|  |
| --- |
| Serena Sadek | TPI 2017 |
| TripTracker |
| Documentation Technique |

Table des matières

[1 Introduction 3](#_Toc485459348)

[1.1 Organisation 3](#_Toc485459349)

[1.2 Objectifs 3](#_Toc485459350)

[2 Analyse concurrentielle 4](#_Toc485459351)

[2.1 Facebook 4](#_Toc485459352)

[2.2 Travel Diaries 4](#_Toc485459353)

[2.3 MyTripJournal 4](#_Toc485459354)

[2.4 Travel Pod 5](#_Toc485459355)

[2.5 Positionnement par rapport à la concurrence 5](#_Toc485459356)

[3 Cahier des charges 6](#_Toc485459357)

[3.1 Définition de l’audience 6](#_Toc485459358)

[3.2 Définition du contenu 6](#_Toc485459359)

[3.3 Fonctionnalités 6](#_Toc485459360)

[3.4 Modèle conceptuel 7](#_Toc485459361)

[3.5 Maquette 8](#_Toc485459362)

[3.5.1 Arborescence des vues 8](#_Toc485459363)

[3.5.2 Page d’accueil 8](#_Toc485459364)

[3.5.3 Page personnelle 9](#_Toc485459365)

[4 Méthodologie 10](#_Toc485459366)

[4.1 Outils utilisés 10](#_Toc485459367)

[4.1.1 PHP 10](#_Toc485459368)

[4.1.2 MySQL 10](#_Toc485459369)

[4.1.3 JavaScript / JQuery 10](#_Toc485459370)

[4.1.4 AJAX 10](#_Toc485459371)

[4.1.5 Bootstrap 10](#_Toc485459372)

[4.1.6 Google Map API 10](#_Toc485459373)

[5 Analyse du contenu 11](#_Toc485459374)

[5.1 Arborescence 11](#_Toc485459375)

[5.1.1 Organisation 11](#_Toc485459376)

[5.1.2 Rôles 11](#_Toc485459377)

[5.1.3 Dépendances 12](#_Toc485459378)

[5.2 Usage d’éléments externes 12](#_Toc485459379)

[5.2.1 Bootsrap DateTimePicker 12](#_Toc485459380)

[5.2.2 Bootstrap File Input 13](#_Toc485459381)

[5.3 Fonctions remarquables 13](#_Toc485459382)

[5.3.1 Inscription 13](#_Toc485459383)

[5.3.2 Connexion 13](#_Toc485459384)

[5.3.3 Interface de création 14](#_Toc485459385)

[5.3.4 Gestion des Routes 15](#_Toc485459386)

[5.3.5 Enregistrement des informations 17](#_Toc485459387)

[5.3.6 Navigation 18](#_Toc485459388)

[5.3.7 Modification 20](#_Toc485459389)

[5.3.8 Suppression des voyages 21](#_Toc485459390)

[5.3.9 Sécurisation de la Map 21](#_Toc485459391)

[6 Test de l’application 22](#_Toc485459392)

[6.1 Stratégie de test 22](#_Toc485459393)

[6.2 Rapport de test restreins 22](#_Toc485459394)

[7 Conclusion 23](#_Toc485459395)

[7.1 Justesse de la planification 23](#_Toc485459396)

[7.1.1 Planning initiale 23](#_Toc485459397)

[7.1.2 Planning réel 23](#_Toc485459398)

[7.1.3 Commentaire sur la planification 23](#_Toc485459399)

[7.2 Modèle conceptuel 23](#_Toc485459400)

[7.3 Critique du résultat obtenu 23](#_Toc485459401)

[7.4 Améliorations possibles 24](#_Toc485459402)

[7.5 Intérêt personnel et gain de connaissance 24](#_Toc485459403)

[8 Bibliographie 25](#_Toc485459404)

[9 Table des figures 26](#_Toc485459405)

# Introduction

Cette documentation retrace les différentes étapes, de l’analyse préliminaire aux tests, du site web « TripTracker », réalisé dans le cadre d’un travail de fin d’étude. Cette application a pour but d’offrir un support ou l’utilisateur peut publier des articles concernant les étapes de ses voyages, tout en les situant sur une carte. Il permettrait à tout voyageur féru de technologie de garder une vue d’ensemble et d’organiser tous ses souvenirs. L’interface du site lui permettrait soit à l’utilisateur de naviguer à l’aide d’une carte, soit en sélectionnant un élément dans une liste.

Personnellement, j’ai choisi ce sujet à la fois pour son aspect technique que pour son objectif. Je suis quelqu’un qui aime beaucoup voyager et qui aime garder une trace écrite de mes aventures. Malheureusement, les photos qu’on garde dans un smartphone et les notes deviennent vite désorganisées et il m’arrive souvent de ne plus retrouver certaines d’entre elles. Une application permettant de regrouper chaque étape avec ses commentaires, ses images et sa localisation, et regrouper toute ces étapes sous forme de « road trip » est, à vrai dire, tout ce dont j’aurais besoin.

