l'afficher à l'utilisateur.

3

État de l'art/veille

Généralement pour la réalisation du projet on doit disposer de notre interface web, un calculateur d'itinéraire, un Frame Work pour avoir des cartes interactives avec des marqueurs.

On regroupe dans ce chapitre les différentes technologies utilisables ainsi que leurs comparatifs.

1 Calculateurs d'itinéraire

1.1 Open Trip Planner OTP

OTP "Open Trip Planner" est la plateforme Open Source de référence pour le calcul de trajets multimodaux pour une planification de toutes tailles et complexités. Lancé en 2009 avec une collaboration entre TriMet et Open Plans, le projet s'est depuis développé et a attiré une grande communauté d'utilisateurs et de développeurs, avec des déploiements dans une dizaine de pays (Espagne, USA,).

OTP permet de combiner des informations sur des itinéraires à pieds, en vélos et en voiture via une interface web intégrée et pour des applications tierces en fournissant une API "RESTful". Il s'appuie sur des standards ouverts de données (ODS : Open Data Standard) pour le réseau routier (GTFS, GTFS-RT, OSM, MNT,).

1. GTFS

Le GTFS (General Transit Feed Specification) définit un format commun pour la planification des transports publics et des informations géographiques associées pour assurer l'interopérabilité. Il a été conçu par Bibiana McHugh, puis développé par Google et Portland Trimet. Un flux GTFS est composé d'une série de fichiers texte dans une archive ZIP. Chaque fichier modélise un aspect particulier de l'information TC (arrêts, itinéraires, voyages, calendrier...)

2. GTFS-RT

Le format GTFS-RT (GTFS-RealTime), c'est une extension du GTFS qui permet aux organismes de transport en commun de fournir des mises à jour en temps réel de leur flotte. GTFS-RT est conçu autour de la facilité de la mise en œuvre, et mettant l'accent sur l'information des passagers.

Le flux des mises à jour est fourni par l'organisme de transport en commun par une URL vers un fichier "gtfs-realtime.proto". Ce dernier est basé sur le Protocol Buffers de Google (un langage

de sérialisation de données structurées comme le XML, mais en plus rapide, plus facile et plus petit).

3. MNT

Un Modèle numérique de Terrain (MNT) est une représentation 3D de la topographie (altimétrie et/ou bathymétrie) d'une zone terrestre sous une forme adaptée à son utilisation par un calculateur numérique, créé à partir des données d'altitude du terrain. OpenTripPlanner est composé de plusieurs modules, chacun avec un rôle précis. Parmi les modules les plus importants, on trouve :

- Opentripplanner-graph-builder : Une application JAVA en ligne de commande qui permet de générer un objet graphique à partir du référentiel de données grâce auquel on peut calculer les itinéraires.
- Opentripplanner-routing : Le noyau de base du calcul d'itinéraire, on y trouve les classes des sommets (vertex) et des arcs (edge), les interfaces qui composent la structure du graphe ainsi que les différents algorithmes utilisés pour trouver le plus court chemin dans le graphe.
- Opentripplanner-updater : C'est un module permettant de patcher à la volée (dans la mémoire) le Graphe OTP construit afin d'y intégrer les informations en temps réel.
- Opentripplanner-api-webapp : Une application web qui fournit une API WEB RESTful pour effectuer des requêtes sur OTP.
- Opentripplanner-webapp : Une application web front-end communiquant avec OpenTripPlanner fournissant une interface utilisateur pour faciliter l'utilisation d'OTP.

Open Trip Planner utilise 2 différentes méthodes pour les calculs d'itinéraire :

- Algorithme A* : est un algorithme de recherche du chemin entre le nœud initial et le nœud final.
- Méthodes hiérarchiques : sont appliquées sur des graphes plus larges, le concept de ces méthodes est qu'un grand graph peut être contracté en supprimant les sommets un à la fois et en remplaçant tous les chemins par le sommet enlevé avec un raccourci représentant ce chemin.

OTP est sous licence LGPL.

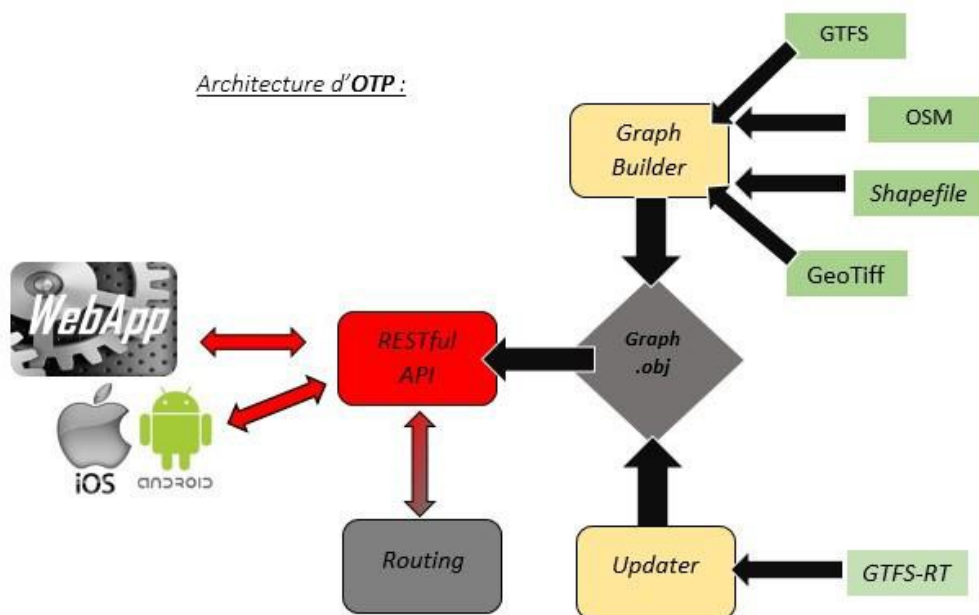


Figure 1 – Architecture Open Trip Planner

1.2 Prise en main d'OTP

1.2.1 Récupération du JAR

OpenTripPlanner est développé en Java et distribué comme un fichier JAR exécutable unique, nous pouvons le récupérer sur l'adresse suivante : <http://docs.opentripplanner.org/en/latest/Getting-OTP/>

1.2.2 Récupération des données de transport

À fin de pouvoir construire notre graph et faire un calcul d'itinéraire on aura besoin des données GTFS et Open Street Map de la ville de paris.

On récupère le fichier OSM à l'adresse suivante <https://mapzen.com/data/>

Paris, France

OSM PBF 173.8 MB	OSM XML 250.3 MB	OSM2PGSQL SHP 360.0 MB	OSM2PGSQL GEOJSON 233.9 MB	IMPOSM SHP 270.7 MB	IMPOSM GEOJSON 361.7 MB
---------------------	---------------------	---------------------------	-------------------------------	------------------------	----------------------------

Figure 2 – Support de téléchargement des données OSM pour la ville de paris

Ainsi que le fichier GTFS à l'adresse : <http://files.transilien.com/horaires/gtfs/export-TN-GTFS-LAST.zip> Les données OSM peuvent être livrées en XML ou en format binaire PBF plus compact.

"Open Trip Planner" accepte les 2 formats, mais nous travaillons toujours avec PBF, car il est plus petit et plus rapide.

Les données OSM téléchargées doivent concorder avec la même région géographique de notre flux GTFS.

1.2.3 Démarrage d'OTP

Comme tout programme Java OTP doit être exécuté sous une machine virtuelle Java (JVM), il est relativement gourmand en mémoire.

On commence par ouvrir un cmd.exe, puis on se met au niveau du répertoire qui contient les fichiers. On lance le serveur OTP avec la commande suivante : `java -Xmx2G -jar otp-0.18.0.jar -build -inMemory` Le paramètre `-Xmx2G` nous permet de définir la limite de consommation en mémoire (dans ce cas 2G => 2Go).

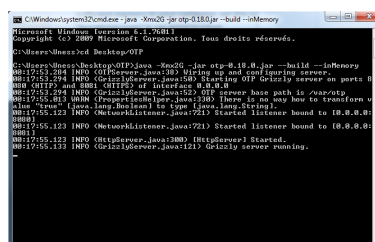


Figure 3 – Démarrage d'OTP

Une fois la commande exécutée le "Graph" est généré après quelques minutes :



Figure 4 – Exécution d'un calcul d'itinéraire

À ce stade, il suffit d'ouvrir : <http://localhost:8080/> dans un navigateur web, et on procède à l'envoi d'une requête d'itinéraire.

1.3 NAVITIA

NAVITIA est un calculateur d'itinéraire pour les transports en commun, il représente un serveur qui expose une API REST qu'on intègre dans divers services (site web, application mobile...).

NAVITIA permet de fournir :

- La planification d'un voyage multimodal - Support de plusieurs régions ou pays. - Recherche à proximité de l'information de la carte. - Horaires pour les lignes de transport en commun. - Départs et arrivées à venir à un arrêt donnés.

Navitia est sous la licence AGPL.

Le calcul routier est simplement l'implémentation de l'algorithme de Dijkstra avec la librairie Boost. La partie transports en commun est l'implémentation de l'algorithme Raptor.

Algorithme de Dijkstra :

L'algorithme est appliqué sur un graphe orienté à partir d'un sommet considéré comme la source. On construit progressivement au court de chaque itération un sous-graphe où les différents sommets sont classés par ordre croissant de leur distance minimale par rapport au sommet de départ. La distance totale correspond à la somme des poids des arcs empruntés.

Navitia utilise un format de données de transit très similaire à GTFS, il permet aussi la gestion des mises à jour en temps réel, mais il reste une solution moins bien optimisée comparée à OTP. Les 2 calculateurs implémentent la "copy on write snapshots" le traitement dans Navitia prend plus de temps, car il fait une copie intégrale alors qu'OTP effectue des copies ciblées.

2 Fournisseurs des cartes

2.1 Open Street Map "OSM"

Open Street Map est un projet fondé en 2004 qui se base sur des données cartographiques libres des routes, voies ferrées, les rivières, les forêts, les bâtiments...

Les données sont ajoutées par des personnes qui veulent contribuer à OSM ainsi un très grand nombre de données cartographiques sont disponible actuellement :

- 800 millions de points géographiques.
- 60 millions de routes, avec un ajout journalier de 150 000 nouvelles routes.

Open Street Map est sous licence : Open Data Base License « 'ODbl »' qui permet d'exploiter publiquement ou commercialement, la base des données sous condition de maintenir la licence sur cette dernière, ainsi que sur les modifications qui y sont apportées.

2.2 Google Maps API

L'API Google Maps permet l'intégration de Google Maps sur les pages web des développeurs externes, en utilisant une interface simple JavaScript ou une interface flash. Il est conçu pour fonctionner sur les deux appareils mobiles, ainsi que les applications traditionnelles du navigateur de bureau. L'API inclut la localisation par région et le géocodage, cette bibliothèque de données va être interrogée par l'utilisateur via des requêtes afin d'afficher une carte sur sa page Web, On peut personnaliser les fonds de cartes (taille, niveau de zoom...).

Google Maps API favorise l'interopérabilité entre les API d'autres services et applications en ligne.

2.3 Comparatif d'OSM et GoogleMAPS

OpenStreetMap est un projet de données ouvert. Les cartes sont publiées dans leur forme la plus brute, gratuitement avec une licence ouverte. Les données brutes sous-jacentes des données qui construisent une carte sont reconnues en tant que données "vecteur".

L'accès à ces données cartographiques brutes permet à tous les types de développeurs de pouvoir créer une gamme de produit et services très intéressants et ambitieux.

Google Maps n'a pas une politique ouverte au niveau de l'exploitation des données cartographiques brutes, la raison de faire ça et de pouvoir maintenir un avantage commercial, tout en exposant uniquement des produits et services en aval générés à partir des données cartographiques brutes. Ils protègent également leurs droits sur les données cartographiques sous-jacentes avec différents droits d'auteur et termes de restrictions d'utilisation.

Malgré le fait que Google maps est largement considéré comme une plateforme de cartographie très ouverte, il y a une prise de conscience croissante que les données cartographiques brutes, et la possibilité de télécharger l'intégralité du "planet.osm" peut générer une nouvelle vague d'innovation qui ne seront pas possible sur la plateforme restreinte de Google.

Néanmoins les services en aval sont importants, OpenStreetMap a des services en aval faisant écho à un grand nombre de services observés au sein de l'offre de Google. Certains sont fournis de base dans OpenStreetMap, mais la plupart sont fournies par des entités tierces.

Que ce soit pour OpenStreetMap ou GoogleMaps, la vue sur la carte est un service "downstream", les cartes sont générées par le processus de "rendering" qui représente une conversion 'vecteur' to "raster", différents styles sont ajoutés à l'image de base qui est ensuite découpée en plusieurs images carrées qu'on appelle des "tuiles", ces dernières sont ensuite introduites en tant que cartes sur le web ou une application mobile.

3 Bibliothèques JavaScript de carte interactive

L'utilisation des bibliothèques JS permet d'afficher des marqueurs sur les cartes, tracer des lignes d'itinéraire personnalisé, montrer une boîte de dialogue dans le cas où l'utilisateur survole certains points sur la carte...

On peut personnaliser les cartes afin de les rendre interactives chaque librairie utilise une source de donnée cartographique (par exemple OSM).

Parmi les librairies les plus connues, on trouve :

3.1 LEAFLET

C'est une librairie qui se caractérise par sa petite taille, ce qui la rend idéale pour les applications mobiles, elle facilite l'ajouter des points sur la carte représentant par exemple les stations vélos ainsi que différentes informations grâce aux « Markers » de leaflet, on peut aussi ajouter plusieurs formes au niveau de la carte comme des pop-up, cercles, rectangles... Le style est personnalisable en utilisant du CSS3.

Source de donnée cartographique utilisée : Open Street Map.

Licence : BSD (Berkeley Software Distribution License)

3.2 OPENLAYERS

OpenLayers est un framework JavaScript open source à haute performance pour construire des cartes interactives en utilisant divers services de cartographie. On peut choisir la source de calque en utilisant la couche de tuiles ou de la couche de vecteur à partir d'un certain nombre de services. OpenLayer est adapté à la construction des cartes à travers les navigateurs. On peut utiliser les CSS pour personnaliser la carte.

Source de donnée cartographique utilisée : Open Street Map.

Licence : BSD (Berkeley Software Distribution License)

3.3 GEOCOMPLETE

GeoComplete est une librairie très utilisée qui dépend de JQuery elle permet d'ajouter un « 'input »' avec la Map qui permet de faire de l'auto complétion des lieux.

Source de donnée cartographique utilisée : Google Maps.

Licence : MIT

3.4 GMAPS

GMaps permet la customisation rapide des cartes, il est compatible avec des données au format JSON.

Source de donnée cartographique utilisée : Google Maps.

Licence : MIT

3.5 MAPLACE

Maplace est un plug-in jQuery pour générer les cartes à travers l'API de Google Maps. Il a la particularité de fonctionner avec tous les navigateurs, y compris IE6.

Source de donnée cartographique utilisée : Google Maps.

Licence : MIT

3.6 Comparatif des différentes librairies

On regroupe les comparatifs des différentes librairies dans les tableaux suivants :

Table 1 – récapitulatif des caractéristiques de chaque librairie.

Caractéristique général					
	<i>Leaflet</i>	<i>OpenLayers</i>	<i>GeoComplete</i>	<i>Gmaps</i>	<i>Maplace</i>
Contributeur	433	164	23	49	—
Première version	13-mai-11	26-juin-06	avr-13	2005	2014
Tutoriels	✓				
Licence	BSD	BSD	MIT	MIT	MIT

Table 2 – Récapitulatif des options disponibles au niveau des UI pour chaque librairie.

Interface Utilisateur					
	<i>Leaflet</i>	<i>OpenLayers</i>	<i>GeoComplete</i>	<i>Gmaps</i>	<i>Maplace</i>
Marqueur	✓	✓	✓	✓	✓
PopUp	✓	✓	X	✓	✓
Tooltip	✓	✓	X	✓	✓
Zoom	✓	✓	✓	✓	✓
Echelle d'affichage	✓	✓	✓	✓	✓
Lat & Long	✓	✓	✓	✓	✓

Table 3 – Récapitulatif des options disponibles au niveau des couches vectorielles pour chaque librairie.

Couches vectorielles					
	<i>Leaflet</i>	<i>OpenLayers</i>	<i>GeoComplete</i>	<i>Gmaps</i>	<i>Maplace</i>
Chemin	✓	✓	X	✓	✓
Polyligne	✓	✓	X	✓	✓
Polygone	✓	✓	X	✓	✓
Cercle	✓	✓	X	✓	✓
Rectangle	✓	✓	X	✓	✓
Toile	✓	✓	X	✓	✓

4 SPA Simple Page APP

Au début des années 90, le web avait pour unique but le partage des informations à travers le monde pour cela il contenait à la base de simples pages HTML statiques contenant du texte on peut donc conclure à ce stade que le web avait pour nature d'être uniquement consultatif.

À partir des années 2000 la notion du web a complètement évolué, l'ergonomie d'une page a beaucoup évolué : l'utilisation du CSS pour la mise en forme, sans oublier le Java Script ainsi que les librairies JQuery pour alléger la syntaxe JS.

Ces éléments ont permis de faire passer l'utilisateur qui avait un rôle consultatif dans le passé vers un rôle d'acteur qui interagit avec une page.

En 2005 le terme SPA (Single Page Application) a été introduit. La différence entre un site web et une SPA réside dans leurs structures ainsi que la relation navigateur serveur.

4.1 Site web

Un site web représente un ensemble de pages dans lesquelles l'utilisateur peut naviguer. Le serveur permet de fournir les pages à afficher et interagit avec les actions de l'utilisateur comme la soumission d'un formulaire.

Le navigateur quant à lui il s'occupe de fournir les pages envoyées par le serveur et transmettre à ce dernier les actions de l'utilisateur.

4.2 SPA

L'avantage des SPA c'est qu'on peut profiter des avantages d'une plateforme web (on a pas besoin d'installer quoi que se soit), le déploiement des mises à jour est quasi instantané.

Une SPA comme son nom l'indique se compose d'une seule page web, l'utilisateur navigue sur plusieurs vues tout en restant sur la même page, la navigateur gère tout ce qui est récupération des données et générer la page à afficher. Le serveur fournit les ressources de l'application (Template, CSS, JS, ...) il expose les données à la manière d'une application web.

On peut trouver comme exemples des SPA très utilisées :

One Drive : c'est un service de stockage de fichier son interface permet de naviguer dans les répertoires en ligne comme sur un répertoire local. GMAIL il permet de consulter, envoyer, rédiger ..., des mails tout en restant sur la même page on rafraichit uniquement les zones nécessaires de la page. On peut conclure que les SPA nécessitent la mise en place du JavaScript pour la gestion de la navigation sur une seule page, mais ceci va poser des problèmes au niveau du : Typage dynamique. La non-compilation du JS.

Ces éléments rendent la création d'app complexe difficile, c'est pour cela que plusieurs Framework JS ont vu le jour.

5 Framework JS pour WebAPP

L'utilisation du JavaScript avec du jQuery permet de construire des interfaces web complexes, mais cela demandait beaucoup d'efforts et de la complexité lors du développement du code et son entretien.

L'utilisation de Framework JavaScript va permettre à l'utilisateur de se concentrer sur le développement des éléments interactifs de l'interface, sans avoir à se soucier de la structure et la maintenance du code.

La plupart des Frameworks JavaScript se basent sur un modèle MVC pour assurer l'évolutivité, la maintenabilité et la réutilisabilité du code JavaScript. Il n'est cependant pas nécessaire que lors de l'utilisation d'un Framework de respecter le modèle MVC, on trouve beaucoup de variations à ce dernier comme MV*, MVVM, MVP selon les besoins du projet.

5.1 ANGULAR JS

AngularJS est le produit de Google, c'est le framework JavaScript le plus ancien. Il a été d'abord publié en 2009 et mis à disposition en tant que framework open source sous licence MIT. Depuis sa sortie, l'écosystème Angular a augmenté au-delà de l'imagination. Il compte actuellement la plus grande communauté de développeurs.