Ce projet représente donc pour moi un intérêt autant technique, pour ses technologies variées et sa construction atypique, que personnel.

## Organisation

Élève : Maître d’apprentissage :   
Serena Sadek Sandrine Wever  
[serena.sdk@eduge.ch](mailto:serena.sdk@eduge.ch) [sandrine.weber@edu.ge.ch](mailto:sandrine.weber@edu.ge.ch)

Experts :   
Sébastien Ducret Karim Maillard  
[sebastien.ducret@bluewin.ch](mailto:sebastien.ducret@bluewin.ch) [allani.karim@bluewin.ch](mailto:allani.karim@bluewin.ch)

Le travail s’effectuera sur 80 heures et sur dix jours. Il commencera le 6 Juin 2017 et se terminera le 19 du même mois. L’ensemble du travail devra être effectué dans ce laps de temps, à l’exception des prérequis demandés. L’organisation du travail devra correspondre le plus possible à la planification proposée dans le cahier des charges.

J’utiliserai pour réaliser ce projet l’ensemble des compétences que j’ai acquises durant ces trois dernières années d’études au CFPT. Pour ce projet, j’aurai l’usage de PHP, MySQL, JavaScript, JQuery, AJAX et Google Map API.

## Objectifs

Les objectifs de ce projet sont les suivants. Il faut créer un site web dont l’accès est restreint aux utilisateurs connectés. Une fois l’utilisateur connecté, ce dernier peut naviguer du global au détail de ses voyages, à la fois grâce à des indicateurs sur la carte, mais aussi grâce à une liste accessible en parallèle. De la même manière, l’utilisateur peut créer des trajets et étapes en leur attribuant une position.

En parallèle de ce développement, un journal de bord doit être tenu. Il permet de retracer toute les étapes du travail en temps réel, durant ces deux semaines. Une documentation utilisateur et une documentation technique doivent également être rédigées dans ce laps de temps.

Finalement, le travail devra être présenté aux experts dans le cadre d’une soutenance orale.

# Analyse concurrentielle

## Facebook

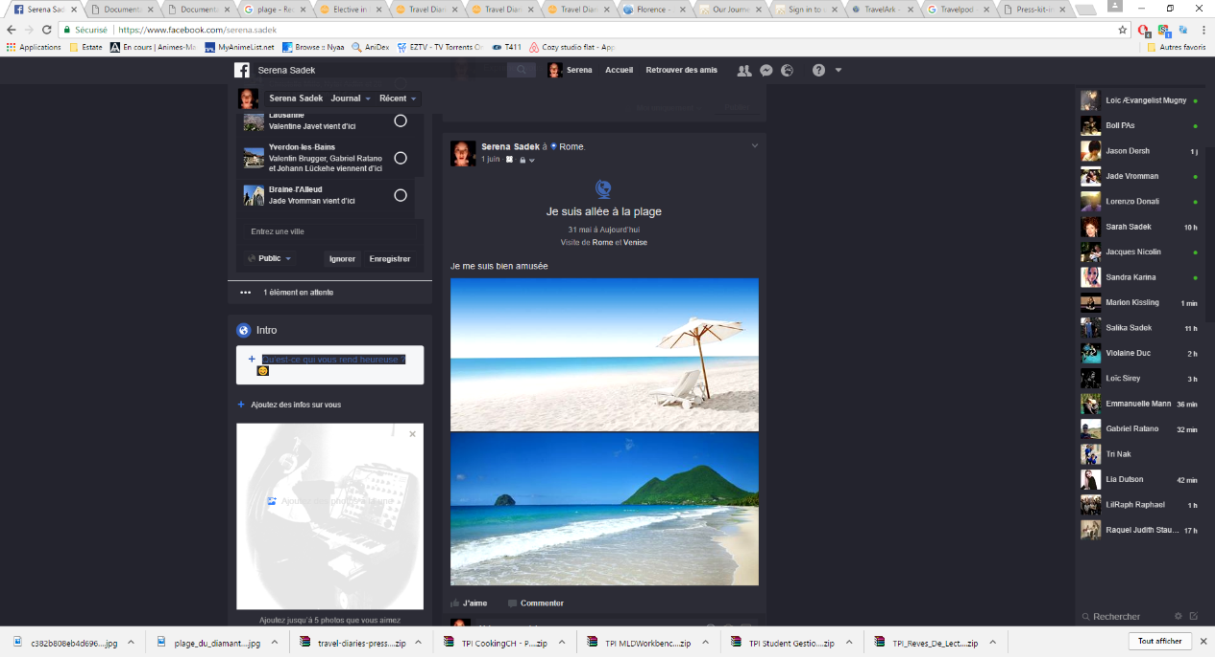


Figure  : Exemple de voyage que l’on peut publier sur Facebook

Dans cette analyse concurrentielle, je pourrais ne faire état que des sites spécialisés dans le même domaine que celui dans lequel j’aimerais situer mon application. Néanmoins, une grande partie de la concurrence se situe dans les réseaux sociaux, et particulièrement Facebook. Bien qu’ils ne reprennent pas exactement le même concept, il est toujours possible d’allier localisation, image et texte de différentes manières. Les réseaux sociaux de ce genre sont, grâce à leur polyvalence et leur visibilité, ceux qui rassemblent le plus grand nombre d’utilisateurs. En revanche, le système de filtrages des informations n’offre ni à l’utilisateur ni à ses amis une réelle visibilité sur son voyage. Facebook être le plus gros conçurent des journaux de voyages en ligne, mais aussi le moins fidèle.

## Travel Diaries

Ce site, tournant autour du concept de journal de voyage, est le plus récent, et donc le plus esthétique. Il permet à un utilisateur connecté de tenir ou plusieurs carnets de voyages, qui se présentent sous forme de texte auxquels on peut ajouter images et cartes. Les cartes peuvent contenir soit des points, soit des trajets, mais elles ne peuvent qu’être statiques. Ces cartes statiques n’étant pas légendées, il n’y a aucun moyen de situer directement la mini carte sur le globe. Ce site a la particularité de gérer les informations dans leur ensemble, c’est-à-dire qu’images, cartes et texte sont placés les uns à la suite des autres au gré de l’utilisateur.

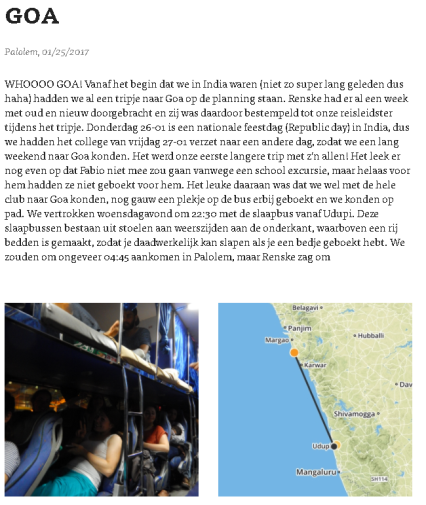


Figure  : Extrait d’un journal de voyage du site Travel Diaries

En conclusion, cette application web reprend les trois contenus (images, texte et localisation) qui vont être inclus dans le projet. Cependant, la navigation n’est pas organisée autour de la carte, et le découpage des voyages se fait en chapitres plutôt qu’en étapes.

## MyTripJournal

Cette application est beaucoup plus ancienne et son design en pâti : l’apparence n’est pas forcément esthétique, ni ergonomique. Cependant, elle est l’une des plus proches de ce que j’aimerais réaliser : Chaque utilisateur a une liste de voyages, contenant une description. Les voyages sont découpés en étapes, auquel on peut accéder soit en cliquant sur des marqueurs placés sur une carte statique, soit en les sélectionnant dans une liste. La carte de navigation est disponible sous plusieurs points de vue (différents continents ou pays), mais reste plus statique que dynamique.



Figure  : Un voyage de MyTripJournal avec sa description et ses étapes

Quant aux autres articles sur une autre page. Le contenu des articles est uniquement textuel, mais une section est réservée aux images. Le site fourni un grand nombre de champs dans les détails de ses étapes, mais l’obligation de charger une page pour chaque article pèse un peu sur l’expérience de l’utilisateur.

En conclusion, ce site à la construction la plus proche de celle de TripTracker. Son seul défaut qui serait à corriger est son manque de dynamisme et d’esthétisme.

## Travel Pod

Travel Pod a été, d’après Wikipédia, le premier site à offrir la possibilité de tenir un journal de voyage alliant texte, images (à partir de 2005) et localisation. Avec plus de vingt ans d’ancienneté, il offre même la possibilité d’imprimer et de relier ses carnets de voyage. Il a bénéficié du soutien de média tel que National Geographic, TripAdvisor ainsi que The New York Times. Dans l’ère du temps, il offre la possibilité de se connecter avec son compte Facebook et possède même une application disponible sur l’App store.

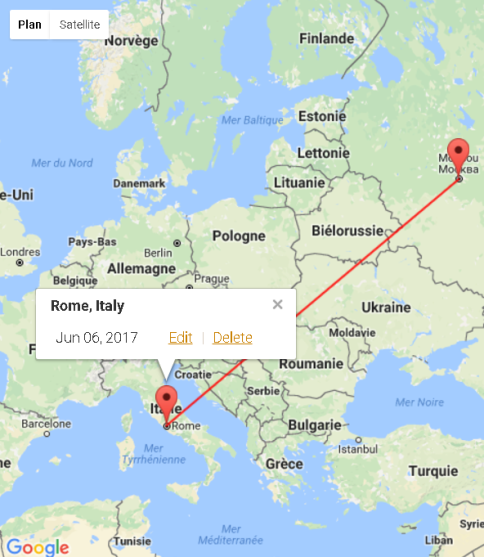


Figure  : Navigation dynamique et tracés, une fonctionnalité de Travel Pod

Malheureusement (et à ma grande surprise), le site va fermer ses portes le jour où ce travail sera rendu (c’est-à-dire le 19 Juin 2017). Il est le site le plus complet et dont l’approche est la plus proche de celle que je souhaiterais avoir. Le site offre la possibilité de naviguer parmi les étapes, et utilise probablement la même technologie (Google Map API) que TripTracker. Les articles sont avant tout définis par leur localisation, mais peuvent aussi contenir des photos et vidéos, ainsi qu’une date et un contenu textuel.