AngularJS donne des super pouvoirs au HTML en ajoutant toutes les fonctionnalités nécessaires pour créer des vues dynamiques (interface utilisateur interactive). Il donne la possibilité de prolonger les attributs HTML par l'utilisation de directives Angular. L'extension du HTML avec AngularJS est très simple, on peut utiliser directive AngularJS standard ou développer une directive sur mesure et la monter sur une div.

Lorsque le compilateur compile et restitue le code HTML pour l'interface utilisateur, il fait la manipulation des DOM et attache toutes les fonctionnalités offertes par la directive utilisée. Deux voies de liaison de données sont au cœur de Angularjs. Lorsque l'utilisateur interagit avec l'interface et fournit une entrée, la vue et le modèle (objets JavaScript) sont synchronisés, les instructions dans le modèle sont exécutées et le DOM est mis à jour. L'inverse est aussi vrai, si le modèle est mis à jour, vue est re-rendu. Cela permet d'éviter toute l'écriture du code pour la manipulation du DOM.

La principale raison de la croissance continue de Angular est les améliorations et les progrès qu'il apporte avec chaque nouvelle version.

5.2 REACT JS

React.js est un Framework très puissant utilisé par Facebook et Instagram il se caractérise comme étant la base des applications de nature dynamique extrême. Il a été publié en open source en 2013 sous licence BSD, il a été utilisé sur plusieurs interfaces utilisateurs très complexes. Le concept derrière ce Framework c'est l'utilisation de DOM virtuel dont le rendu peut être soit du côté serveur ou du côté client.

Un autre grand avantage de React.JS est la réutilisation qu'il apporte sur la table des formes des composants réactifs. Les bibliothèques de composants peuvent être créées et utilisées dans toutes les applications ou mises à la disposition du public.

5.3 EMBER JS

EmberJS est un Framework JavaScript MVC puissant. Ember a été initialement publié en 2011 comme Framework JavaScript open source sous licence MIT.

Ember est basé également sur le principe "two-way data binding" comme AngularJS, Mise à jour de la vue lors des changements sur le modèle, et le modèle est mis à jour lorsque la vue change, en gardant la vue et le modèle synchronisés à tout moment.

Ember comprend le module Fastboot.js qui permet un rendu du DOM côté serveur, le concept est semblable à ce que React utilise pour les interfaces complexes. Ember vise le meilleur des deux Framework AngularJS ('two-way data binding') et ReactJS (rendu côté serveur).

5.4 POLYMERJS

Polymer a été publié par Google en 2013. Il utilise le concept de composants Web pour étendre les capacités de HTML.

On peut créer de nouveaux éléments HTML personnalisés par exemple <my-audio> qui est un élément personnalisé basé sur <audio>.

Polymer permet d'adopter une bonne structure lors de la construction d'éléments HTML personnalisés à l'aide des technologies de navigation de base qui comprennent des composants Web.

6 Comparatif des Licences

Pour réaliser notre projet, on doit aussi faire attention au niveau des licences des technologies qu'on va utiliser.

Ainsi il est primordial de réaliser un petit comparatif des différentes licences sans oublier de lister un descriptif de chacune d'entre elles.

Table 4 – Tableau récapitulatif des descriptifs pour chaque licence.

Licence	Descriptif
BSD	Berkeley Software Distribution License licence libre utilisée pour la distribution de logiciels. Elle permet de réutiliser tout ou une partie du logiciel sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire. (Wikipédia)
MIT	C'est une licence de logiciel libre ¹ et open source ² , non copyleft, permettant donc d'inclure des modifications sous d'autres licences, y compris non libres. La licence donne à toute personne recevant le logiciel le droit illimité de l'utiliser, le copier, le modifier, le fusionner, le publier, le distribuer, le vendre et de changer sa licence. La seule obligation est de mettre le nom des auteurs avec la notice de copyright. (Wikipédia)
ODbl	Permet d'exploiter publiquement ou commercialement, la base des données sous condition de maintenir la licence sur cette dernière, ainsi que sur les modifications qui y sont apportées.
AGPL	La licence GPL connu sous le copyleft, accorde la permission de réutiliser ou de modifier le code source pour créer des œuvres dérivées, mais dans le cas de distribution du programme, il est obligatoire que la licence du travail réalisé soit sous licence GPL aussi.
GPL	AGPL est comme la licence GPL, mais la GPL ne se déclenche que dans le cas de distribution du travail réalisé. AGPL élargit ce concept pour l'activer uniquement si vous laissez les gens utilisent le travail réalisé sur un réseau.
LGPL	La licence LGPL autorise à lier le programme sous cette licence à du code non LGPL, sans pour autant révoquer la licence. Cette Licence LGPL permet donc de s'affranchir du caractère héréditaire de la licence GPL. C'est donc plus précisément la clause de copyleft que n'a pas la LGPL. Ainsi, il devient possible à un programmeur désireux de faire un logiciel propriétaire, d'utiliser certains outils du monde libre, sans contraindre son logiciel à l'être également. Cependant, toute modification de code source dans la bibliothèque LGPL devra être également publiée sous la licence LGPL. (Wikipédia)

Table 5 – Tableau comparatif des différentes licences

	Copyleft	Compatible GPL v2	Compatible GPL v3	Répertoriée par OSI	Pérennise
BSD	Non	Oui	Oui	Oui	Non
MIT	Non	Oui	Oui	Oui	Non
Odbl	Non	Oui	Oui	Oui	Non
AGPL	Oui	Non	Oui	Oui	Oui
LGPL	Oui	Non	Oui	Oui	Oui

Deuxième partie

Partie 2 : Développement

4

Analyse et Mise en oeuvre

Implémentation de la solution

Conformément à mon cahier de spécification rédigé pendant le premier semestre, le but de mon projet est de développer une application Web en AngularJS qui va permettre à son utilisateur de planifier un itinéraire à partir d'un point de départ et d'arriver.

On retrouve dans le schéma suivant la structure de notre système actuel avec la technologie utilisée pour chaque élément.

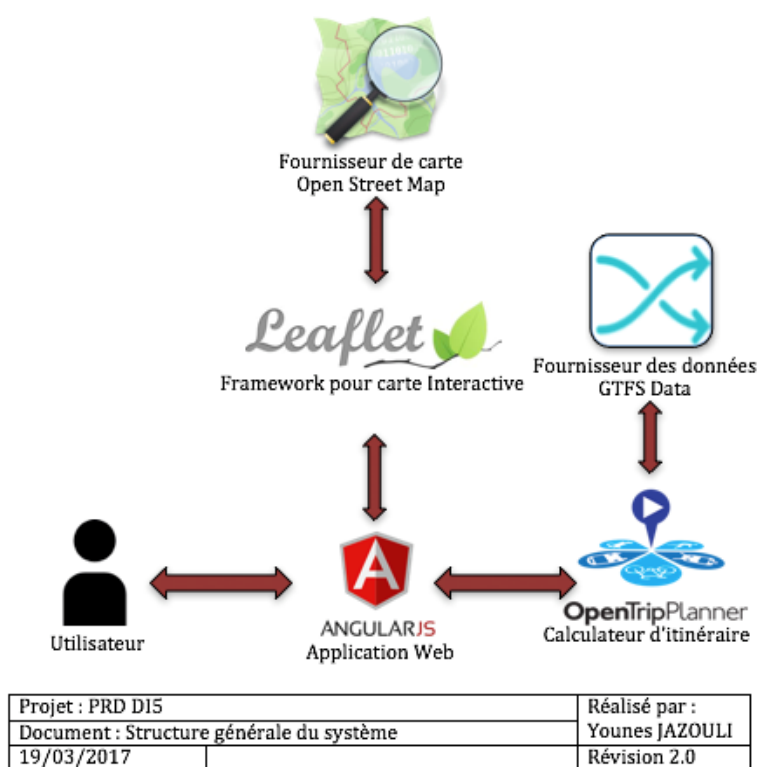


Figure 1 – Structure générale du système

Le choix des technologies de certains éléments a changé contrairement à ce qui a été prévu

à la fin de la phase de recherche du projet. Ainsi les modifications apportées sur ces choix ont dû être approuvées par mon encadrant avant de pouvoir les utiliser pendant la phase de développement.

Pour la réalisation de l'application, j'ai donc utilisé les technologies suivantes :

- Fournisseur des cartes : **Open Street Map**
- Framework pour carte interactive : **Leaflet**
- Service de Géocodage et reverse GeoCoding : **Nominatim**
- Calculateur d'itinéraire : **Open Trip Planner**
- Framework de développement : **AngularJS 2**

Angular Seed

Avant de pouvoir commencer le développement, j'ai eu besoin de me documenter sur AngularJS 2 qui a complètement changé par rapport à son prédécesseur au niveau de la structure et du langage.

Ainsi j'ai décidé de chercher une Seed de projet Angular 2 qui répondra parfaitement aux besoins de mon projet.

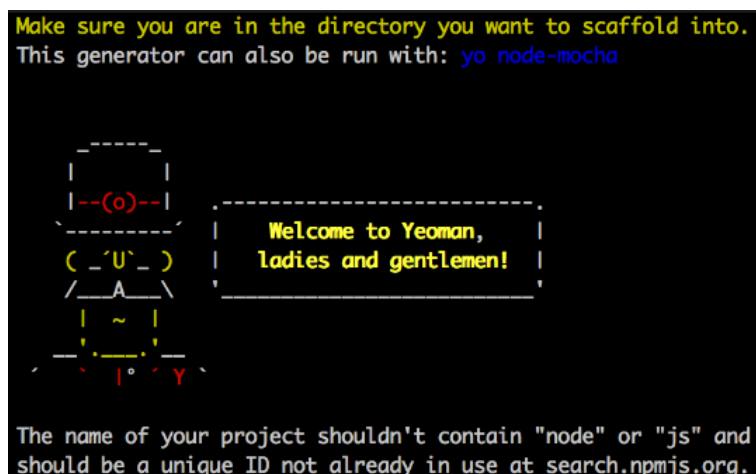


Figure 2 – Seed Angular Yeoman

Une Seed de projet Angular représente un squelette d'application pour une application Web AngularJS typique. On peut l'utiliser pour démarrer rapidement un projet Webapp et l'environnement de développement.

Une Seed contient un exemple d'application AngularJS qui est préconfiguré pour installer le framework Angular et un ensemble d'outils de développement et de test pour une gratification instantanée du développement web.

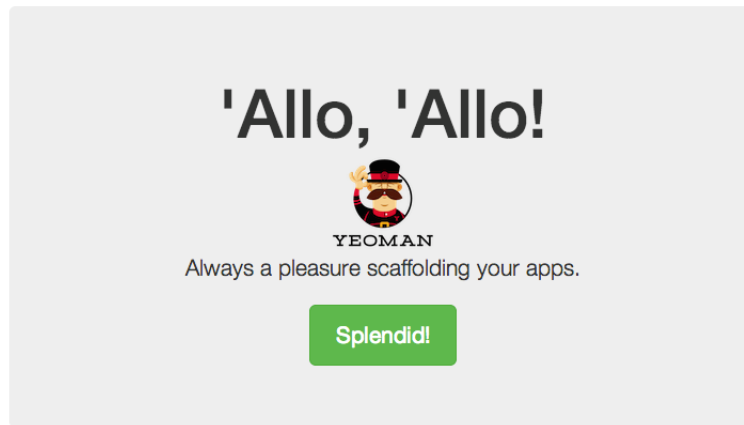


Figure 3 – Hello World d'une seed Angular 2 YeoMan

Une Seed ne fait pas beaucoup, elle montre juste comment lier un contrôleur à une vue en implémentant un modèle d'Hello World sur une page web.

À part le fait d'avoir un squelette de projet fonctionnel, la vraie raison derrière l'utilisation d'une Seed est de prendre en main rapidement la structure de projet Angular. Après avoir testé plusieurs Seed Angular mon choix final s'est porté sur un Template open source SB Admin BS 4 Angular2 basé sur StartAngular et StrapUI.

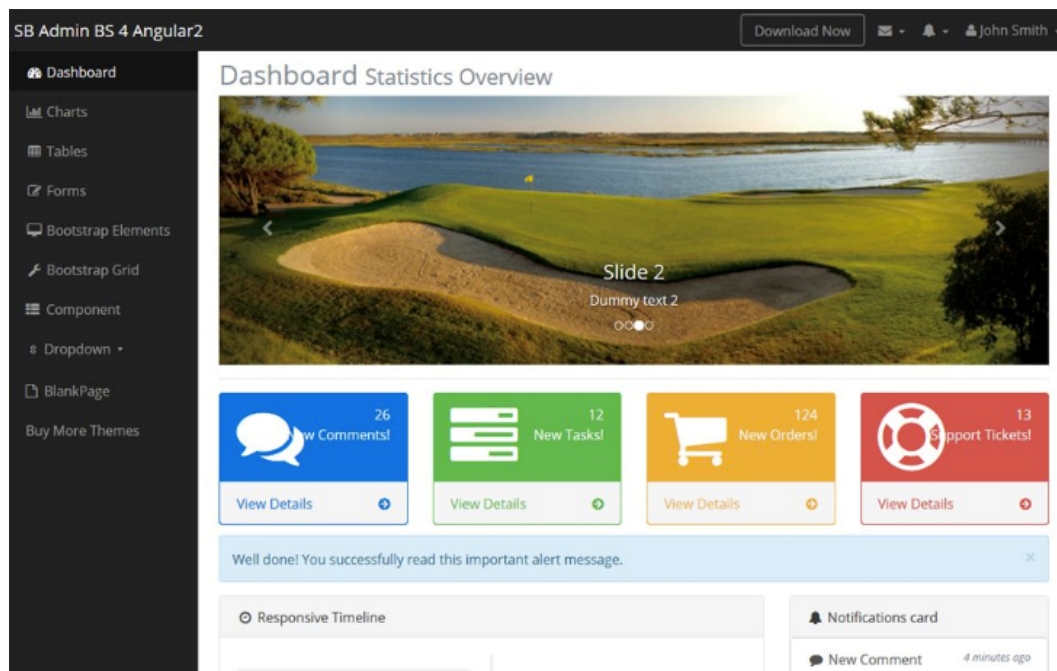


Figure 4 – Interface du template SB Admin

Le modèle de base fournit les fonctionnalités suivantes :

- Permet d'avoir un build en mode **Prod** ou **Dev**.
- Prise en charge de la compilation **Ahead-of-Time**.
- Tests de bout en bout avec **Protractor**.
- Serveur de développement avec **Livereload**
- Gestionnaire des définitions de type en utilisant **@types**
- Utilisation de Gulp pour le build système.
- **Css-lint**.

L'interface de base permet de présenter des éléments du Framework CSS Boot Strap 4.

Implémentation du premier module

Une fois le squelette de l'application en place, j'ai commencé par nettoyer le projet de tous les éléments dont je n'ai pas l'utilité puis j'ai essayé d'implémenter un premier module pour mettre en pratique les connaissances que j'ai commencé à acquérir.

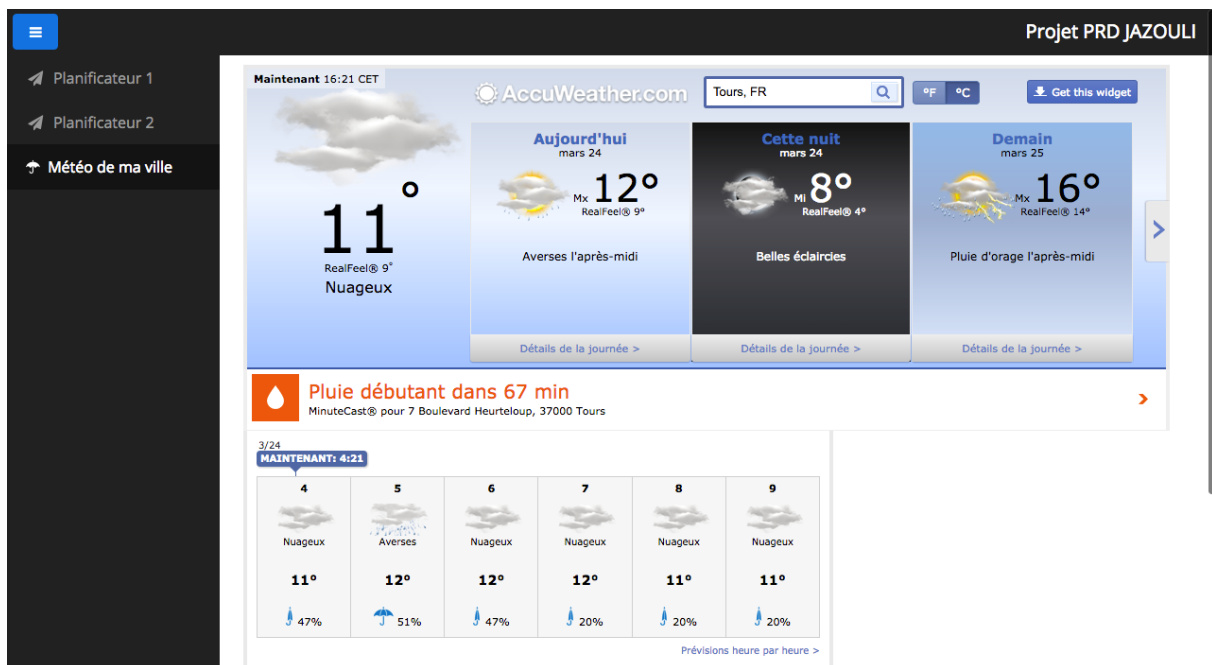


Figure 5 – Interface du module Météo de ma ville

Le premier module que j'ai ajouté à mon application fut d'afficher une weather API basé sur AccuWeather API.

Pour importer l'api il a fallu importer son script JS au niveau du component de mon module. Sauf qu'il est interdit d'avoir des balises `<script></script>` à l'intérieur du Template de son module. Ainsi pour intégrer le JS il a fallu l'injecter avec un service dans le component qui a son tour va l'utiliser sur le Template. Grâce à cette manipulation j'ai pu comprendre et implémenter mon premier service que j'injecte à mon component, chose que m'a été très utile au niveau de mes prochains modules.

Implémentation de mon Planificateur

Planificateur 1 :

Une fois que j'ai commencé à prendre en main Angular JS 2, en implémentant mon premier module. J'ai commencé à implémenter l'interface utilisateur de mon planificateur qui se compose de 2 parties.

Planifier mon itinéraire

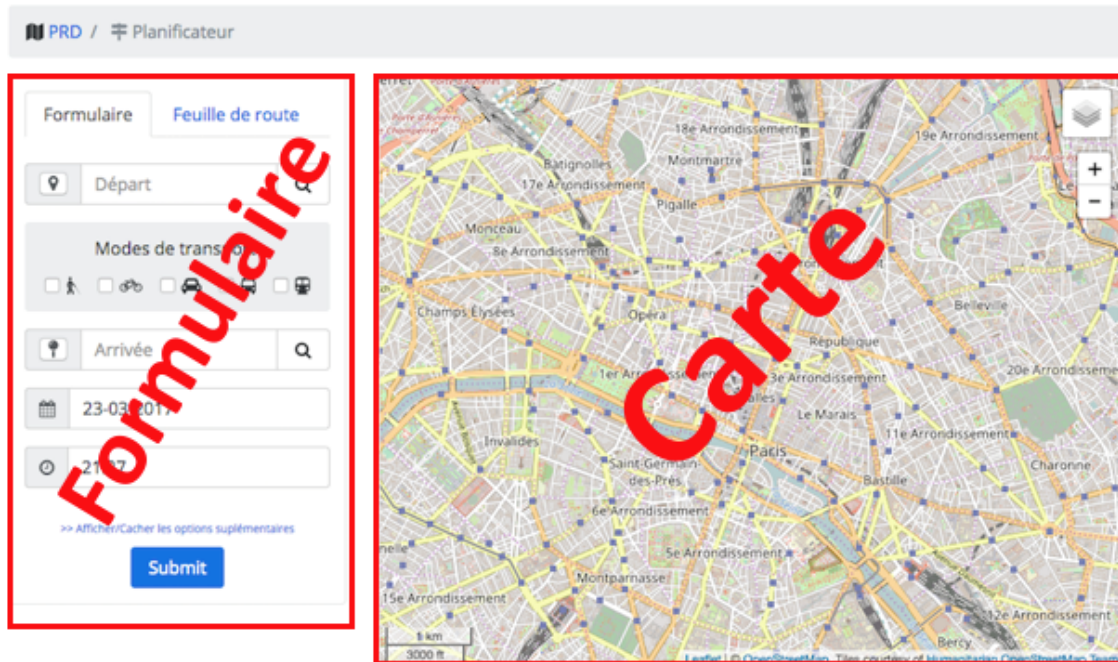


Figure 6 – Interface du planificateur 1

Le formulaire : qui représente l'ensemble des données pour la planification d'itinéraire. Comme je l'ai bien détaillée au niveau de mon guide de l'utilisateur. Le formulaire se compose de :

- Un point de départ.
- Un point d'arrivée.
- Modes de transport.
- Choix de la date de départ.
- Choix de l'heure de départ.