En conclusion, Travel Pod aurait été, dans un contexte lucratif, le plus gros concurrent de TripTracker. Bien qu’il ne tire pas pleinement parti de sa carte dynamique, elle est tout de même présente. Personnellement, je regrette que le site ferme ses portes maintenant : il m’a donné envie de tester plus en détail ses fonctionnalités.

## Positionnement par rapport à la concurrence

Ce travail se fait dans un cadre scolaire, et n’a donc pas de but lucratif à proprement parler. Néanmoins, avoir un aperçu de son impact éventuel peut-être une réelle motivation, et ce particulièrement si une application strictement identique n’a pas encore été développée.

Bien qu’il existe un certain nombre d’application web faisant office de journal de voyage, aucun ne reproduit exactement ce qui se veut être fait dans TripTracker. La plupart du temps, la conceptualisation se fait sur la base d’un journal papier, et lorsqu’une carte est intégrée, elle n’est souvent pas interactive. Le dernier « concurrent » possède, lui, une navigation avec une carte similaire à celle utilisée dans cette application. Malheureusement, les tracés ne sont que des lignes droites, ne respectant même pas la courbure de la terre.

En misant tout sur le dynamisme, et avec un peu d’investissement, TripTracker pourrait avoir la prétention d’être un concurrent sérieux. Comme je l’ai précédemment évoqué, la finalité de ce travail n’est pas lucrative mais instructive, évaluative. Néanmoins, ce placement par rapport à la concurrence est une source de motivation supplémentaire, et donne envie de s’investir.

# Cahier des charges

Les zones grisées sont tirées du cahier des charges, soumis avant le début du projet et validé par les experts. Ces descriptions mériteraient, à mon sens, d’être plus détaillés, et c’est pour cette raison que je me suis permis d’y rajouter certains points.

## Définition de l’audience

Ce site web à pour cible des utilisateurs souhaitant conserver des données textuelles, géographiques et des images relatives à leurs voyages, et qui souhaites les organiser.

Ceci décrit uniquement les fonctionnalités du site web. En réalité, le projet tel qu’il est rendu n’est pas accessible via le web, et nécessite l’installation d’un serveur local. Ce serveur local, EasyPhp, requière une version de Windows égale ou postérieure. Il n’est pas impossible de mettre en place le site sur d’autres structures, mais le manuel utilisateur est dédié à des utilisateurs de Windows.

Un autre aspect restrictif du site est que, bien qu’il soit, tel qu’il est rendu, destiné à un serveur local, il nécessite tout de même une connexion à internet pour utiliser les services de Google Map API.

## Définition du contenu

Voici le texte contenu dans l’énoncé validé par les experts :

Les utilisateurs doivent obligatoirement s'identifier pour accéder au contenu du site. Il est donc dirigé sur une page de connexion à son arrivée sur le site. Une fois connecté, il accède à une unique page. Sur cette page, l'utilisateur peut créer un voyage, qui retrace ses aventures à ses différentes étapes. L’utilisateur doit indiquer la localisation (actuelle ou à entrer) et la date de son étape (actuelle ou à sélectionner), mais peut également rédiger un commentaire accompagné d’images.

Les tracés des différents voyages s’affichent sur la carte. La liste de ces voyages s’affiche dans l’aside. Lorsque l’utilisateur sélectionne un voyage, les autres voyages disparaissent de la carte et la liste des étapes s’affiche. L’utilisateur peut visualiser les détails de l’étape en cliquant soit sur son marqueur (sur la Map), soit dans la liste (dans l’aside). Un bouton “retour” permet de repasser d’un article à la liste des étapes, et des étapes à la liste des voyages.

Néanmoins, je n’ai que très peu décrit, dans cet énoncé, la phase de création. Celle-ci se fait dans une troisième section. L’utilisateur peut y créer et éditer ses étapes. La localisation peut être entrée manuellement sous forme d’adresse, ou en cliquant sur la Map. Une fonction de géolocalisation est également disponible.

## Fonctionnalités

* L’utilisateur peut s’inscrire
* L’utilisateur déjà inscrit est reconnu
* Lors de la connexion, l’utilisateur est redirigé avec succès vers la page principale du site
* L’utilisateur peut se déconnecter et être redirigé vers la page d’accueil
* Les voyages s’affichent dans la fenêtre de navigation avec un code couleur correspondant aux tracés sur la carte
* La liste des voyages est paginée par groupe de cinq éléments
* Lorsqu’un clique sur le titre d’un voyage ou son tracé correspondant, les autres tracés disparaissent, la liste de ses étapes et les marqueurs correspondants s’affichent et la carte se cadre sur le voyage.
* Lorsqu’on clique sur le titre d’une étape ou le marqueur lui correspondant, les détails apparaissent dans la section dédiée.
* Si l’étape contient des images, elles apparaissent sous forme de carrousel
* Une seule section peut être ouverte à la fois, entre la navigation, les détails d’une étape et la création. La carte et son contenu s’adaptent à la section ouverte.
* La section d’ajout permet de générer dynamiquement des onglets, correspondant aux étapes du voyage.
* Dans l’onglet d’une étape, l’utilisateur peut sélectionner la localisation en cliquant à un emplacement de la carte.
* L’utilisateur peut sélectionner l’emplacement d’une étape à l’aide d’une barre de recherche.
* Les étapes possédant une localisation sont identifiées par des marqueurs sur la carte
* L’utilisateur a la possibilité de choisir sa localisation actuelle.
* Lorsqu’une localisation change, un tracé entre les localisations connues est généré.
* L’utilisateur peut sélectionner une date (la date par défaut est celle du jour même)
* L’utilisateur peut importer des images
* L’utilisateur peut ajouter un commentaire
* L’utilisateur peut supprimer une étape lors de son édition
* L’utilisateur peut enregistrer son voyage
* Si l’utilisateur n’a pas complété certaines informations d’un voyage, les informations ne seront pas envoyées et les champs non complétés correctement seront mis en surbrillance.
* Si les dates sont incohérentes, le voyage ne sera pas non plus validé.
* Lorsque le voyage est validé, la section d’ajout est fermée et les deux autres sections sont de nouveau accessibles.
* L’utilisateur peut demander à modifier un voyage via la section de navigation. Cette action ouvrira une section de création contenant toute les informations existantes.
* L’utilisateur peut supprimer un voyage

## Modèle conceptuel

Ce modèle conceptuel est celui ayant été validé par les experts, lors de la proposition du cahier des charges.