Et dans les options supplémentaires, on peut définir :

- La distance maximale que l'on souhaite parcourir à pied.
- Ainsi que lors de la planification de notre itinéraire de choisir des passages adaptés aux personnes à mobilité limitée.

La carte qui va non seulement afficher le résultat de la planification d'un itinéraire, mais elle va aussi permettre à l'utilisateur de définir son point de départ et d'arrivée via un simple clic sur une position de la carte ce qui va lui faciliter la sélection d'un point de départ et d'arrivée.

Conformément à ce qui a été écrit dans mon cahier de spécification lors du PRD 1, je devais utiliser Open Layer 3 comme librairie de carte interactive. Malheureusement vu le manque de tutoriels, j'ai demandé à mon encadrant de passer sous la librairie Leaflet, qui est utilisée par une vaste communauté de développeur et qui dispose de plusieurs Tutos. Une fois tous les

éléments de mon interface en place j'ai listé toutes les fonctionnalités que j'ai besoin de codé. Le tableau suivant résume les fonctionnalités du formulaire **Table 1** :

Table 1 – Liste des fonctionnalités développées pour le formulaire

Liste des fonctionnalités pour le formulaire	
Élément	Tâches développées
Point de départ et d'arrivée	1) Gestion de l'autocomplétion lors de la recherche d'une adresse 2) Gestion du reverse géocoding pour mettre une adresse comme un point de départ ou d'arrivée 3) Géo localiser sa position comme point de départ 4) Géo localiser sa position comme point d'arrivée
Mode de transport	1) Pouvoir choisir au minum un moyen de transport ou plusieurs moyens de transport en même temps
Choix de la date de départ	1) Pouvoir sélectionner une date de départ avec un Date Picker 2) La date d'un itinéraire par défaut doit être automatiquement la date
Choix de l'heure de départ	1) Pouvoir sélectionner une heure de départ avec un Time Picker 2) L'heure d'un itinéraire par défaut doit être automatiquement l'heure
Distance maximale que l'on souhaite parcourir à pied	Cette distance représente la somme des distances que l'utilisateur doit parcourir pour aller d'arrêt à un autre. 1) Par défaut cette distance doit être égale à 750m 2) L'utilisateur doit pouvoir changer cette distance.
Itinéraire adapté aux personnes à mobilité limité	Cette option représente un paramètre qui sera pris en compte par notre calculateur d'itinéraire 1) Par default cette option n'est pas activé
Valider le formulaire et soumettre une requête d'itinéraire	1) Avant de soumettre une requête d'itinéraire on vérifie qu'on a tout les éléments nécessaire. 2) Si l'utilisateur oublie de renseigner un champ dans le formulaire, on le notifie en lui précisant le champ qu'il a oublié 3) Si toutes les informations sont fournis on crée notre requête d'itinéraire 4) On effectue les traitements sur le résultat retournée et on affiche notre itinéraire sur la carte 4) Notre requête sera transmises par la suite à notre instance Open Trip Planner qui va nous retourner un résultat

Le tableau suivant résume les fonctionnalités développées pour la carte **Table 2** :

Table 2 – Liste des fonctionnalités développées pour la carte

Liste des fonctionnalités pour la carte	
Élément	Tâches développées
Marqueur de départ et d'arrivée	1) Mettre un marqueur de départ quand on choisit un point de départ 2) Mettre un marqueur d'arrivée quand on choisit un point d'arrivée 3) Mettre à jour automatiquement la position d'un marqueur dans le cas ou l'utilisateur change le point de départ ou d'arrivée
Zoomer / dézoomer	1) On peut zoomer et dézoomer sur la carte
Changer de source cartographique	1) L'utilisateur peut choisir entre 3 types de cartes (Esri, Carto DB, Open Street Map) 2) La source cartographique qu'on utilise par défaut est Open Street Map
Afficher l'échelle sur la carte	1) Quand on crée notre carte , on doit pouvoir afficher l'échelle au niveau de la carte.
Map Clic	1) Si l'utilisateur clique sur un endroit de la carte une pop up s'affiche qui lui permet de choisir : Le point où il a cliqué comme point de départ ou comme point d'arrivée. Faire le reverse géo coding pour afficher l'adresse

Ce planificateur représente l'implémentation de ma solution de planification d'itinéraire principale qui se base sur ce que j'avais défini dans mon cahier de charge.

Un planificateur d'itinéraire multimodal en utilisant comme back-end le calculateur Open Source Open Trip Planner.

Planificateur 2 :

Lors de mes recherches sur comment je peux faire l'affichage d'un itinéraire sur une carte Leaflet, j'ai trouvé un outil très intéressant du nom de Leaflet Routing Machine qui représente un moyen facile, flexible et extensible pour ajouter le routage à une carte Leaflet. Avec juste quelques lignes de code, on peut ajouter un routage entièrement fonctionnel. C'est une solution qui permet de personnaliser presque tous les aspects de l'interface utilisateur ainsi les interactions.

Vu la facilité de prise en main de cet outil, j'ai décidé avec l'accord de mon encadrant d'implémenter un nouveau module qui va se baser sur Leaflet Routing Machine.



Figure 7 – Interface du planificateur 2

L'avantage de Leaflet Routing Machine c'est qu'on a juste besoin de créer une Map qu'on lie à un appelle de fonction pour faire un routage d'itinéraire pour avoir un formulaire qui s'intègre dans notre carte. Comme je l'ai précisé dans le guide de l'utilisateur, la solution par défaut nous permet de planifier un itinéraire avec : Un point de départ Un point d'arrivée Un ou plusieurs points intermédiaires. Le calculateur d'itinéraire implémenté par défaut avec Leaflet Routing Machine nous retourne le trajet avec la plus courte distance en prenant compte que ce trajet va s'effectuer en voiture.

J'ai ajouté à ma carte le support de différent type de source cartographique (Carto DB, OSM, Esri).

On regroupe dans le tableau suivant **Table 3** un comparatif des planificateurs d'itinéraire que j'ai implémenté dans mes deux modules.

Table 3 – Comparatif des éléments des planificateurs implémentés

Comparatif des éléments des planificateurs implémentés			
	Liste des fonctionnalités	Planificateur 1	Planificateur 2
Formulaire	Définir un point de départ et d'arrivée	✓	✓
	Définir un ou plusieurs points intermédiaires	X	✓
	Choisir un mode de transport	✓	X
	Pouvoir combiner une multitude de moyens de transport	✓	X
	Définir la date de départ	✓	X
	Définir l'heure de départ	✓	X
	Définir la distance maximale à parcourir à pied	✓	X
	Prise en compte des itinéraires adaptés aux personnes à mobilité réduite	✓	X
	Géocodage	✓	✓
	Reverse Geocoding	✓	✓
	Auto-complétion des adresses	✓	✓
	Géolocaliser sa position comme point de départ / arrivée	✓	X
	Inverser le point de départ et d'arrivée	X	✓
	Affichage de la feuille de route	X	✓
Carte	Zoomer / dézoomer sur la carte	✓	✓
	Changer de source d'affichage des cartes	✓	✓
	Affichage de l'échelle au niveau de la carte	✓	✓
	Gestion du drag & drop d'un marqueur	X	✓
	Gestion du clic sur la carte pour définir un point de départ et d'arrivée	✓	✓
	Affichage de l'itinéraire sur la carte	✓	✓

5

Phase de développement et d'intégration

1 Comprendre la structure de projet AngularJS

1.1 Pourquoi utiliser Angular JS 2

Angular JS 2 est un Framework cross plateforme qui permet de charger rapidement les données cotées serveur. Il permet de manipuler un large set de donnée avec une utilisation minimale de la mémoire sans oublier qu'il offre une structuration simple des projets.

1.2 Environnement du projet

Angular utilise un langage TypeScript qui est un set de code JavaScript migré en code TypeScript. L'image suivante représente une architecture qui met en évidence les fichiers de configuration les plus importants du projet

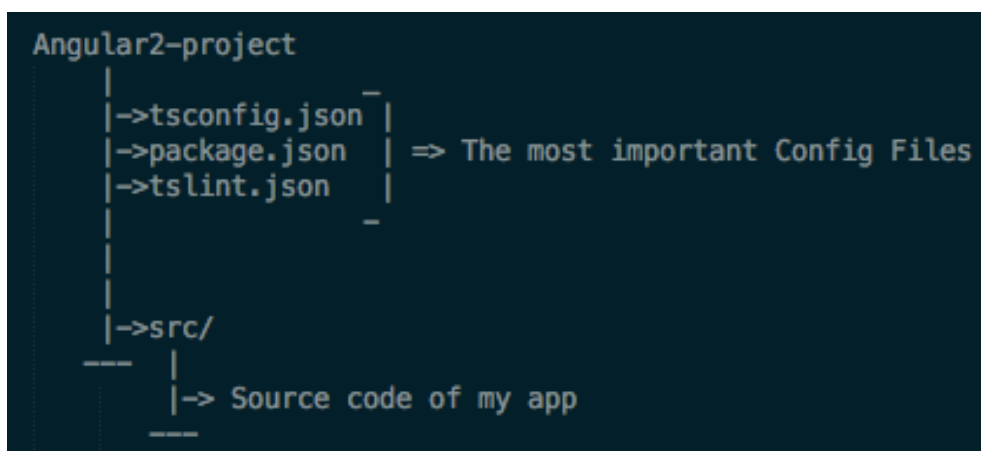


Figure 1 – fichiers de configuration du projet

Parmi les fichiers de configuration qu'on retrouve dans mon projet :

*** `tsconfig.json` : représente le fichier de configuration pour le compilateur TypeScript, il guide le compilateur pour générer les fichiers JavaScript.

*** `tslint.json` : TSLint vérifie que notre code TypeScript est lisible par le compilateur et qu'il ne contient pas d'erreur.

*** `package.json` : contient les packages requis par notre App, ces packages sont installés et maintenus par NPM.

Dans notre dossier `src`, le point d'entrée de notre application sera le fichier `index.html`, on configure ce dernier en utilisant une version du `typescript`. Le système JS transpile le code `typescript` en code JavaScript avant de lancer l'application.

Le TypeScript génère les Meta data pour chaque class du code, quand l'option : `"CompilerOptions: emitDecoratorMetadata: true "`, cette option se trouve dans le fichier `tsconfig.json` qui est au même niveau que le fichier `index.html`. Si cette option n'est pas spécifiée un grand nombre de Meta data inutilisés vont être générés ce qui va affecter la taille du fichier et le Run Time de l'application.

Quand Angular appelle la fonction `Bootstrap` dans le `main.ts`, il lit le Meta data de l'élément, trouve le `app selector`, localise l'élément par son `name tag`, et charge l'application entre 2 tags.

On retrouve dans un projet Angular les notions suivantes :

Module : Un module est conteneur pour les différentes parties de l'application : contrôleurs, services, filtres, directives ...

Component : Un component est un type particulier de directive qui utilise une configuration plus simple qui convient à une structure d'application à base de composants.

Template : Un Template représente le ou les fichiers HTML qui peuvent contenir des éléments et des attributs spécifiques à AngularJS.

Metadata : Les Meta Data sont une façon de traiter la classe. On utilise les Meta Data dans une classe pour indiquer à Angular qu'une cette class est un component et que les Meta Data peuvent être attachées au TypeScript en utilisant le décorateur.

Data Binding : la notion de Data Binding s'explique par le fait que la vue est une projection du modèle en tout temps. Lorsque le modèle change, la vue reflète le changement, et vice versa.

Service : Les services AngularJS sont des objets substituables qui sont câblés en utilisant l'injection de dépendance. On peut utiliser les services pour organiser et partager le code dans notre application.

Directive : sont des marqueurs sur un élément DOM (comme un attribut, un nom d'élément, un commentaire ou une classe CSS) qui indiquent au compilateur HTML d'AngularJS (compile) d'attacher un comportement spécifié à cet élément DOM (par exemple via des `events listeners`).

Dependency Injection : est un design pattern qui traite comment un component récupère les dépendances dont il a besoin. Le système d'injection d'AngularJS prend en charge la création des composants, leur fournir les dépendances, et les fournir à d'autres component s'ils sont demandés.

On peut résumer l'interaction de ces éléments dans le diagramme suivant 2 :

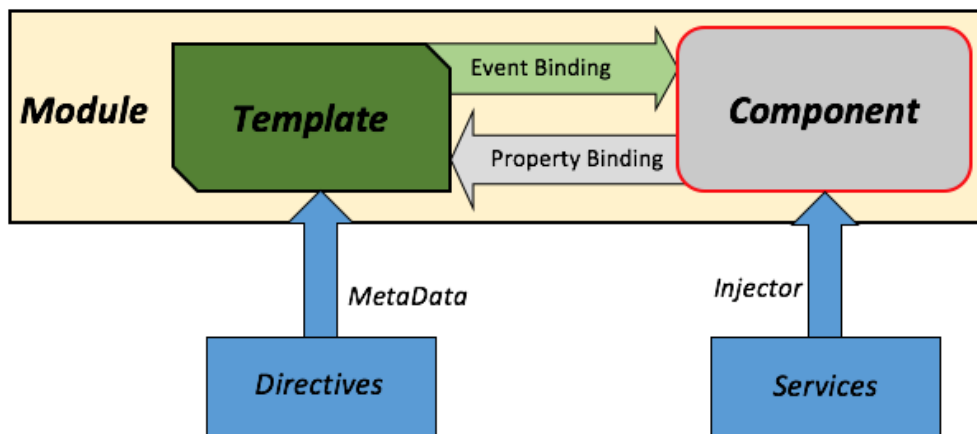


Figure 2 – Diagramme de structure d'un module

1.3 Prise en main du projet

Convention de nommage

La convention de nommage dans un projet AngularJS à respecter est la suivante: lors de l'écriture de mots composés ou de phrases, il faut que chaque mot soit séparé par un tiret (-). Cette forme de nommage est également connue sous le nom kebab-case. Un exemple d'utilisation de la convention de nommage sur ce projet 3.

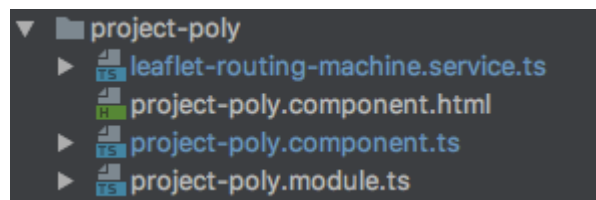


Figure 3 – Exemple d'utilisation de la convention de nommage

Un fichier TypeScript représentant un Component sera écrit de la manière suivante: "mon-fichier.Component.ts". Dans le cas d'un fichier représentant un module : "mon-fichier.module.ts".

Structure du projet

On peut retrouver la structure complète du projet dans le fichier README.md comme on peut le voir le projet se compose de 8 modules:

- app
 - Dashboard
 - Entry-program
 - Secondary-planner
 - Weather-page
- Shared
 - Sidebar
 - TopNav

Le module app représente le module parent dans lequel on retrouve tous les autres modules du projet.

On peut résumer l'apparence des modules avec le diagramme suivant 4 :

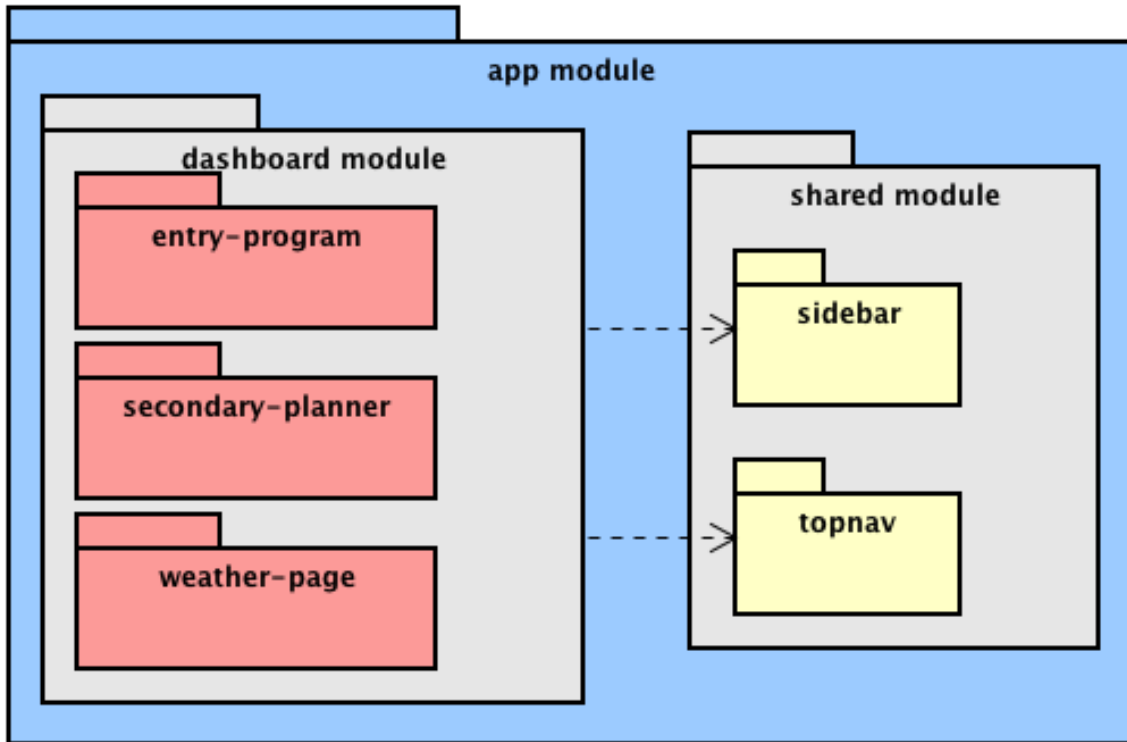


Figure 4 – Les différents modules du projet

L'image suivante 5 représente à quel endroit chaque module interagit.

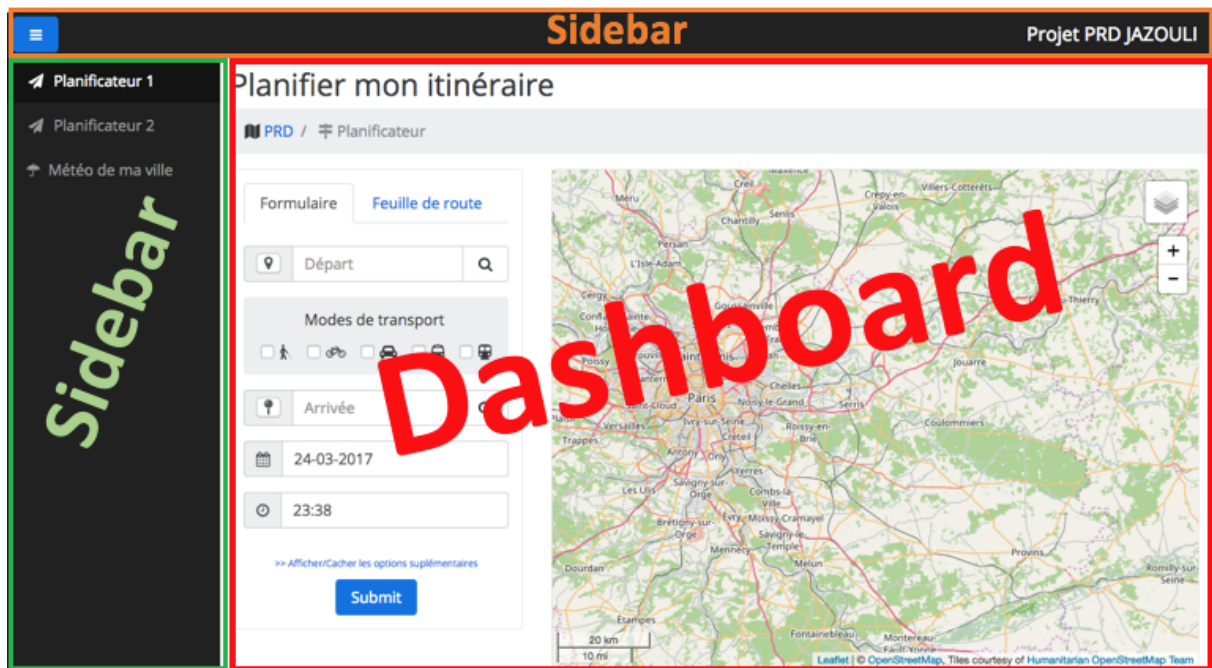


Figure 5 – Zones sur lesquelles interagissent les modules

Module shared

Le module shared se compose de la Sidebar et Topnav qui représentent les éléments qui seront toujours affichés dans notre application web c.-à-d. la barre de navigation en haut de la page et la barre qui se trouve sur le côté la page qui va nous permettre de changer la vue de la Dashboard.

Pour changer la vue de la Dashboard. On utilise la directive Angular "[routerLink]" dans le Template de notre module Sidebar pour faire appel à un des 3 composants de la Dashboard (entry-program, secondary-planner, weather-page).

On résume avec le diagramme suivant 6 la structure du module Sidebar :

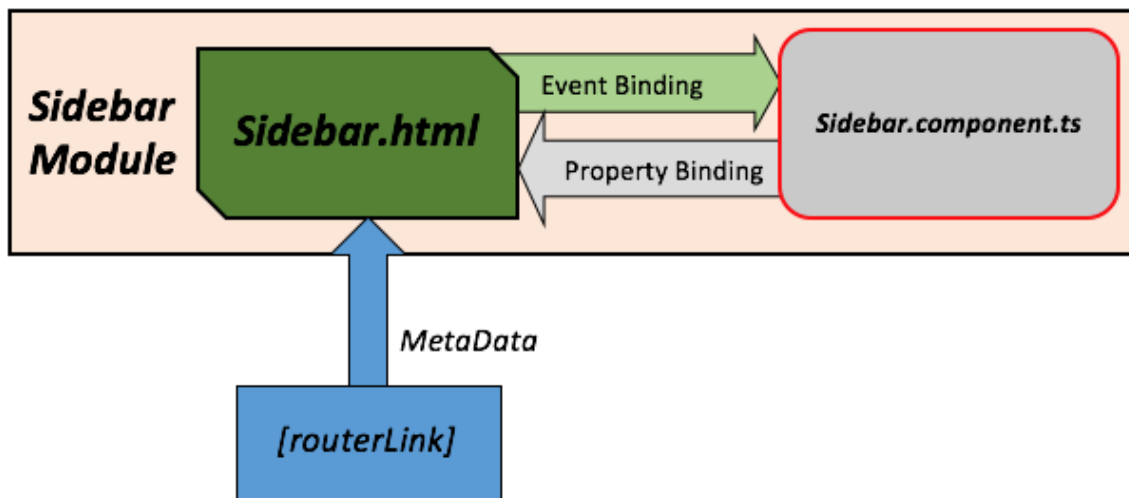


Figure 6 – Structure du module Sidebar

Module Dashboard

On a 3 modules dans notre Dashboard :

Entry-program : qui représente le planificateur qui envoie une requête au calculateur d'itinéraire Open Trip Planner et puis il affiche la réponse de ce dernier sur une carte. On retrouve sa structure dans le diagramme suivant 7 :

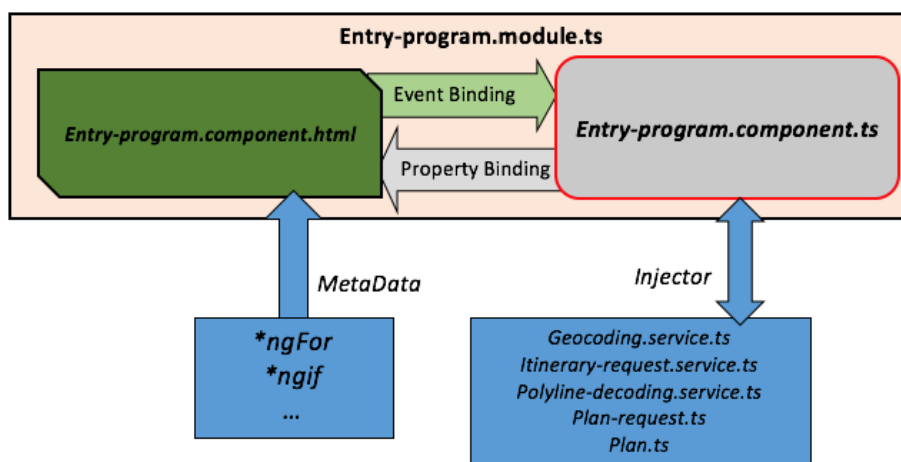


Figure 7 – Structure du module Entry-prog

Comme on peut le constater, ce module se compose de plusieurs éléments :

Le fichier `entry-program.module.ts` : qui va nous permettre d'importer les dépendances qui seront utiliser par le component et aussi d'exporter ce dernier pour qu'il soit accessible de l'extérieur du module.

Le fichier `entry-program.component.ts` : dans ce fichier on va implémenter les principales méthodes pour le traitement du formulaire de requête d'itinéraire, gestion des éléments de la carte, affichage d'un itinéraire ... Pour une meilleure reprise du code, j'ai implémenté plusieurs services qui vont avoir des tâches précises :

`Itinerary-request.service.ts` : représente un service qui va envoyer une requête HTTP en GET au back end qui est le calculateur d'itinéraire. Les paramètres de cette requête sont fournis par les objets de la Class `Plan-request.ts`. La réponse reçue sera ensuite enregistrée dans des objets de la Class `Plan.ts` qui seront ensuite utilisés par la méthode d'affichage de l'itinéraire sur la carte.

`Polyline-decoding.service.ts` : représente un service qui nous permette de décoder la latitude et longitude d'une "leg".

Une "leg" représente une étape d'itinéraire, par exemple : Leg 1 : aller du point A au point B à pied. Leg 2 : aller du point B au point C à vélo.

`Geocoding.service.ts` : est un service qui envoie une requête au service de géo codage grâce à cette requête on va pouvoir récupérer l'adresse complète un point géographique.

Secondary-planner : Ce module représente un planificateur d'itinéraire basé sur Leaflet Routing Machine après avoir téléchargé la dépendance "leaflet-routing-machine : 3.2.5", il suffit de faire appel à la méthode de routage pour avoir un formulaire qui s'intègre avec la carte. On retrouve sa structure dans le diagramme suivant 8:

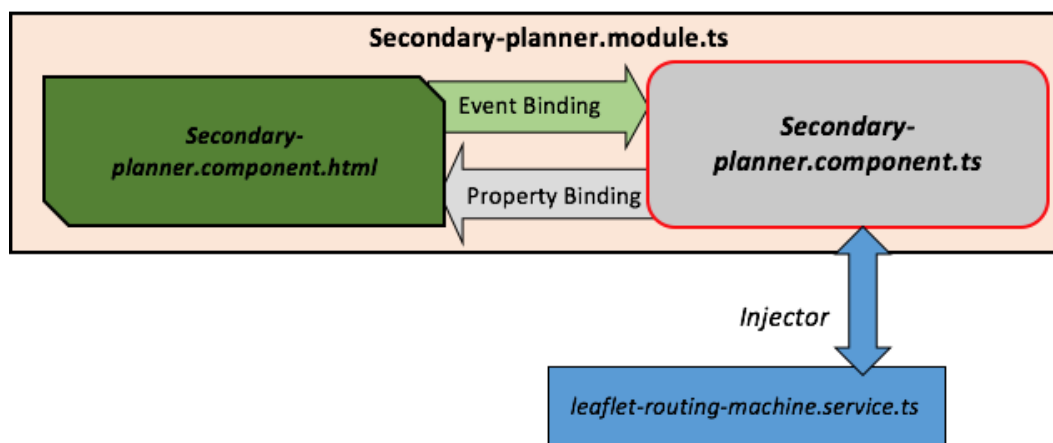


Figure 8 – Structure du module *Secondary-planner*

Notre module utilise un seul service qui est à la base un service que j'ai implémenté pour injecter différentes sources des cartes dans la vue.

Le calcul d'itinéraire avec ce planificateur est réalisé par le calculateur OSRM développé en C++, Leaflet Routing Machine utilise par défaut le serveur démo d' OSRM pour la planification des itinéraires.

Weather-Page : Ce module représente l'intégration d'une application météo basée sur l'API de accuweather. On retrouve sa structure dans le diagramme 9.

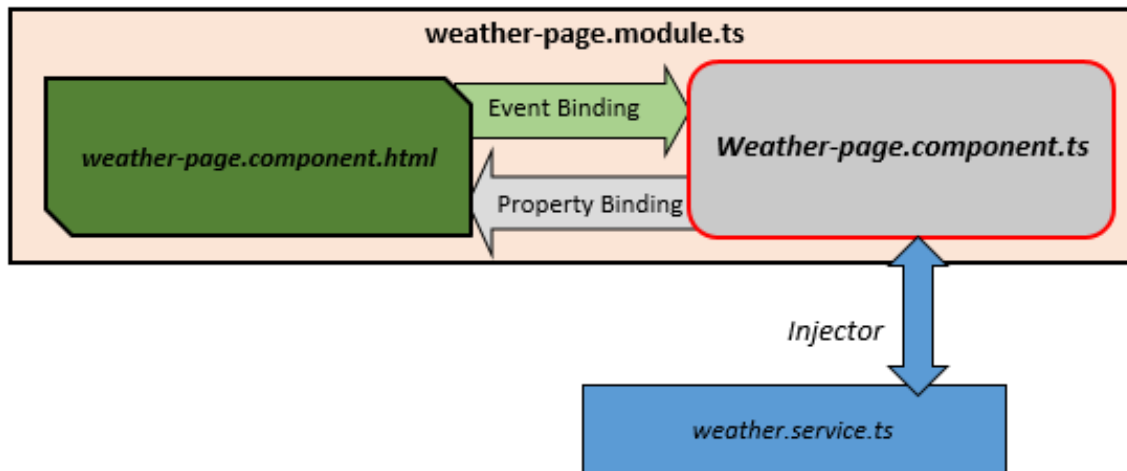


Figure 9 – Structure du module Weather-Page

Comme nous pouvons le constater, ce module se compose de plusieurs éléments. Nous avons le module, le component, le template ainsi qu'un service qui permet d'injecter le JavaScript de l'API de accuweather.



Conclusion

Ce projet m'a permis de travailler sur la résolution de la problématique du routage multimodale des personnes à travers le développement d'une application web.

Pendant le premier semestre de ce projet, j'ai fait une étude sur les différentes technologies qu'on peut utiliser pour la réalisation du projet.

Durant le PRD2, j'ai développé une application dont laquelle j'ai implémenté deux solutions de planification d'itinéraire.

Au niveau organisationnel, mon encadrant était toujours disponible ce qui m'a permis de faire le point régulièrement, de résoudre rapidement les problèmes rencontrés et de valider chaque étape du projet.

Troisième partie

Annexe

6

Interfaces et Spécifications

1 Description des interfaces externes du logiciel

1.1 Interfaces matérielles/logiciel

L'utilisateur aura besoin d'un ordinateur connecté à internet avec n'importe quel système d'exploitation récent comportant un navigateur web compatible JavaScript.

1.2 Interfaces homme/machine

La SAP va se composer de plusieurs parties :

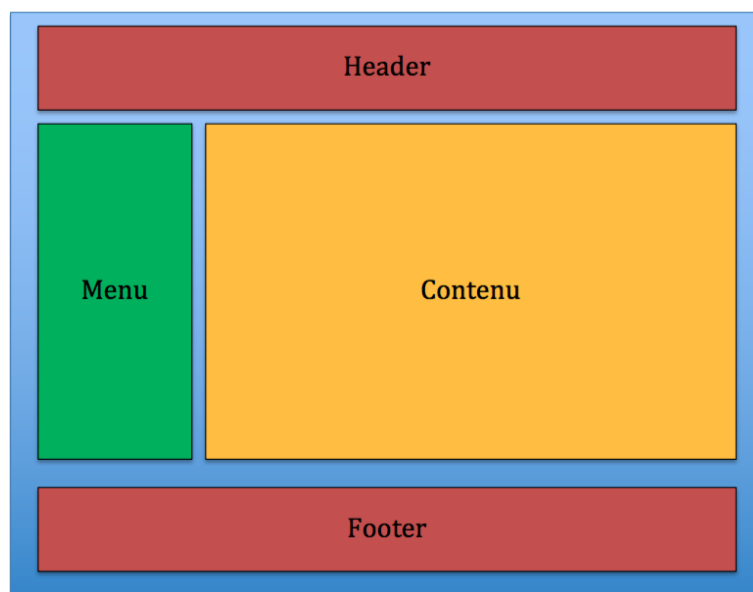


Figure 1 – Structure d'une SAP

Au niveau du header les données météo du jour sont affichées. Ceci peut jouer un rôle très important pour l'utilisateur dans le choix d'un mode de transport. Dans la partie menu l'utilisa-

teur peut sélectionner : * Un lieu de départ * Un lieu d'arrivée * Date et heure de départ * Mode de transport : (Voiture, Tram, Bus, Vélo, à pied).

La partie "Contenu" est consacrée à l'affichage du trajet sur la carte interactive permettant de zoomer/dézoomer, ainsi que sur une liste textuelle.

La zone du "Footer" comporte des informations sur la version de l'application ainsi que l'adresse de contact de l'administrateur.

Le schéma suivant représente un "Sketch" de l'application :

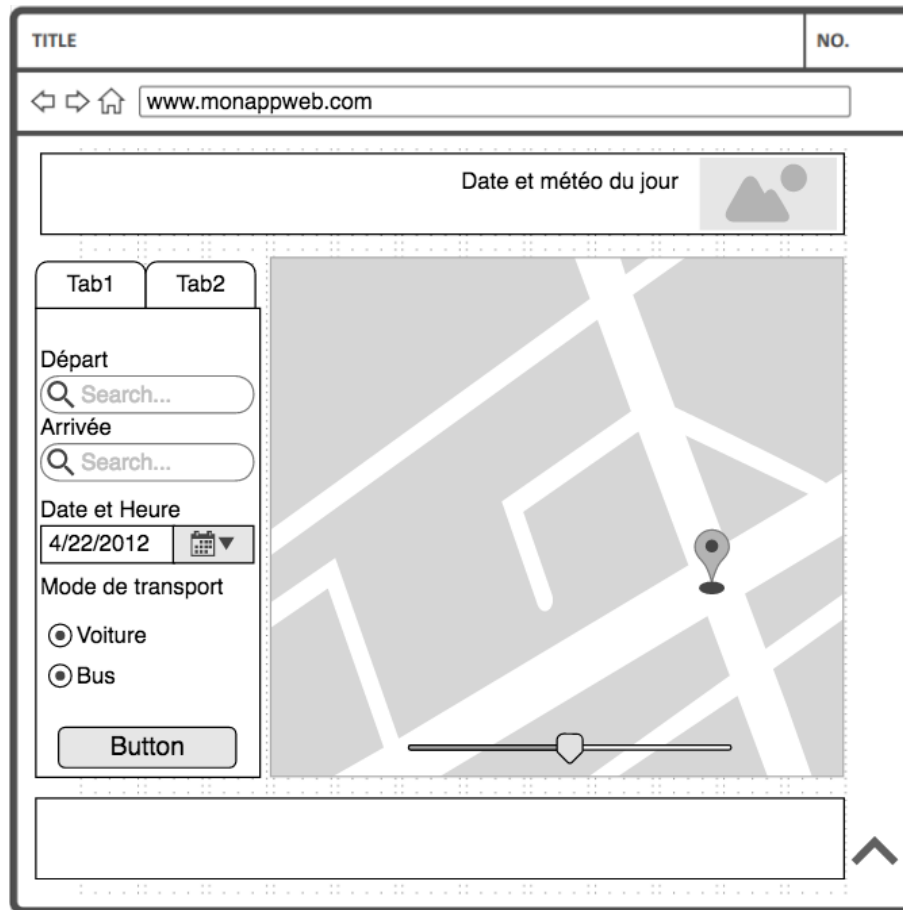


Figure 2 – Design de l'application

Table 1 – Sketch prototype de l'application

Projet : PRD DI5		Réalisé par :
Document : Sketch de l'application		Younes JAZOULI
15/11/2016	Création et élaboration du document	Révision 1.0

Les dispositions de plusieurs éléments peuvent être susceptibles de changer durant la phase de développement pour offrir une meilleure ergonomie à l'utilisateur.

1.3 Interfaces logiciel/logiciel

Le développement de l'application se fera sur un serveur local. Une fois déployé sur un serveur, l'accès se fera avec un navigateur web via une adresse.

En ce qui concerne les différentes interactions des composantes de notre architecture système. L'utilisateur va effectuer une recherche d'itinéraire sur l'interface qui est une SAP, suite à cette

recherche les critères cette dernière seront ensuite transmis au Backend qui est le calculateur d'itinéraire. À son tour il va rechercher au niveau des fichiers Google Fed transit les points de départ et d'arriver, puis générer un graph et avec son module routing.

Il va retourner un itinéraire en prenant en compte les modes de transport. Les données de cet itinéraire seront sous format Json ou XML avec ces données notre librairie de carte interactive va nous tracer l'itinéraire sur une carte Open Street Map.

2 Spécifications fonctionnelles

2.1 Définition de la fonction de recherche d'itinéraire

Cette fonction permet de lancer une recherche d'itinéraire. Elle prend en entrée :

- Point de départ • Point d'arrivée • Choix des modes de transport • Heure/Date de départ/arrivée

Elle permet de retourner un itinéraire optimisé au niveau du temps et de l'afficher sur une carte.

*Priorité : primordiale

2.2 Définition de la fonction de récupération des données météo

Cette fonction est réalisée en intégrant une WEATHER API, elle affichera la météo journalière.

*Priorité : facultative

2.3 Définition de la fonction d'affichage des détails d'un itinéraire

Grâce à cette fonction, on peut afficher les différentes étapes de notre itinéraire en partant par le point de départ vers le point d'arrivée, tout en mettant en évidence les détails de chaque étape intermédiaire.

Chaque étape se caractérise par ces éléments :

- Un point de départ et d'arrivée • L'heure de départ et d'arrivée • Le temps nécessaire pour le passage du point A vers un point B • Le nom des rues • Le nom des arrêts dans le cas d'utilisation des transports en commun

*Priorité : primordiale.

2.4 Définition de la fonction de Zoom sur la carte

Avec cette fonction, l'utilisateur va pouvoir zoomer et dézoomer sur la carte pour avoir une vue globale ou détailler selon son choix. Cette fonction sera réalisée avec le Framework JS "OpenLayer" pour cartes interactives.

*Priorité : primordiale.

2.5 Définition de la fonction d'auto complétion pour les champs de recherche des lieux de départ/arrivée

Cette fonction va permettre à l'utilisateur de voir une liste de propositions s'afficher lors de la recherche d'un lieu de départ/arrivée, en fonction de ce qu'il écrit.