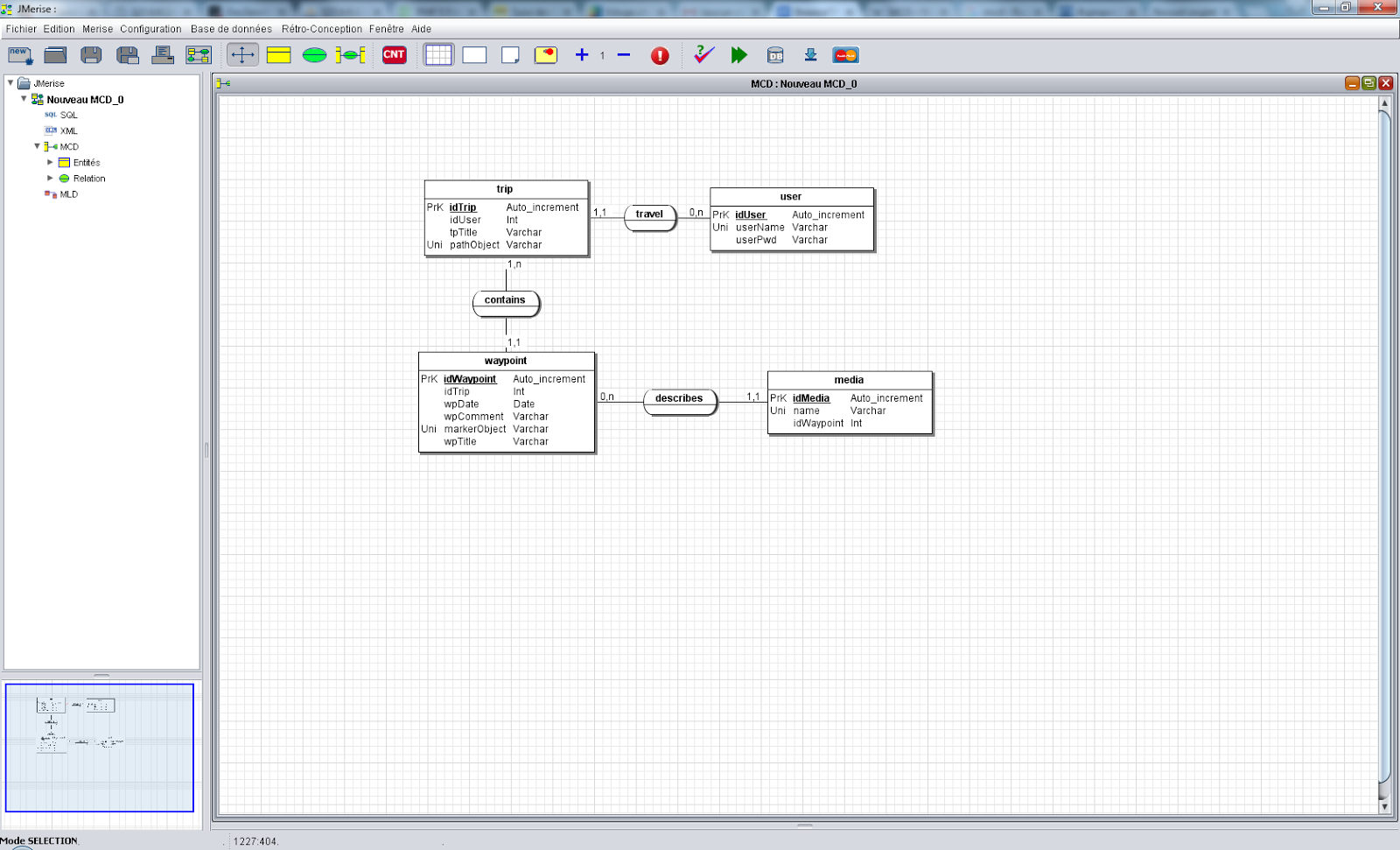


Figure  : Modèle conceptuel du projet soumis avec le cahier des charges

## Maquette

### Plan de site

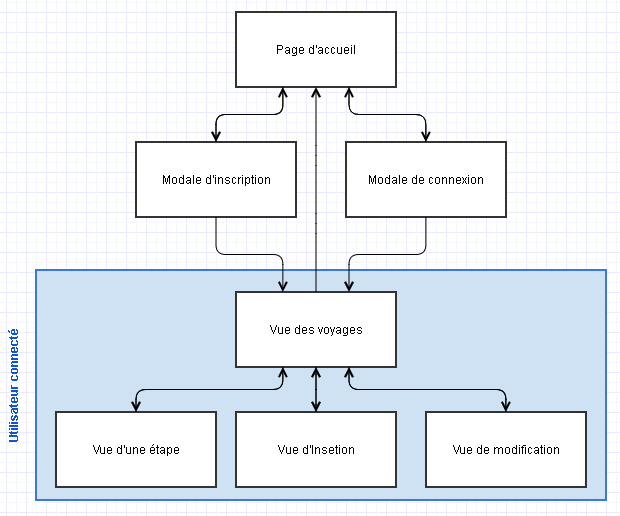


Figure  : Arborescence des vues dans l’application

### Page d’accueil

C’est la première vue avec laquelle l’utilisateur interagira. Elle contient un carrousel d’image décrivant globalement les fonctionnalités du site. Par les boutons dans le corps de la page ou ceux dans la barre de navigation, l’utilisateur peut afficher les modales de connexion (1) et d’inscription (2). Une fois inscrit ou connecté, l’utilisateur sera redirigé vers sa page personnelle.

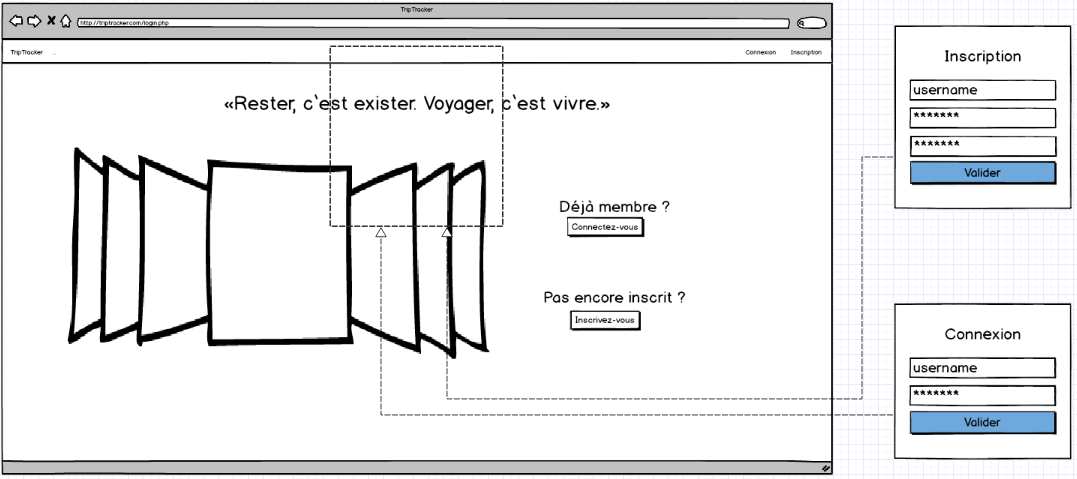


Figure  : Maquette de la page d’accueil avec ses modales d’inscription et de connexion

### Page personnelle

1. C’est la vue sur laquelle l’utilisateur arrivera une fois connecté ou inscrit. Dans le panneau qui apparait sur la gauche, l’utilisateur aura un aperçu sous forme de panneaux réductibles de ses voyages. Ses panneaux contiennent la liste des étapes. L’utilisateur pourra charger les pages suivantes à l’aide de liens présents en dessous des voyages. Il pourra également modifier ses voyages en cliquant sur le pinceau à côté de leur titre, ou les supprimer en cliquant sur la croix au même endroit.
2. C’est la vue sur laquelle l’utilisateur pourra afficher les détails d’une étape. Elle s’affichera lorsqu’il a cliqué sur un marqueur ou sur le lien d’une étape. A noter que dans le cas où l’étape ne possède pas d’image, la section du carrousel sera masquée.
3. C’est la vue de création, mais aussi de modification. L’utilisateur y accède soit en cliquant sur la miniature en bas à droite de l’écran, soit en cliquant sur le pinceau à côté du titre d’un voyage existant. Dans le dernier cas, les informations existantes seront chargées.