*Priorité : primordiale.

2.6 Définition de la fonction de sélection de la date et l'heure

Cette fonction va permettre à l'utilisateur de choisir la date et l'heure durant laquelle il souhaite faire son trajet. L'itinéraire peut varier selon les disponibilités des moyens de transport public qui dépendent du temps de travail durant la journée.

*Priorité : primordiale.

2.7 Définition de la fonction de géolocalisation

La fonction de géolocalisation va permettre à l'utilisateur une certaine flexibilité au niveau du choix de son lieu de départ. Ainsi il pourra choisir le lieu actuel ou il se trouve comme point de départ pour l'itinéraire qu'il souhaite planifier.

*Priorité : primordiale.

3 Spécifications non fonctionnelles

3.1 Contraintes de développement et de conception

Pour le développement de ce projet, on utilise un ordinateur sous Windows.

Pour la partie logicielle, on utilise l'IDE IntelliJ IDEA en version 2016.2.5, et Google Chrome comme navigateur web.

On utilise Google Maps comme fournisseur de carte, Angular JS comme Framework JS pour le développement de la SAP, et des bibliothèques JS pour les cartes interactives.

3.2 Contraintes de fonctionnement et d'exploitation

3.2.1 Performances

Au niveau des performances de l'application lors de l'utilisation en temps réel il faut prendre en compte certaines mesures :

- Point de vue de l'utilisateur : le temps de réponse pour une requête de calcul d'itinéraire ne doit pas dépasser les 10 secondes, le résultat final sera l'affichage du trajet sur la carte avec toutes les informations complémentaires qui lui sont liées.
- Point de vue de l'environnement : L'application doit pouvoir gérer plusieurs clients, et donc pouvoir répondre à plusieurs requêtes en parallèle.

3.2.2 Capacités

Les capacités limites des traitements en parallèle qui peuvent être effectués par l'application sont en rapport avec la puissance matérielle du serveur dans lequel l'API sera hébergée.

3.2.3 Sécurité

Afin de protéger le serveur contre les attaques DDOS ou l'utilisation d'un robot pour faire plusieurs requêtes, l'intégration d'un CAPTCHA reste primordiale.

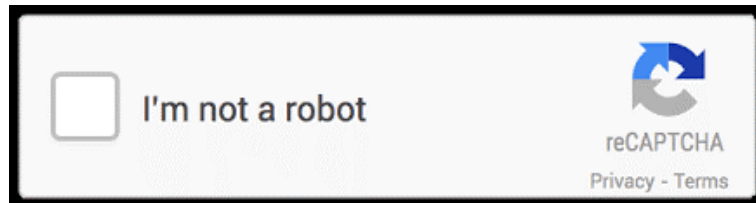


Figure 3 – Captcha

L'utilisateur va devoir cocher le CAPTCHA avant de pouvoir soumettre le formulaire de recherche.

3.2.4 Maintenance et évolution du système

Afin de prendre en compte des critères très importants dans notre application web qui sont l'évolutivité et la maintenabilité cette dernière, dans le cas où elle est reprise par un autre développeur ou un groupe de développeurs, on s'assure d'implémenter un pattern MVC appliqué à un Framework JS.

Comme on travaille sur une SPA on va répartir le traitement de chaque zone de la page sur un contrôleur, ainsi on va avoir plusieurs contrôleurs ce qui va nous offrir une certaine hiérarchie et flexibilité sur le code dans le cas où plusieurs personnes travaillent dessus en même temps.

7

Gestion de projet

1 Planning prévisionnel




























		Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
0			PR&D	139 jours	Mer 21/09/16	Lun 03/04/17	
1			Etat de l'art et CDS	59 jours	Mer 21/09/16	Lun 12/12/16	
2			Documentation, lecture, compréhension	31 jours	Mer 21/09/16	Mer 02/11/16	
3			RDV : 1	0 jour	Mer 28/09/16	Mer 28/09/16	
4			RDV : 2	0 jour	Mer 12/10/16	Mer 12/10/16	
5			Outils liés au développement	31 jours	Mer 12/10/16	Mer 23/11/16	4
6			RDV : 3	0 jour	Mer 23/11/16	Mer 23/11/16	
7			Rédaction du CDS	53 jours	Mer 28/09/16	Ven 09/12/16	
8			Rédaction de l'état de l'art	53 jours	Mer 28/09/16	Ven 09/12/16	
9			Préparation soutenance	8 jours	Jeu 01/12/16	Dim 11/12/16	2;5
10			Soutenance	0 jour	Lun 12/12/16	Lun 12/12/16	9
11			Développement	76 jours	Lun 19/12/16	Lun 03/04/17	10
12			Génération des données GTFS	70 jours	Lun 19/12/16	Ven 24/03/17	
13			Développement de la SAP	76 jours	Lun 19/12/16	Lun 03/04/17	
14			Mise en place d'un prototype d'une App Angular Js	14 jours	Lun 19/12/16	Jeu 05/01/17	
15			Intégration des cartes interactives	5 jours	Ven 06/01/17	Jeu 12/01/17	14
16			Intégration des différents éléments de l'interface de recherche	5 jours	Ven 13/01/17	Jeu 19/01/17	15
17			Intégration d'un moteur d'auto complétion	10 jours	Ven 20/01/17	Jeu 02/02/17	16
18			Intégration du Backend	5 jours	Ven 03/02/17	Jeu 09/02/17	17
19			Affichage des résultats en textuelle	5 jours	Ven 10/02/17	Jeu 16/02/17	18
20			Affichage des résultats sur la carte interactive	10 jours	Ven 17/02/17	Jeu 02/03/17	19
21			Affichage des résultats sur la page web	15 jours	Ven 03/03/17	Jeu 23/03/17	20
22			Phase de Test	7 jours	Ven 24/03/17	Lun 03/04/17	21
23			Développement d'un service rest pour le temps réel	22 jours	Mer 01/03/17	Jeu 30/03/17	
24			Préparation soutenance	6 jours	Lun 27/03/17	Lun 03/04/17	
25			Soutenance	0 jour	Lun 03/04/17	Lun 03/04/17	24

Figure 1 – Répartition des tâches

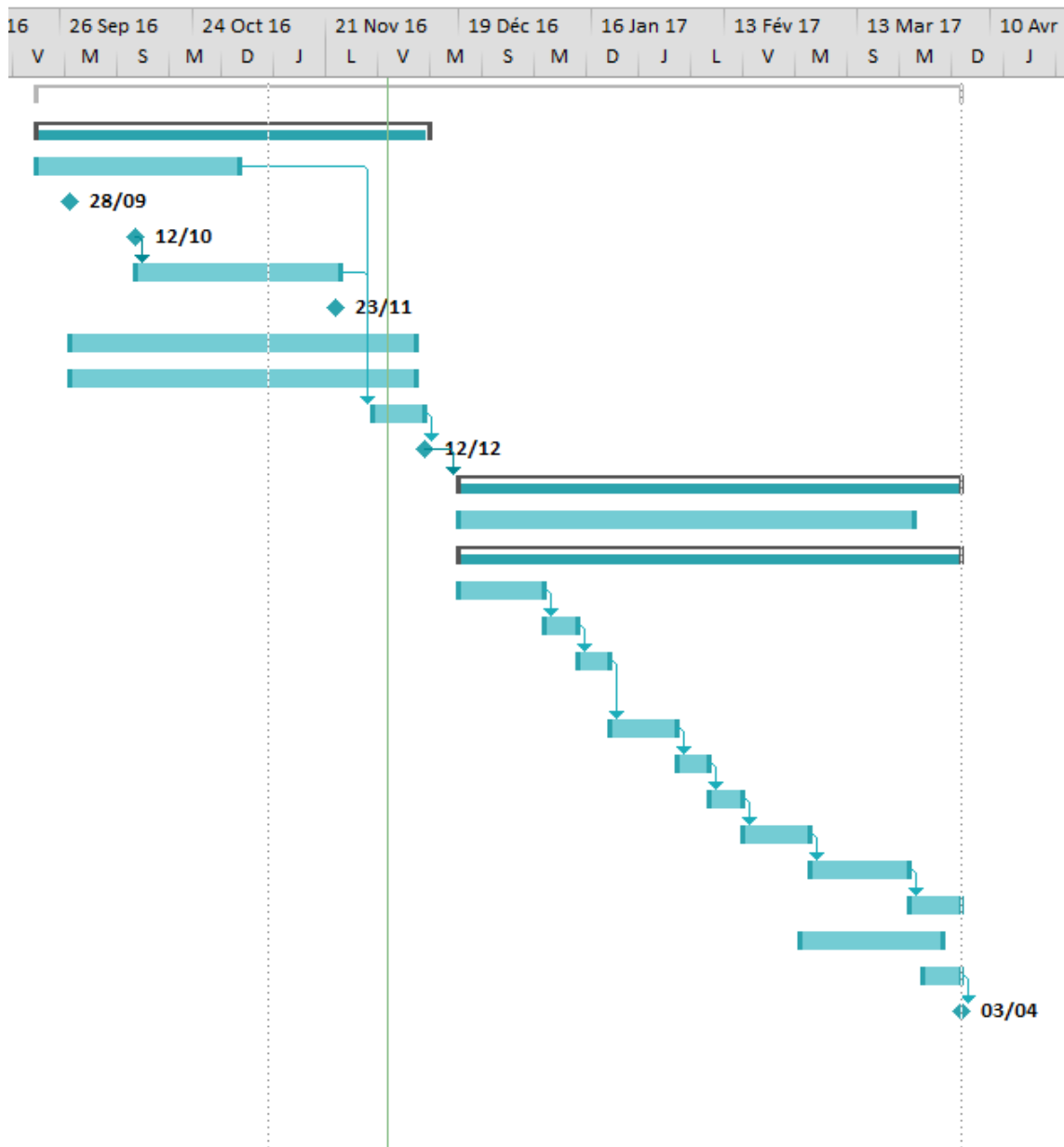


Figure 2 – Diagramme de Gant prévisionnel

2 Répartition des tâches

2.1 Documentation, lecture, compréhension

Description de la tâche :

Durant cette tâche il a fallu comprendre le contexte du projet, fixé les objectifs à atteindre à la fin du PRD. J'ai débuté la recherche d'information les technologies existantes, connaissances à acquérir pendant la phase de développement.

Livrable :

Un rapport de 10 pages concernant un début de recherche qui sera utilisé dans la rédaction du rapport de l'état de l'art.

Cycle de vie et contrainte temporelle :

Cette tâche doit être validée par le MOA du projet.

2.2 Outils liés au développement**Description de la tâche :**

Cette tâche est basée uniquement sur des recherches internet après la mise en évidence de l'architecture générale du projet. Elle comprend un ensemble d'outils technologiques ainsi que leurs comparatifs pour pouvoir choisir ceux qui sont le mieux adaptés à la réalisation du projet.

Livrable :

Un rapport sur les différentes technologies et leurs comparatifs qui sera ensuite introduit dans le rapport de l'état de l'art.

Cycle de vie et contrainte temporelle :

Cette tâche doit être validée par le MOA du projet.

2.3 Rédaction de l'état de l'art**Description de la tâche :**

Cette tâche consiste à la rédaction du rapport de l'état de l'art en se basant sur les différents rapports fournis en amont durant les différentes recherches.

Livrable :

Un rapport contenant l'état de l'art du projet

Cycle de vie et contrainte temporelle :

La rédaction de l'état de l'art comprend toute la durée du PRD1 "de septembre au 9 décembre".

2.4 Rédaction du CDS**Description de la tâche :**

Le but de cette tâche étant la rédaction du cahier de spécification avec la définition du contexte, de l'objectif ainsi que les fonctionnalités attendues .

Livrable :

Un cahier de spécification avec un diagramme de Gant prévisionnel pour le projet

Cycle de vie et contrainte temporelle :

La rédaction de l'état de l'art comprend toute la durée du PRD1 "de septembre au 9 décembre".

2.5 Développement**Description de la tâche :**

Cette tâche regroupe toutes les tâches en rapport avec le développement et l'intégration des différents éléments pour la réalisation de l'application.

Elle regroupe 2 grandes tâches :

1. Développement de la SAP.

2. Génération des données GTFS.

Livrable :

Plusieurs versions de l'application qui doivent être validées par le MOA à fur et à mesure d'ajout de nouvelles fonctionnalités

Cycle de vie et contrainte temporelle :

Cette étape va se faire pendant la durée du PRD2.

2.5.1 Génération des données GTFS

Description de la tâche :

Cette étape consiste à la création des données GTFS de transport en commun pour la ville de Tours qui seront utilisées par le calculateur d'itinéraire.

Livrable :

Les différentes versions des données

Cycle de vie et contrainte temporelle :

Cette étape sera effectuée en parallèle avec le développement de l'application.

2.5.2 Développement de la SAP

Description de la tâche :

Cette tâche regroupe toutes les soutaches relatives au développement de la SAP.

Livrable :

Plusieurs versions de l'application qui doivent être validées par le MOA à fur et à mesure d'ajout de nouvelles fonctionnalités

Cycle de vie et contrainte temporelle :

Cette étape va se faire pendant la durée du PRD2.

Mise en place d'un prototype d'une application Angular JS

Description de la tâche :

Durant cette tâche je dois réaliser un premier prototype de l'application, à fin de comprendre et mettre en pratique les connaissances acquises en Angular JS

Livrable :

Une première version de l'application

Intégration des cartes interactives

Description de la tâche :

Cette tâche consiste à intégrer une librairie pour carte interactive dans l'application, sans oublier la mise à disposition des tuiles OSM pour l'affichage des cartes

Livrable :

Une version 2.0 de l'application

Intégration des différents éléments de l'interface de recherche

Description de la tâche : Cette tâche consiste à intégrer les différents éléments nécessaires à l'interface utilisateur pour pouvoir effectuer une recherche d'itinéraire.

Livrable :

Une version 3.0 de l'application

*Intégration d'un moteur d'auto complétion***Description de la tâche :**

Cette tâche consiste à intégrer un moteur d'autocomplétion pour la recherche d'un lieu de départ et d'arrivée

Livrable :

Une version 3.5 de l'application

*Intégration du Backend***Description de la tâche :**

Durant cette tâche on va intégrer un back-end baser sur le calculateur d'itinéraire Open Trip Planner

Livrable :

Une version 4.0 de l'application

*Affichage des résultats en textuelle***Description de la tâche :**

Cette tâche va nous permettre de retourner les résultats d'une requête d'itinéraire sous format textuelle.

Livrable :

Une version 4.1 de l'application

*Affichage des résultats sur la carte interactive***Description de la tâche :**

Cette tâche va nous permettre de retourner les résultats d'une requête d'itinéraire sur une carte interactive.

Livrable :

Une version 4.2 de l'application

*Affichage des résultats sur la page web***Description de la tâche :**

Cette tâche consiste à regrouper les résultats sous forme textuelle et sur carte dans l'application

Livrable :

Une version 5.0 de l'application

Phase de Test

Description de la tâche :

Cette phase correspond à la dernière étape avant la livraison du projet, durant cette étape il faudra que l'application puisse valider un certain nombre de tests afin de s'assurer le bon fonctionnement de toutes les fonctionnalités implémentées

Livrable :

l'application web avec tous les fichiers tiers nécessaires au fonctionnement de cette dernière.

Cycle de vie et contrainte temporelle :

La phase de test final ne peut se faire qu'à la fin de la phase du développement du backend et du frontend.

2.6 Préparation soutenance

Description de la tâche :

Cette tâche consiste à la préparation pour la présentation du travail réalisé pendant le PRD2

Livrable :

Slides de la présentation et le rapport en version finale.

3 Planning final

3.1 Méthodologie de gestion de projet

Le but de ce projet intitulé : Routage multimodal des personnes, est le développement d'une application Web en Angular JS 2 qui va permettre à son utilisateur de planifier un itinéraire d'un point "A" vers "B" avec un ou plusieurs modes de transport.

Le projet dispose de plusieurs guides de lancement et d'utilisation, ce qui va faciliter la reprise du projet. J'ai ajouté un fichier ReadMe.MD, qui se trouve dans la racine du projet. Il décrit en détail les étapes à suivre pour lancer l'application.

Une méthodologie "Agile" a été choisie comme méthode de gestion de projet. Tout d'abord, des rendez-vous hebdomadaires ont eu lieu avec mon encadrant, pour lui montrer l'état d'avancement du projet, ensuite des livrables fonctionnels de l'application ont dû être donnés durant le PRD2.

Pour la gestion du Versionning, le projet était sur Repository personnel de mon encadrant, ce qui facilitait le suivi au niveau du code de l'application.

"TexPortable" (un outil regroupant "MikTex", "TexMaker" et "SumatraPDF") est utilisé pour la création de ce rapport.

Plusieurs versions du rapport et du code sont sauvegardées sur le Repo git de mon encadrant.

3.2 Planning PRD 2

Le diagramme de gant est généré avec MSProject.

Nom de la tâche	Durée	Début	Fin	Prédéce
PR&D	142 jours	Mer 21/09/16	Jeu 06/04/17	
État de l'art et CDS	59 jours	Mer 21/09/16	Lun 12/12/16	
Documentation, lecture, compréhension	31 jours	Mer 21/09/16	Mer 02/11/16	
RDV : 1	0 jour	Mer 28/09/16	Mer 28/09/16	
RDV : 2	0 jour	Mer 12/10/16	Mer 12/10/16	
Outils liés au développement	31 jours	Mer 12/10/16	Mer 23/11/16	4
RDV : 3	0 jour	Mer 23/11/16	Mer 23/11/16	
Rédaction du CDS	53 jours	Mer 28/09/16	Ven 09/12/16	
Rédaction de l'état de l'art	53 jours	Mer 28/09/16	Ven 09/12/16	
Préparation soutenance	8 jours	Jeu 01/12/16	Dim 11/12/16	2;5
Soutenance	0 jour	Lun 12/12/16	Lun 12/12/16	9
PRD2	79 jours	Lun 19/12/16	Jeu 06/04/17	10
Développement de l'application	76 jours	Lun 19/12/16	Lun 03/04/17	
Prise en main d'Angular JS 2	35 jours	Lun 19/12/16	Ven 03/02/17	
Mise en place d'une Seed de projet	7 jours	Lun 06/02/17	Mar 14/02/17	13
Ajout d'une Weather API	3 jours	Mer 15/02/17	Ven 17/02/17	14
Intégration des différents éléments de l'interface de planification d'itinéraire	5 jours	Mer 15/02/17	Mar 21/02/17	14
Mise en place d'un planificateur avec Leaflet Routing Machine	7 jours	Mer 22/02/17	Jeu 02/03/17	16
Intégration d'un moteur d'auto complétion et reverse geocoding	10 jours	Mer 22/02/17	Mar 07/03/17	16
Mise en place d'une instance OTP	3 jours	Mer 08/03/17	Ven 10/03/17	18
Affichage des résultats sur la carte interactive	7 jours	Lun 13/03/17	Mar 21/03/17	19
Phase de Test et correction de bugs	3 jours	Mer 22/03/17	Ven 24/03/17	20
Affichage des résultats sur la page web	4 jours	Mar 28/03/17	Ven 31/03/17	21
Préparation soutenance	6 jours	Lun 27/03/17	Lun 03/04/17	
Soutenance	1 jour	Jeu 06/04/17	Jeu 06/04/17	23

Figure 3 – Diagramme de Gant final : S10

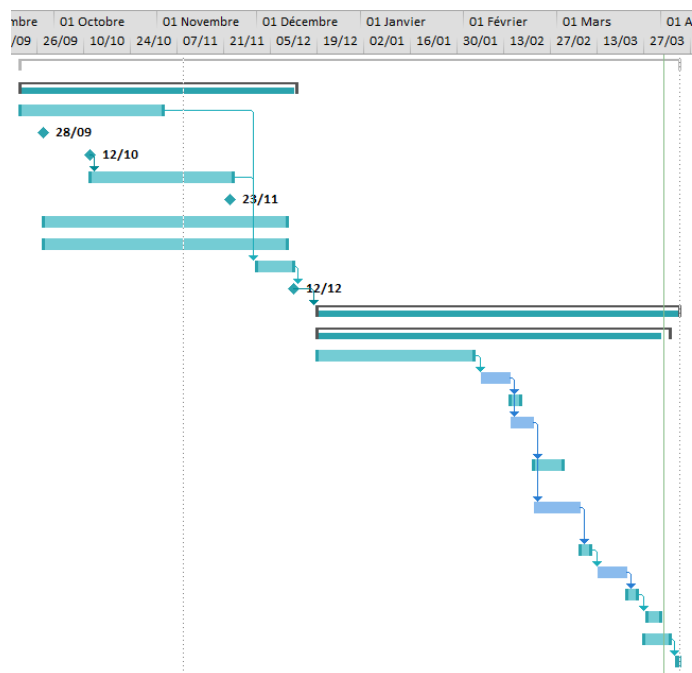


Figure 4 – Diagramme de Gant final : S10

De grandes modifications ont eu lieu dans la partie PRD2 :

- Comme j’ai trouvé des fichiers GTFS fournis par la RATP, il était inutile de générer ces fichiers.
- Implémentation de 2 planificateurs de trajet au lieu d’un seul.
- Utilisation d’Angular JS en version 2 et non la version 1.4.
- La prise en main du langage Type Script a pris beaucoup plus de temps que prévu (35jours).
- Je n’ai pas pu finir l’affichage de la feuille de route.
- J’ai utilisé Leaflet au lieu d’Open Layers, car les tutoriels de Leaflet étaient plus accessibles.
- L’envoi de la requête a pris plus de temps que prévu, suite au fait que je devais implémenter un service HTTP d’Angular 2.

4 Répartition des tâches

4.1 Développement de l’application :

Description de la tâche :

Cette tâche regroupe toutes les souches relatives au développement de mon application.

Livrable :

Plusieurs versions de l’application qui sont validées par le MOA à fur et à mesure d’ajout de nouvelles fonctionnalités.

Cycle de vie et contrainte temporelle :

Cette tâche s’est effectuée pendant toute la durée du PRD2.

Prise en main d’Angular JS 2

Description de la tâche :

Cette tâche consistait à l’apprentissage du langage Type Script. Ainsi que la mise en place de plusieurs “Seed” pour essayer de trouver la plus adaptée aux besoins du projet.

Livrable :

Aucun.

Mise en place d’une Seed du projet

Description de la tâche :

Durant cette tâche j’ai mis en place le squelette de mon application sur lequel j’ai implémenté ma solution de planification.

Livrable :

Une première version de l’application.

Ajout d’une Weather API

Description de la tâche :

Cette tâche consiste à intégrer un module qui représentait une application météo, ce qui m’a permis de mettre en pratique mes acquis concernant l’injection du Java Script avec un service Type Script.

Livrable :

Une version 0.5 de l’application.

*Intégration des différents éléments de l'interface de planification d'itinéraire***Description de la tâche :**

Cette tâche consistait à intégrer les différents éléments nécessaires à l'interface utilisateur pour pouvoir effectuer une recherche d'itinéraire :

- Un formulaire.
- Une Carte.

Livrable :

Une version 0.6 de l'application.

*Mise en place d'un planificateur avec Leaflet Routing Machine***Description de la tâche :**

Durant mes recherches j'ai découvert un outil très intéressant qui permet de planifier les itinéraires, après en avoir parlé avec mon encadrant, j'ai implémenté dans un nouveau module de mon application un nouveau planificateur basé sur Leaflet Routing Machine.

Livrable :

Une version 1.0 de l'application.

*Intégration d'un moteur de géocodage et reverse geocoding***Description de la tâche :**

Cette tâche consistait à intégrer un moteur d'autocomplétion pour la recherche d'un lieu de départ et d'arrivée, ce moteur permet aussi de faire un reverse geocoding pour trouver l'adresse d'un lieu à partir des coordonnées géographiques.

Livrable :

Une version 1.8 de l'application.

*Mise en place d'une instance d'Open Trip Planner***Description de la tâche :**

Durant cette tâche j'ai déployé une instance du calculateur Open Trip Planner sur un serveur.

Livrable :

Une version 1.8 de l'application.

*Affichage des résultats d'une requête sur la carte***Description de la tâche :**

Le but de cette tâche était de pouvoir afficher les résultats d'une requête d'itinéraire sur la carte.

Avant de pouvoir implémenter la méthode d'affichage d'un itinéraire sur la carte je devais tout d'abord ajouté un service http d'Angular JS 2 qui va s'occuper de faire l'envoi de la requête à mon Back-end.

Livrable :

Une version 2.5 de l'application.

*Phase de test et correction de bugs***Description de la tâche :**

Cette tâche consistait à commenter le code, reformater plusieurs méthodes ainsi que la correction de plusieurs bugs mineurs.

Livrable :

Une version 2.8 de l'application.

*Affichage des résultats sur la page web***Description de la tâche :**

Cette tâche consistait à l'affichage de la feuille de route dans l'application, malheureusement je n'ai pas pu finir totalement la méthode d'affichage.

Livrable :

Une version 3.0 de l'application.

4.2 Préparation soutenance**Description de la tâche :**

Cette tâche consiste à la préparation pour la présentation du travail réalisé pendant le PRD2

Livrable :

Slides de la présentation et le rapport en version finale.

5 Utilisation du git

J'ai utilisé le repo git personnel de mon encadrant mon adresse sur le repo est la suivante younes_21406448_git@makeitmap.com.

J'ai effectué actuellement plus de 65 commit sur le repo. Pour la gestion du versionning, j'ai utilisé l'outil gitKraken qui fournit une interface très intuitive.

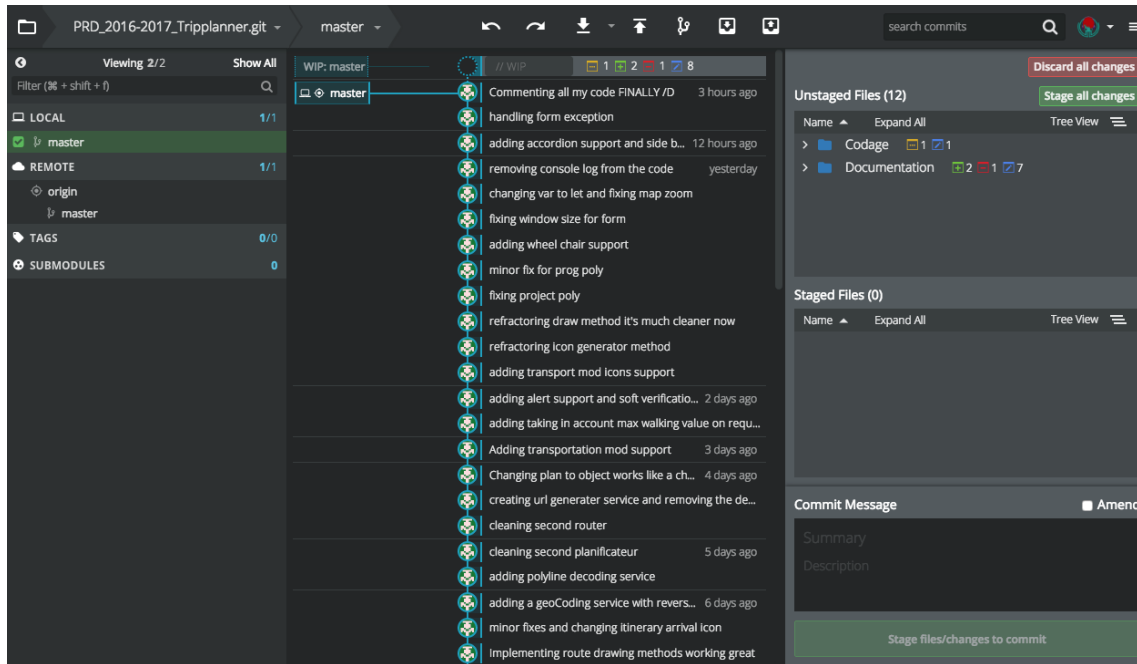


Figure 5 – Représentation du repo git

Comme je suis le seul à travaillé sur le projet, je n'ai pas eu besoin d'utiliser plusieurs branches tous mes commit ont été effectués sur la branche Master.

8

Guide d'installation et d'utilisation

1 Guide de lancement de l'application

Le lancement du projet nécessite :

- **NodeJs** en version 4.x.x ou supérieur.
- **Npm** en version 2.14.7 ou supérieur.

Lien de téléchargement pour NodeJs et Npm : <https://nodejs.org/en/download/>

Avant de pouvoir lancer le projet on doit installer les dépendances nécessaires.

Dans un terminal on exécute les commandes **Figure 1**.

```
# Prise en charge du typeScript :
$ sudo npm install -g typescript
$ sudo npm install -g typings
# Téléchargement du projet :
$ git clone https://gitlab.com/Uness77/Projet510-Angular2
# Installation des dépendances du projet:
$ npm install
# Lancement du projet:
$ npm start
# Configuration de base du serveur:
var PORT          = 5555;
var APP_BASE      = '/';
# Configuration au runtime:
npm start -- --port 8080 --reload-port 4000 --base /my-app/
```

Figure 1 – Commandes du lancement de l'application

Informations complémentaires:

Le projet est basé sur un Template open source SB Admin BS 4 Angular2 basé sur StartAngular et StrapUI.

Le modèle de base fournit les fonctionnalités suivantes :

- Avoir un build en mode "Prod" ou "Dev".
- Prendre en charge de la compilation "Ahead-of-Time".
- Tests de bout en bout avec "Protractor".
- Serveur de développement avec "Livereload".
- Gestionnaire des définitions de type en utilisant "@types".
- Utilisation de "Gulp" pour le "build" système.
- Css-lint.

Le projet dispose d'un fichier **README.md** qui contient plusieurs informations notamment les étapes à suivre pour lancer le projet et installer les dépendances.

2 Guide de lancement d'un serveur OTP

Pour lancer un serveur Open Trip Planner il nous faut :

- Les cartes **Open Street Map**.
- Les fichiers **Shapefile**.
- Optionnellement les fichiers **GTFS** de la zone choisie.

À l'intérieur de notre dossier parent, on va créer des dossiers en suivant la structure suivante **Figure 2** :

▼	graphs	hier à 22:16	--	Dossier
▼	centre	hier à 22:56	--	Dossier
▶	centre-latest-free.shp	hier à 22:23	--	Dossier
	centre-latest-free.shp.zip	hier à 12:03	302,5 Mo	Archiv...hier
	centre-latest.osm.pbf	hier à 22:19	155 Mo	Document
	Graph.obj	hier à 23:00	160,6 Mo	Geom...Form
▼	idf	hier à 22:27	--	Dossier
	Graph.obj	hier à 22:55	827,6 Mo	Geom...Form
	gtfs-paris.zip	hier à 11:39	42,1 Mo	Archiv...hier
	ile-de-france-latest.osm.pbf	hier à 22:25	234,7 Mo	Document
	ile-de-france-latest.shp.zip	hier à 11:39	70,5 Mo	Archiv...hier
	otp-0.19.0-shaded.jar	hier à 11:54	60,3 Mo	Fichier...R Je

Figure 2 – Structure des fichiers du serveur OTP

- Le ".jar" d'Open Trip Planner en version 0.19.0.
- Un dossier "graph" qui va contenir 2 sous dossiers :
 - centre :
 - Le fichier "pbf" d'Open Street Map pour la zone centre.
 - Le fichier Shape file de la zone centre dé zipper.
 - idf :
 - Le fichier "pbf" d'Open Street Map pour la zone ile de France.
 - Le fichier Shape file qui lui correspond zipper.
 - Les fichiers GTFS dans un seul zip.

Notre serveur OTP va pouvoir faire des calculs d'itinéraires sur la zone Centre et ile de France. Une fois qu'on a tous nos fichiers, on peut lancer la génération du Graph.obj pour chaque zone.

On ouvre un terminal au niveau de notre dossier parent et on tape les commandes suivantes :

```
# pour générer le Graph.obj pour la zone IDF :
$ java -Xmx3g -jar otp-0.19.0-shaded.jar --basePath . --build graphs/idf
# pour générer le Graph.obj pour la zone Centre :
$ java -Xmx3g -jar otp-0.19.0-shaded.jar --basePath . --build graphs/centre
```

Le paramètre `-Xmx3G` nous permet de définir la limite de consommation en mémoire (dans ce cas 3G => 3Go). La génération du fichier `Graph.obj` peut prendre plusieurs minutes, l'image suivante montre un exemple du temps nécessaire pour la génération du graphe pour la zone IDF. Maintenant qu'on a nos fichiers `graph.obj`, on peut lancer le serveur avec la commande suivante :

```
# commande pour lancer le serveur :
$ java -Xmx2g -jar otp-0.19.0-shaded.jar --basePath . --autoScan --server
```

Il faut patienter un peu le temps que notre instance OTP se lance. Une fois le serveur lancé on peut accéder à notre instance Open Trip Planner à partir du navigateur web avec l'adresse suivante : `localhost:8080`.

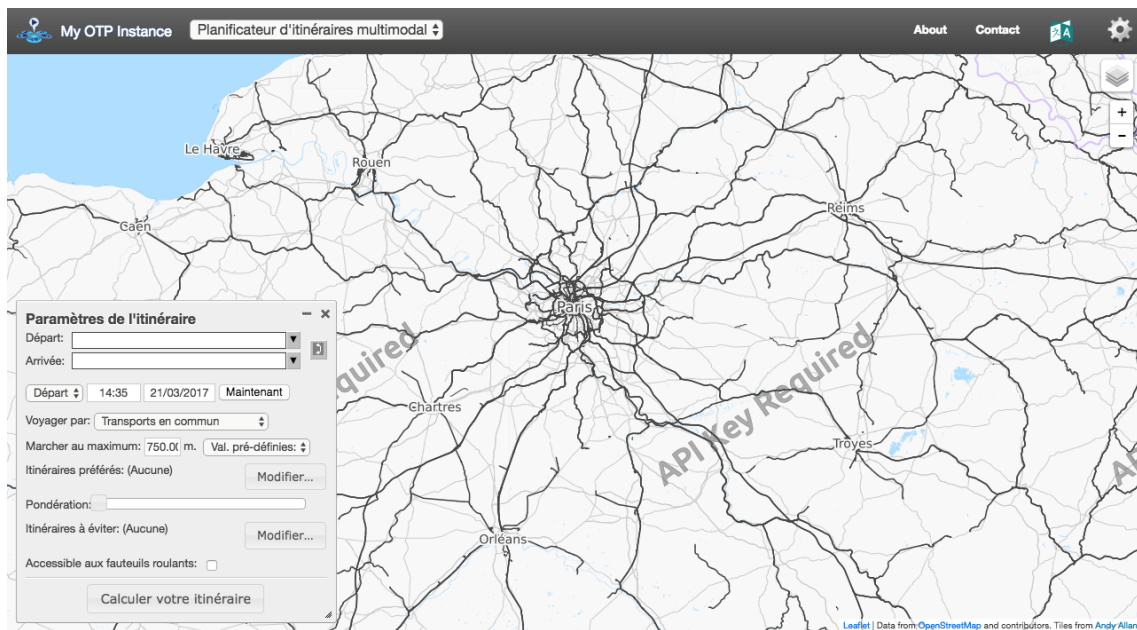


Figure 3 – Exemple d'une instance OTP

Liens de téléchargement des fichiers :

Le lien suivant nous permet de télécharger OTP en version 0.19 : <http://maven.conveyal.com.s3.amazonaws.com/org/opentripplanner/otp/0.19.0/otp-0.19.0-shaded.jar>

On peut trouver dans ce lien les fichiers OSM et Shape File pour la zone souhaitée : <http://download.geofabrik.de/europe/france/centre.html>

Pour les fichiers GTFS de la ville de paris : <https://www.data.gouv.fr/fr/datasets/offre-transport-de-la-ratp-format-gtfs-ratp/>

Ce lien contient des informations complémentaires sur comment lancer un serveur OTP : <http://www.oodlestechnologies.com/blogs/How-to-set-up-Open-Trip-Planner-Server>

3 Guide d'utilisation de l'application

Dans ce chapitre je vais décrire comment on peut utiliser l'application. L'interface a été conçue de manière intuitive et ne nécessite pas des connaissances en informatique.

3.1 Point d'entrée de l'application

Une fois l'application lancée, le navigateur s'ouvre automatiquement avec une page à l'adresse suivante : <http://localhost:5555/dashboard/entryProgram>

N.B. Le port 5555 est le port par défaut que j'ai paramétré pour l'application.

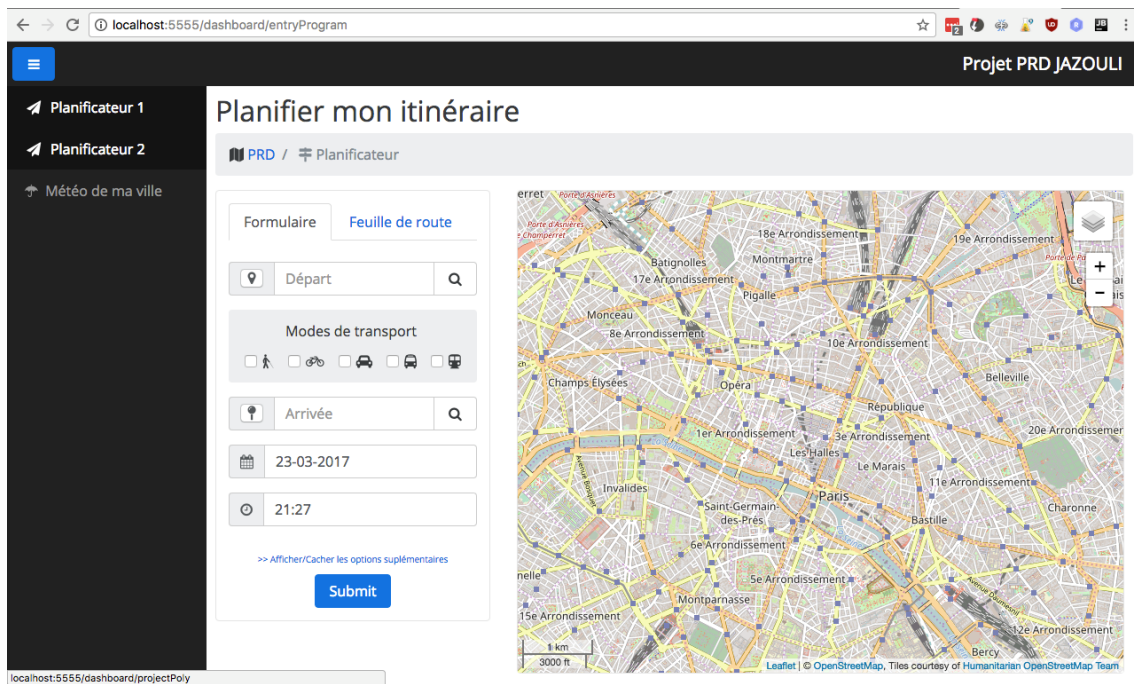


Figure 4 – Image représentant le point d'entrée de l'application

J'ai implémenté 3 modules :

- Planificateur 1 : représente une solution de planification d'itinéraire multimodale en utilisant un back-end basé sur le calculateur Open source : Open Trip Planner.
- Planificateur 2 : représente une solution de planification d'itinéraire basée sur la librairie Leaflet Routing Machine qui permet de planifier un itinéraire en voiture en se basant sur le plus court chemin.
- Météo de ma ville : représente une api météo qui permet comme son nom l'indique de donner des informations sur la météo.