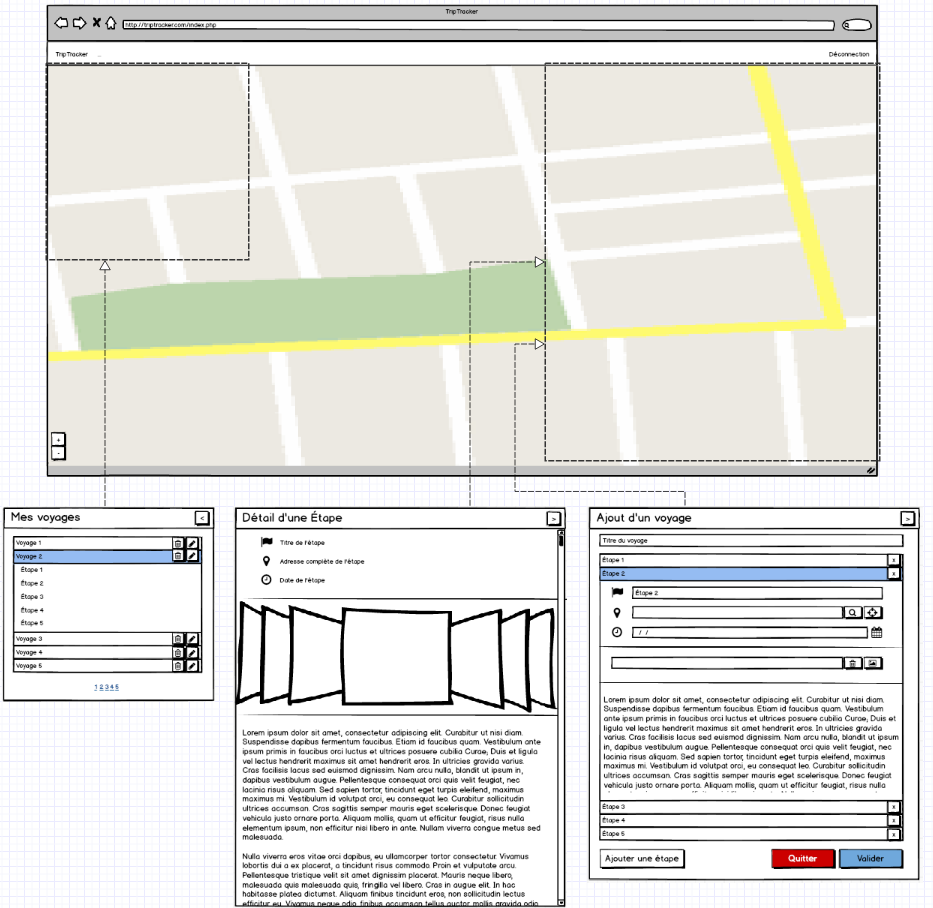


Figure  : Maquette de la page personnelle de l’utilisateur avec ses trois vues

# Méthodologie

## Outils utilisés

Bien qu’à n niveau technique de CFC, le choix des outils se fasse plus par restriction que par sélection, il n’en est pas moins possible tirer profit de ces derniers. Dans ce point sont décrites toutes les technologies que j’utilise ainsi que leurs outils dont j’ai tiré profit dans mon projet.

### PHP

Ce langage est sans doute le langage avec lequel nous avons acquis le plus d’expérience, du moins dans le cadre scolaire. A ce jour, trois modules ayant pour thématique le PHP ont été dispensés, sans parler des heures de pratique en atelier. En ce qui concerne le web, c’est le seul langage dans lequel je maitrise les sessions, les échanges avec la base de données ou l’upload de média depuis le client. Ainsi, bien que mon application repose en grande partie sur une API fonctionnant en JavaScript, tous les points concernant la base de données, les médias ou la session sont relégués au serveur.

### MySQL

Qui dit base de données dit requête, et qui dit requête dit SQL. MySQL est le premier et le seul langage de requête structurée auxquels nous avons été confrontés, mais aussi le langage implémenté dans EasyPhp, qui nous fait office de serveur local. Dans mon application, MySQL est utilisé pour effectuer des requêtes à l’aide d’objet PDO, en PHP.