3.1.1 Planificateur 1

Au niveau de l'interface du planificateur 1 on peut constater 2 choses :

Le Formulaire :

Le formulaire permet à l'utilisateur de renseigner les informations nécessaires pour effectuer une requête d'itinéraire il se compose des éléments suivants :

Formulaire **Feuille de route**

Départ **Arrivée**

Modes de transport

23-03-2017

21:42

[>> Afficher/Cacher les options supplémentaires](#)

Submit

Bouton pour géo localiser sa position comme point de départ ou d'arrivée

Recherché une adresse comme point de départ

Choix du mode de transport : A pied ; Vélo ; Voiture ; Bus ; Tram & Métro
L'utilisateur peut sélectionner autant de mode qu'il veut

Recherché une adresse comme point d'arrivée

Sélectionner une date départ

Sélectionner l'heure de départ

Afficher cacher les options supplémentaires

Soumettre la requête et afficher l'itinéraire

Figure 5 – Image représentant les éléments du formulaire du planificateur 1

Les options supplémentaires sont :

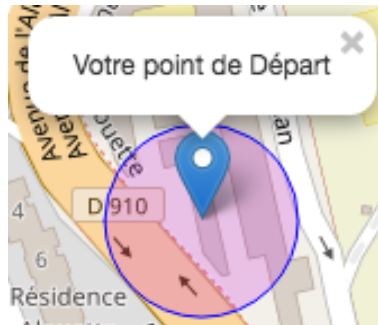
☐ ☐ Max Walking Distance (m)

☐ ☐ Wheel Chair

- Définir la valeur de la distance maximale que l'utilisateur veut parcourir à pied avec une unité de "mètre".
- Prendre en compte des passages pour les personnes à mobilité réduite lors de la recherche d'itinéraire.

Le point de départ et d'arrivée :

Les points "Départ et Arrivée" par saisie d'adresse disposent d'un système d'autocomplétion qui permet à l'utilisateur de sélectionner parmi les résultats retournés. Le point de départ ou d'arrivée est mis à jour à chaque fois qu'il fait une saisie.



Si par exemple l'utilisateur décide de faire de sa position un point de départ il n'a qu'à cliquer sur le bouton de géo localisation qui se trouve au niveau de la zone de saisie du point de départ, l'adresse de l'endroit où il se trouve va automatiquement remplir le champ Départ et un marqueur de départ va se mettre automatiquement sur la carte.

Modes de transport :

Les moyens de transport supportés sont :

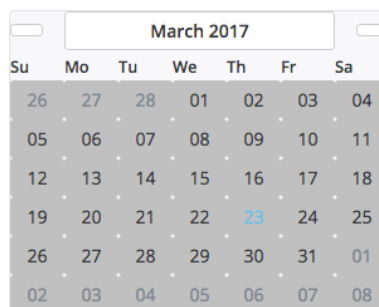
- À Pied
- Vélo
- Voiture
- Bus
- Tram
- RER
- Métro

Les modes de transports sont représentés par des check-Box. L'utilisateur doit sélectionner au minimum un seul mode de transport.

Le calculateur d'itinéraire supporte tous les modes de transport. Si l'utilisateur coche toutes les cases, le résultat de la requête d'itinéraire sera basé sur le trajet le plus rapide avec un ou plusieurs moyens de transport !

Date et Heure de départ :

La date de départ est par défaut la date actuelle, l'heure de départ représente l'heure actuelle. L'utilisateur peut changer la date et l'heure. Avec un simple clic sur la zone de saisie de la date, un calendrier va apparaître il suffira à l'utilisateur de cliquer sur la date qu'il souhaite.



La modification de l'heure se fait de la même façon. Il suffit de faire un clic sur la zone d'affichage de l'heure pour faire apparaître une fenêtre avec une zone de modification de l'heure et une zone pour les minutes.

N.B. Si l'utilisateur sélectionne une heure invalide, par exemple 28 : 90 l'heure va se remettre automatiquement à l'heure actuelle.

Options supplémentaires :

- **Max Walk Distance** : L'utilisateur peut définir la distance maximale qu'il veut parcourir à pied. Cette distance représente la somme des distances qu'il va parcourir pour aller d'un arrêt à un autre. L'unité de cette distance est le Mètre. La valeur par défaut de cette distance est définie à 750m. N.B. Si l'utilisateur décide de faire une requête d'itinéraire uniquement à pied la valeur du Max Walk Distance n'est pas prise en compte.
- **Wheel Chair** : Si le check box "Wheel Chair" est coché cela permet d'indiquer au niveau de la requête d'itinéraire qu'on souhaite prendre en compte des chemins qui dispose de passage adapté aux personnes à mobilité réduite.

La carte :

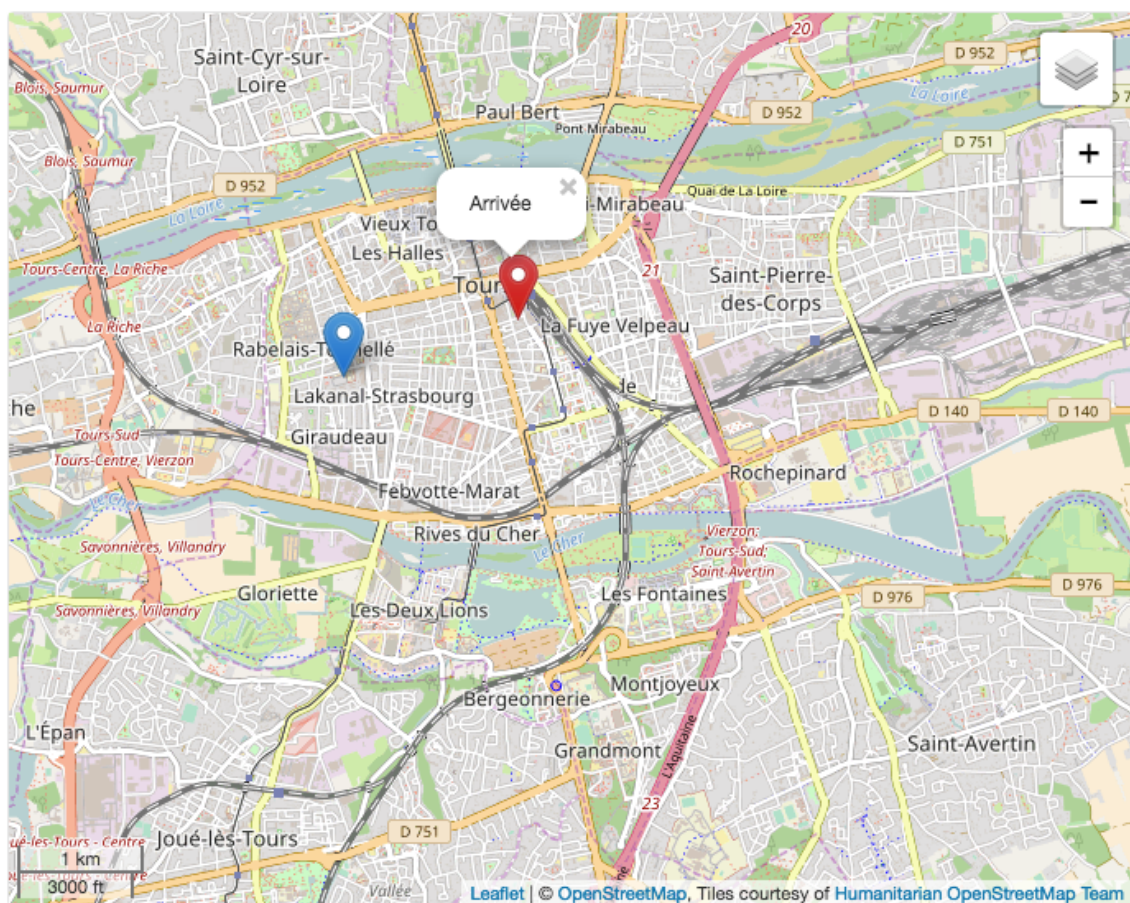
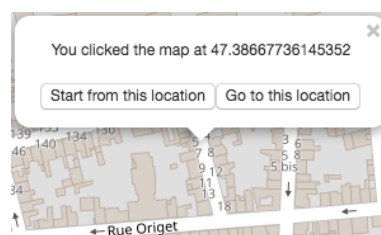


Figure 6 – Image représentant la carte du planificateur 1

La carte va nous permettre d'afficher nos itinéraires, définir notre point de départ ou d'arrivée ...

Définir un point de départ ou d'arrivée :



Si l'utilisateur fait un clic sur un endroit de la carte un pop-up va apparaître au niveau de l'endroit où l'utilisateur a cliqué, grâce à ce dernier il peut faire de cet endroit son point de départ ou d'arrivée.

- Le bouton "Start from this location" : permet de définir l'endroit comme un point de départ.
- Le bouton "Go to this location" : permet de définir l'endroit comme un point d'arrivée. Un marqueur se mettra automatiquement si l'utilisateur a choisi le point comme point de départ ou d'arrivée.

Les marqueurs : Le marqueur bleu représente le point de départ et le marqueur rouge représente le point d'arrivée.

Si l'utilisateur clique sur un marqueur, un pop-up apparaîtra avec un message qui est prédéfini selon le marqueur sur lequel il a cliqué.

Layers : On peut changer la vue de la carte selon des modèles de cartographie avec un simple clic. On peut choisir entre 3 types de fournisseurs de carte :

- Open Street Map
- CartoDB
- Esri



Figure 7 – Image représentant les layers de la carte

Zoom :

L'utilisateur peut zoomer sur la carte pour voir les rues plus en détail, la valeur par défaut du zoom est de 13 et la valeur maximum est de 18. La valeur minimum est de 5.

Affichage de l'itinéraire :

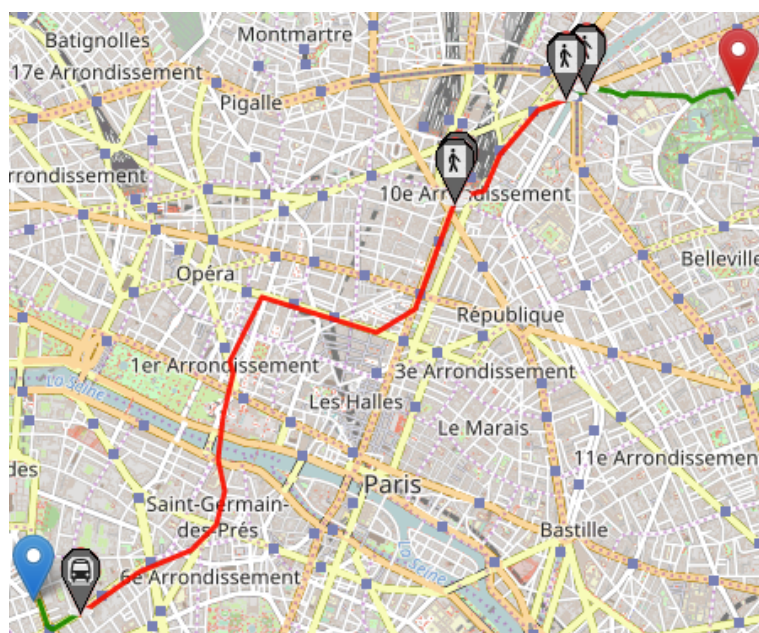


Figure 8 – Image représentant un itinéraire affiché la carte

Le rôle le plus important de la carte et l'affichage de l'itinéraire retourné par le calculateur. La couleur du trajet dépend du mode de transport utilisé. Par exemple dans l'image **Figure 8** on peut voir un trajet planifier avec 3 modes de transport : Vélo, Bus, à pied.

- Les trajets à vélo sont de couleur verte.
- Les trajets en bus sont de couleur rouge.
- Ceux à pied sont de couleur noire.

3.1.2 Planificateur 2

Le module "planificateur 2" représente une solution de planification de trajet en voiture.

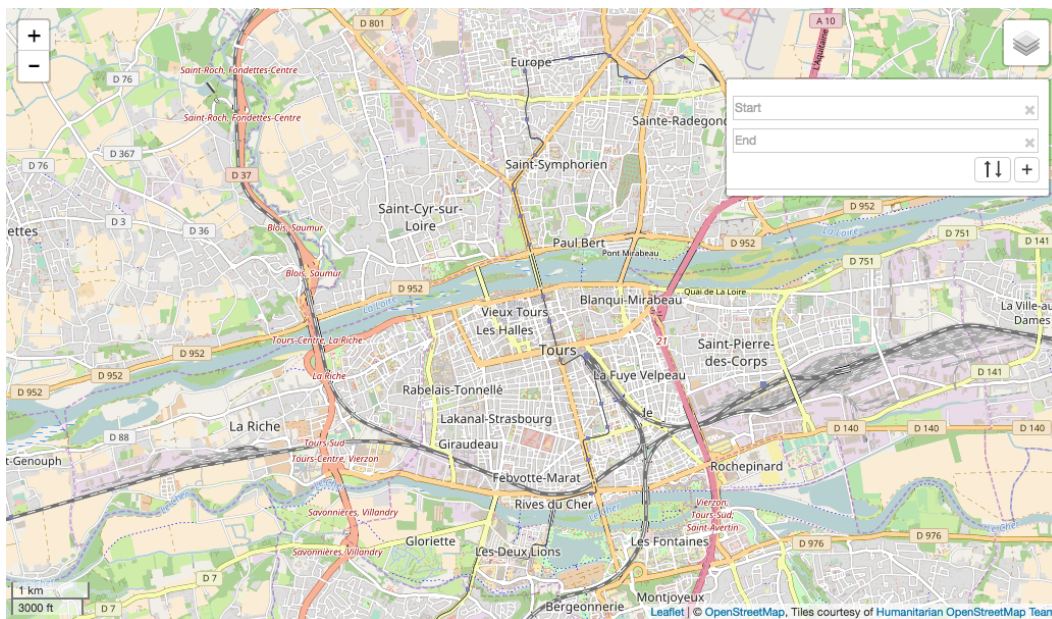


Figure 9 – Image représentant l'interface du planificateur 2

La carte dispose des mêmes fonctionnalités que la carte du planificateur 1 :

- Définir un point de départ ou d'arrivée en cliquant sur la Map.
- Zoomer / dézoomer.
- Changer de "layer" : On peut choisir entre Esri, Carto DB, Open Street Map.
- Afficher un itinéraire.

Le formulaire se trouve au niveau de la carte, on peut définir :



Figure 10 – Image représentant les éléments du formulaire du planificateur 2

- Un point de départ.
- Un point d'arrivée.
- Un ou plusieurs points intermédiaires (7 points intermédiaires au maximum).
- Inverser le point de départ et d'arrivée.

Les points 'Start, End et Via' disposent d'un système d'auto complétion pour la saisie d'adresse.

Si l'utilisateur fait un clic sur un endroit de la carte un popup va apparaître au niveau de l'endroit où l'utilisateur a cliqué, grâce à ce dernier l'utilisateur peut faire de cet endroit son point de départ ou d'arrivée.

La sélection d'un point de départ ou d'arrivée va automatiquement mettre un marqueur sur la carte. Une fois qu'on a un point de départ et d'arrivée qui est défini, l'itinéraire s'affiche sur la carte et la feuille de route s'affichera en bas du formulaire.

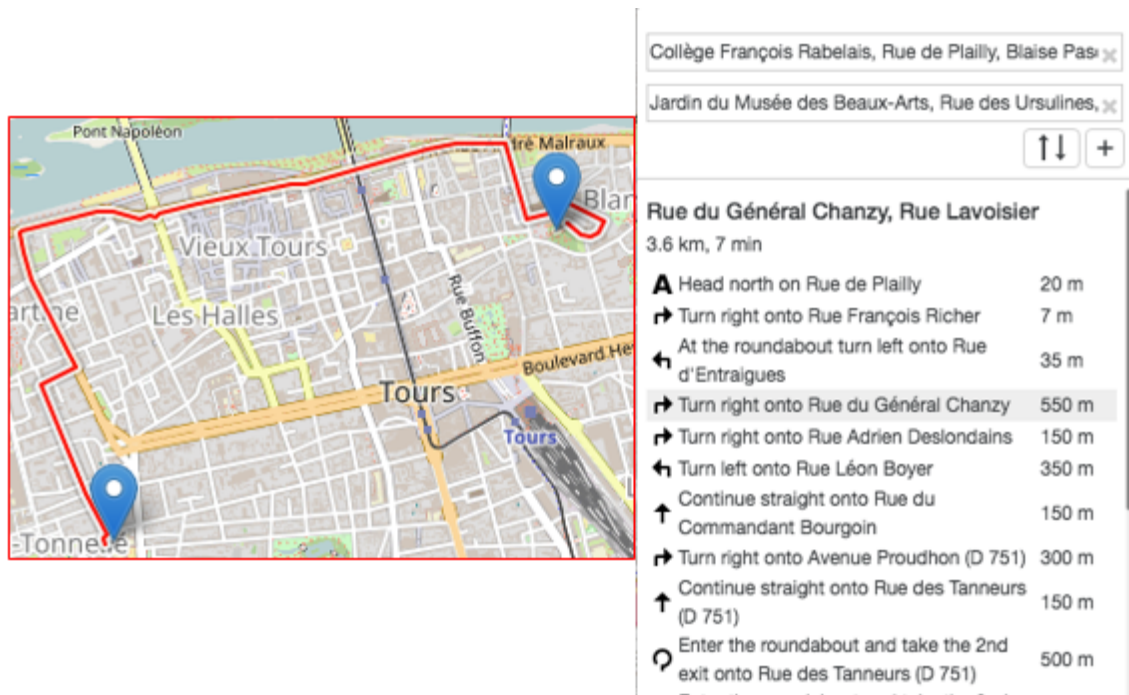


Figure 11 – Image représentant l'affichage d'un itinéraire par le planificateur 2

On retrouve dans la feuille de route la durée totale de l'itinéraire ainsi que la distance totale. Puis les instructions que l'utilisateur doit suivre pour mener à bien son trajet, avec la distance de chaque étape.

3.1.3 Météo de ma ville

Dans ce module, l'utilisateur peut retrouver des informations relatives à la météo de la ville ou il se trouve.



Figure 12 – Image représentant l'affichage du module météo

On retrouve parmi ces informations :

- Le taux d'humidité
- Prévisions météorologiques de la semaine.
- Les prévisions météorologiques en vidéo.

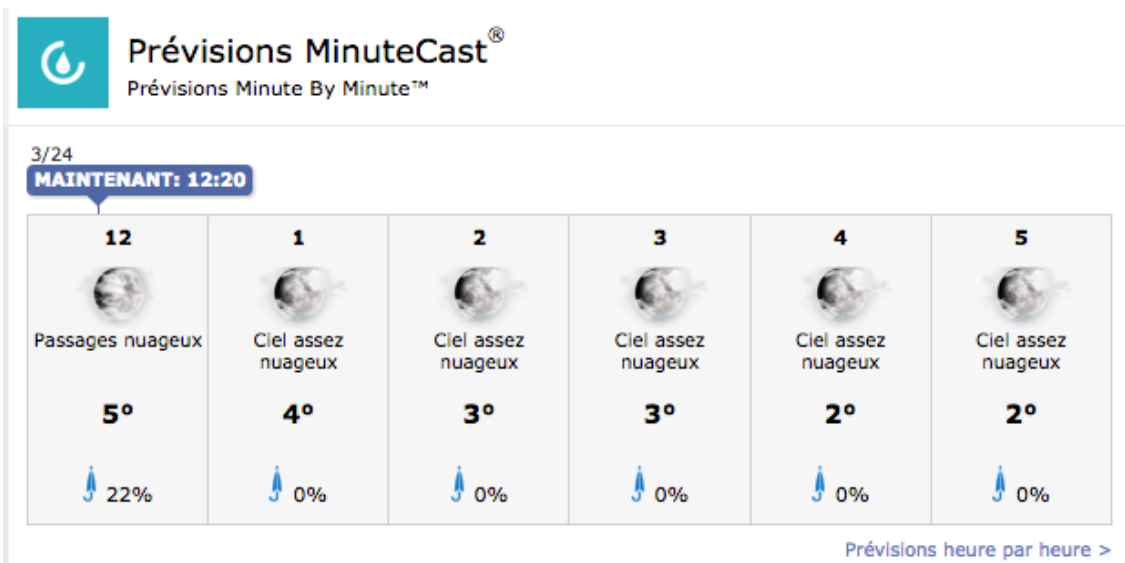


Figure 13 – Image représentant les prévisions MinuteCast

Ce module se base sur l'API Accuweather.

Quatrième partie

Bibliographie

Sébastien, Olivier. et Pierre Alexandre, GURY. (2015). "Développez aujourd'hui les applications web de demain". Edition ENI. 366 p. - (Collection Expert IT, ISSN 1958-9913).

Gaël, SAUVANET. (2011). "Recherche de chemins multiobjectifs pour la conception et la réalisation d'une centrale de mobilité destinées aux cyclistes".Tours : SCD de l'université de Tours. 144 p. - (Numéro thèse 2011TOUR4006).

Cinquième partie

Webographie

Auteur : Agus

Titre : librairies JS

Année : 2014

URL : <http://www.hongkiat.com/blog/javascript-libraries-for-interactive-maps/>

Langue : Anglais

Organisation : hongkiat

Date de visite : 2016-10

Auteur : Iweczek

Titre : librairies JS2

Année : 2014

URL : <http://techslides.com/50-javascript-libraries-and-plugins-for-maps>

Langue : Anglais

Organisation : techslides

Date de visite : 2016-10

Auteur : Openlayers

Titre : Openlayers

Année : 2015

URL : <http://openlayers.org/en/latest/apidoc/>

Langue : Anglais

Organisation : OpenLayers

Date de visite : 2016-10

Auteur : Vladimir Agafonkin

Titre : Leaflet

Année : 2015

URL : <http://leafletjs.com/reference-1.0.2.html>

Langue : Anglais

Organisation : leaflet

Date de visite : 2016-10

Auteur : Alex Ivanovs

Titre : Framework JS

Année : 2016

URL : <https://colorlib.com/wp/javascript-frameworks/>

Langue : Anglais

Organisation : colorlib

Date de visite : 2016-10

Auteur : Kurt Jansson Titre : ComparatifJS

Année : 2016

URL : <https://en.wikipedia.org/wiki/Comparison-of-JavaScript-frameworks>

Langue : Anglais

Organisation : wikipedia

Date de visite : 2016-11

Auteur : opentripplanner

Titre : OTP

Année : 2016

URL : <http://docs.opentripplanner.org/en/latest/>

Langue : Anglais

Organisation : opentripplanner

Date de visite : 2016-10

Auteur : Pautard

Titre : OSM

Année : 2016

URL : <https://fr.wikipedia.org/wiki/OpenStreetMap>

Langue : Français

Organisation : wikipedia

Date de visite : 2016-10

Comptes rendus hebdomadaires

Compte rendu n°1 du 26/09/2016

Tâches effectuées :

- Lecture des principaux chapitres (2 ; 3 ; 5) de la thèse de Mr SAUVANET.
- Début de l'état de l'art :
 - * Faire un comparatif des principales librairies JAVA script pour carte interactive (Leaflet, Open Layers, GeoComplete, Gmaps, Maplace).
 - * Lister les informations relatives aux calculateurs d'itinéraires (Open Trip Planner (avec son architecture), NAVITIA).
 - * Lister les informations sur Open Street Map.

Tâches à faire :

- Détailler le fonctionnement des Algorithmes de recherche utilisés par les calculateurs d'itinéraires (A*, Dijkstra, Raptor, méthodes hiérarchiques).
- Développer plus la comparaison entre NAVITIA et OTP, rechercher plus de calculateurs d'itinéraire.
- Lister les informations relatives à chaque licence.
- Détailler les informations sur les frameworks qu'on peut utiliser pour le développement de l'API : AngularJS, NodeJS... .

Difficultés :

- Aucunes.

Compte rendu n°2 du 29/09/2016

Tâches effectuées :

- Faire des recherches sur les Framework JS puis faire un comparatif.
- État de l'art :
 - * Tuto prise en main d'OTP.
 - * Descriptif Open Street MAP vs Google Maps.

* Détailler les informations sur les frameworks qu'on peut utiliser pour le développement de l'API : AngularJS, NodeJS... .

* Lister les informations relatives à chaque licence.

Tâches à faire :

- Me documenter sur les SAP Simple App Page.

- Faire plus des recherches plus détaillées pour enrichir les informations de l'état de l'art.

Difficultés :

- Aucunes.

Compte rendu n°3 du 06/10/2016

Tâches effectuées :

- Faire des recherches sur les SAP en plus d'une documentation sur AngularJS et sa prise en main.

- État de l'art :

Définir une SAP faire une comparaison entre une SAP et un Site Web.

Tâches à faire :

- Commencer le CDS

- Se documenter plus sur Angular JS

Difficultés : - Aucunes.

Compte rendu n°4 du 13/10/2016

Tâches effectuées :

- Lecture des principaux chapitres du livre : AngularJS : Développez aujourd'hui les applications web de demain. Auteurs : Sébastien Ollivier, Pierre-Alexandre GURY.

- J'ai commencé la rédaction du CDS pour les éléments basiques comme l'introduction, contexte

Tâches à faire : - Avancer sur le contenu du CDS essayer de finir la version 1

- Continuer la lecture du livre sur AngularJS Développez aujourd'hui les applications web de demain.

Difficultés : - Aucunes.

Compte rendu n°5 du 20/10/2016

Tâches effectuées : - Avancer sur le contenu du CDS essayer de finir la version 1.

- Continuer la lecture du livre sur AngularJS Développez aujourd'hui les applications web de demain.

- Définir comment peut gérer la prise en main d'une SAP par plusieurs personnes.

Tâches à faire : - CDS version 2.

- MAJ de l'état de l'art.

Difficultés : - Aucunes.

Compte rendu n°6 du 27/10/2016

Tâches effectuées :

- Mise à jour du CDS Version 2 (bases méthodologiques, environnement du projet, performances.)

Tâches à faire :

- Commencer le diagramme de Gant prévisionnel pour le projet.
- Écrire les parties manquantes du CDS (fonctionnalités, Ajout des schémas)

Difficultés : – Aucunes.

Compte rendu n°7 du 04/11/2016

Tâches effectuées : - Maj des spécifications fonctionnelles et non fonctionnelles.

- Ajout des schémas.
- Maj du diagramme de cas d'utilisation.

Tâches à faire : - Créer un sketch de l'API.

- Maj la version 2 du CDS.

Difficultés : – Aucunes.

Compte rendu n°8 du 10/11/2016

Tâches effectuées : - Maj du CDS vers la version 2 ajout de correctif mineur.

- Sketech de l'API.

Tâches à faire : - Passer le CDS en version 3.

- Continuer la lecture du livre sur AngularJS Développez aujourd'hui les applications web de demain.

Difficultés : – Aucunes.

Compte rendu n°9 du 17/11/2016

Tâches effectuées :

- Maj du CDS en version 3

Tâches à faire :

- Correction du CDS
- Faire le diagramme de Gant

Difficultés :

– Aucunes.

Compte rendu n°10 du 24/11/2016

Tâches effectuées :

- Mise à jour des tableaux comparatifs pour librairies JS de carte interactive.
- Correction des parties du CDS suivant les remarques du Pr Soukhal.
- Diagramme de Gant prévisionnel du projet

Tâches à faire : - Mettre l'état de l'art sous format LaTeX.

- Ajout du plan de développement dans le CDS.

Difficultés : – Aucunes.

Compte rendu n°11 du 01/12/2016

Tâches effectuées : - Mettre le CDS et l'état de l'art dans le même rapport sous format LaTeX

- Rédiger le plan de développement
- Mettre à jour le diagramme de Gant

Tâches à faire : - Finir la partie Analyse

- Ajouter une conclusion
- Ajouter les Posters
- Ajouter les rapports Hebdo dans le rapport final

Difficultés : – Aucunes.

Compte rendu n°12 du 08/12/2016

Tâches effectuées :

- Mettre à jour du rapport (Analyse, Conclusion, Posters).

Tâches à faire :

- Préparer la soutenance.

Difficultés :

– Aucunes.

Compte rendu n°13 du 05/01/2017

Tâches effectuées :

- Génération de la structure du projet Angular JS 2 avec YEOMAN Installation des dépendances nécessaires Lancer un modèle de l'application sur un serveur local

Tâches à faire :

- Comprendre la notion du Data binding pour Angular JS 2 - Commencer l'implémentation des premiers éléments de l'interface web

Difficultés : – Aucunes.

Compte rendu n°14 du 12/01/2017

Tâches effectuées :

- Commencer l'implémentation des premiers éléments de l'interface web.
- Utilisation d'un Template pour avoir les éléments de base.
- Intégrer une interface pour une map OpenLayers 3.

Tâches à faire :

- Mettre en place les éléments de l'interface.
- Nettoyer le Template des éléments inutilisable.

Difficultés :

– Aucunes.

Compte rendu n°15 du 19/01/2017

Tâches effectuées :

Implémentation de quelques éléments de l'interface. Ajout de nouveaux éléments dans la Side Bar Structurer le code.

Tâches à faire :

- Compléter l'implémentation des principaux éléments de l'interface.

Difficultés :

– Mise en place d'un routage en 2 child module du même module parent.

Compte rendu n°16 du 26/01/2017

Tâches effectuées :

Ajout d'un DatePicker. Correction de Bug et nettoyage du code.

Tâches à faire :

- Implémenter le JS pour les points de départ intermédiaires. - Finaliser les éléments du formulaire de recherche d'itinéraire - Ajout d'un module d'autocomplétion - Affichage de la Map OSM au niveau du HomeProgram

Difficultés :

– Structuration des sous-modules.

Compte rendu n°17 du 02/02/2017

Tâches effectuées :

J'ai finalisé tous les éléments du planificateur d'itinéraire. J'ai changé un peu la structure du projet et j'ai retiré des éléments inutilisables du Template. J'ai affiché une map avec Openlayers.

Tâches à faire :

Le module d'autocomplétion n'est pas encore implémenté, mon encadrant travaille sur son propre module que je dois intégrer dans le projet prochainement. Essayer de faire un prototype d'un itinéraire

Difficultés :

– Je ne trouve pas des tutoriels qui facilitent la prise en main pour tracer une route, je vais voir avec mon encadrant la possibilité de changer de Framework de cartes interactives pour passer sous Leaflet.

Compte rendu n°18 du 09/02/2017

Tâches effectuées :

Amélioration mineure sur le code du projet. J'ai remplacé Open Layers par Leaflet dans le HomeProgram. Effectuer des recherches sur le traçage des routes avec leaflet.

Tâches à faire :

- Suite à mes recherches sur internet j'ai trouvé un Framework "leaflet routing machine" qui permet de planifier les itinéraires est qui est facile à prendre en main. Je vais essayer d'ajouter un nouveau module dans le projet dans lequel je vais implémenter la solution basée sur leaflet routing machine.

Difficultés :

– J'ai trouvé beaucoup de difficulté à essayer de faire marche OTP avec une URL prototype, je vais plutôt essayer d'implémenter la solution basée sur leaflet routing machine avant de reprendre le travail sur Open Trip Planner.

Compte rendu n°19 du 16/02/2017

Tâches effectuées :

Implémentation de quelques éléments de l'interface. Implémentation d'un exemple d'utilisation de Leaflet routing machine avec uniquement un itinéraire. J'ai changé la façon avec laquelle mon weather service fait appel à mon JS (j'ai trouvé une façon très propre au niveau du code)

Tâches à faire : - Suite à la réunion avec mon encadrant, je vais continuer de travailler sur Leaflet Routing Machine. - J'ai eu accès à son service web qu'il a développé pour l'auto complétion des lieux donc je dois essayer de l'intégrer la semaine prochaine

Difficultés : – Prise en main du format TypeScript.

Compte rendu n°20 du 23/02/2017

Tâches effectuées :

Intégration d'une autocomplétion pour le point de départ et d'arrivée avec Nominatim.

Intégration d'un module dans le projet avec Leaflet Routing Machine qui permet de faire de retourner :

o Un itinéraire.

o Plusieurs points intermédiaires.

o Inverser le point de départ/arrivée.

o Prise en charge d'un nouveau point de départ/arrivée avec un clic sur la Map.

J'ai changé la façon avec laquelle le routage des modules se fait ce qui m'a permis de retirer des index.ts et de faire appel directement aux composants

Tâches à faire :

- Améliorer le temps de réponse pour l'auto complétion - Afficher le point départ/arrivée sur la map à partir des lieux sélectionner sur l'input

Compte rendu n°21 du 02/03/2017

Tâches effectuées :

Intégration d'une fonction pour dessiner les marqueurs de départ et d'arrivée.

Intégration du reverse Géocoding.

Correction de Bugs.

Ajout d'un point de départ ou d'arrivée à partir de la map.

Ajout de la géolocalisation de l'utilisateur pour le point de départ ou d'arrivée tout en prenant compte du reverse geocoding sur l'input.

Tâches à faire :

- Déployer un serveur OTP auquel je vais envoyer mes requêtes.

- Prendre en compte le drag et drop pour les marqueurs de départ et d'arrivée.

- Je vais essayer de reformater ma méthode drawMarker().

Difficultés :

- Rien de particulier en cas de problème je post sur stackoverflow mon problème.

Compte rendu n°22 du 09/03/2017

Tâches effectuées :

Déployer un serveur OTP auquel je vais envoyer mes requêtes.

Reformater ma méthode drawMarker().

J'ai commencé à coder mon service qui va faire la requête d'itinéraire

Tâches à faire : - Finir mon request Service - Trouver comment faire pour afficher un itinéraire sur la carte Difficultés :

- Aucunes

Compte rendu n°23 du 16/03/2017

Tâches effectuées :

Mon Request service marche.

J'ai réussi à afficher un itinéraire prédéfini sur la carte.

Tâches à faire :

- Prise en compte des moyens de transport dans le Request - Gestion des modes de transport avec des marqueurs dans l'affichage de l'itinéraire.

Difficultés : - Problème de callback avec le Json object.

Compte rendu n°24 du 23/03/2017

Tâches effectuées :

Ma fonction d'affichage d'itinéraire marche sans problème.

J'ai rectifié beaucoup de bugs qui étaient dans le code. Affichage d'une alerte si le formulaire de requête est vide.

Tâches à faire :

- Faire un guide de développement et d'utilisation. - Faire un guide pour le lancement du serveur OTP et de l'application client. Difficultés :

- aucunes

Compte rendu n°25 du 30/03/2017

Tâches effectuées :

- Faire un guide de développement et d'utilisation.

- Faire un guide pour le lancement du serveur OTP et de l'application client.

Tâches à faire :

- Commenter le code.

- Essayer de finir la méthode d'affichage de la feuille de route.

- Commencer à rédiger le rapport de PRD 2.

Difficultés :

- aucunes

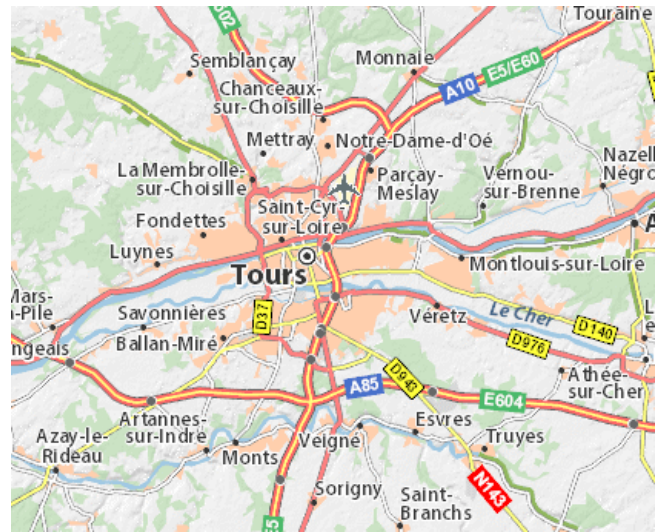
Routage multimodal des personnes

Younes JAZOULI BENLAHBOUB

Encadrement : Ismaïla Abderhamane NDIAYE

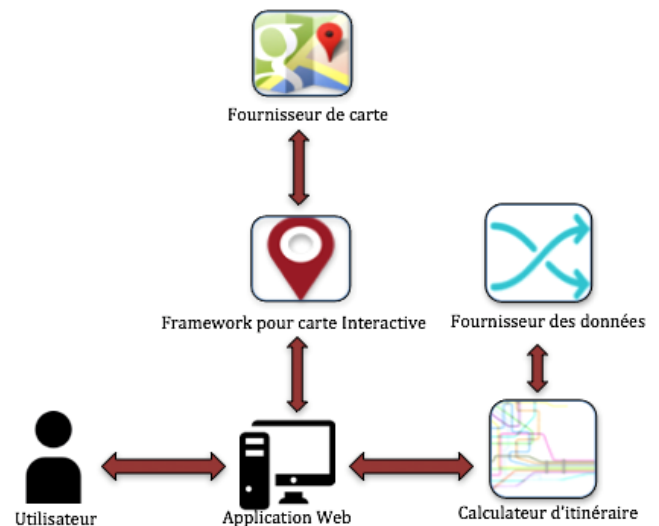
Constat

Lors d'un trajet un voyageur doit souvent emprunter un à plusieurs modes de transports (voiture, tram, métro ...) exploités par plusieurs transporteurs. Les solutions existantes peuvent ne pas être satisfaisantes, car elle ne supporte pas entièrement cette multimodalité.



Solution

Développer une application Web avec un Backend basé sur un calculateur d'itinéraire optimisé qui va nous permettre de faciliter cette multimodalité au niveau des correspondances entre modes et réseaux de transports.



Conclusion

Grâce à la mise en place de plusieurs outils technologiques, on peut développer une application qui répond parfaitement au besoin de l'utilisateur.



OpenTripPlanner



Routage multimodal des personnes

Younes JAZOULI BENLAHBOUB

Encadrement : Ismaïla Abderhamane NDIAYE

Constat

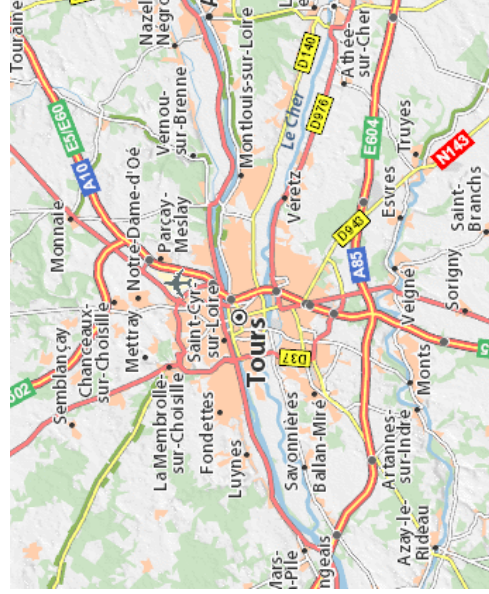
Lors d'un trajet un voyageur doit souvent emprunter un à plusieurs modes de transports (voiture, tram, métro ...) exploités par plusieurs transporteurs. Les solutions existantes peuvent ne pas être satisfaisantes, car elle ne supporte pas entièrement cette multimodalité.

Solution

Développer une application Web avec un Backend basé sur un calculateur d'itinéraire optimisé qui va nous permettre de faciliter cette multimodalité au niveau des correspondances entre modes et réseaux de transports.

Conclusion

Grâce à la mise en place de plusieurs outils technologiques, on peut développer une application qui répond parfaitement au besoin de l'utilisateur.



Routage multimodal des personnes

Résumé

Ce projet de recherche et développement a pour objectif de résoudre la problématique du routage multimodale des personnes, à travers le développement d'une application Web en utilisant divers outils technologiques.

Mots-clés

SAP, AngularJS, OTP, OSM, \LaTeX , \LaTeX

Abstract

This research and development project aims to solve the problem of multimodal routing of people, through the development of a web application using various technological tools.

Keywords

SAP, AngularJS, OTP, OSM

Tuteur académique

Ismâïla Abderhamane NDIAYE

Étudiant

Younes JAZOULI BENLAHBOUB (DI5)