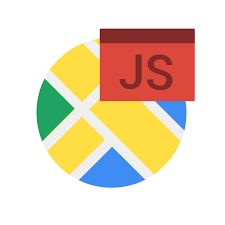
### JavaScript / JQuery

Bien que ce soit le langage avec lequel nous avons le moins de pratique, c’est probablement celui que j’ai le plus utilisé dans le cadre de mon année terminale. Car bien qu’il ne fasse l’objet d’aucun cours dans le cadre du CFC, il est difficile de parler d’application Web sans aborder le sujet. Dans mon application, JQuery me à interagir avec les composants Bootstrap, à interagir avec la Map, mais surtout à modifier, générer, masquer, supprimer les composants de la page web, permettant au site d’être fonctionnel dans sa construction single page.

### AJAX

Avec d’un côté une API basée sur JavaScript et de l’autre une gestion des données faite en PHP, il faut quelque chose pour les relier. Dans ce cas, c’est AJAX qui fait le travail. C’est en partie grâce à cette technologie que la construction de mon application en single page devient possible, car grâce à elle, le rafraichissement de la page n’est pas nécessaire. Ainsi, les informations du client peuvent être conservées.

### Bootstrap

Bootstrap est l’outil qui permet d’avoir à moindre investissement de temps un site esthétique contenant des composants complexes comme des panneaux ou des fenêtres modales. De plus, de nombreux composants non-officiels développés et mis à disposition par des usagers : j’en utilise deux dans ce projet.

### Google Map API

Mettant à disposition une grande quantité de données, une grande variété de services et une documentation bien fournie, Google Map API est l’outil qui par excellence pourrait fournir le résultat attendu en matière de dynamisme.

# Analyse du contenu

## Arborescence

### Organisation

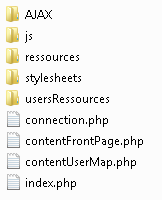


Figure  : Organisation à la racine du projet

Les fichiers sont organisés dans le serveur tel que ci-contre : le fichier principal, ses deux vues ainsi que le fichier servant à établir la connexion avec la base de données se trouvent à la racine. Les fichiers JavaScript se trouvent dans le dossier « js », et les fichiers faisant la liaison entre la base de données et le fichier JavaScript se trouvent dans le dossier « AJAX ». Le dossier « ressource » contient tous les éléments externes aux librairies disponibles et le dossier « Stylesheet » contient les fichiers CSS. Finalement, le dossier « usersRessources » contient deux dossiers : « picture » et « path ». Ces deux dossiers contiennent respectivement les images et les tracés enregistrés par les utilisateurs.

### Rôles

|  |  |  |
| --- | --- | --- |
| **Dossier** | **Fichier** | **Rôle** |
| AJAX | DataInsertModif.php | S’occupe de la création, la suppression et la modification des voyages et de leurs étapes. |
| PictureInsertModfi.php | Appelé par les File Input modifiés. Il gère les images en les ajoutant ou les supprimant du stockage. Il gère également leur référencement dans la base de données. |
| UserDbRelation.php | Faire le lien entre le client et la liste des utilisateurs dans la base de données. Il authentifie et enregistre les utilisateurs. |
| callDisconect.php | Détruit la session pour déconnecter l’utilisateur. |
| navigationData.php | Permet la lecture des voyages et de tout leur contenu (étapes et leurs images comprises). |
| js | LoginRegister.js | Fait le lien entre la vue et le serveur, dans le cadre de l’inscription et de la connexion. Il récupère l’information des champs, les envoie au serveur, puis affiche le résultat obtenu à partir de la réponse de ce dernier. |
| InsertDataTreatment.js | Rassemble les données, les traite et les formate avant de les envoyer au serveur, lors de la validation d’une modification ou d’un ajout de voyage. |
| insertInterface.js | Rassemble toute les fonctions propres au bon fonctionnement de l’interface de création/modification (focus, génération d’onglet, etc.) |
| modifPlugin.js | Permet à l’interface de création d’être utilisée pour la modification en chargeant le contenu du voyage, et en générant des composants préremplis. |
| templateMap.js | Veille au bon fonctionnement du template : gère l’ouverture et la fermeture de panneaux, l’initialisation et la sécurité de la Map. |
| tripNavigation.js | Charge les voyages de l’utilisateur et génère les composants leur correspondant. Il gère également les interactions entre l’utilisateur et les composants dynamiques. |
| stylesheet | frontPage.css | Mise en forme de la page d’accueil (celle disponible lorsque l’utilisateur n’est pas connecté. |
| templateMap.css | S’occupe de la mise en forme de la page personnelle de l’utilisateur. |
| [root] | connection.php | Contient la fonction instanciant une connexion avec la base de données, sous forme d’objet PDO. |
| contentFrontPage.php | Vu affichant un carrousel décrivant les fonctionnalités du site, ainsi que des contrôles, accédant à deux modales, respectivement contenant les formulaires d’inscription et de connexion. C’est la vue de la page d’accueil. |
| contentUserMap.php | Vue affichant contant la carte ainsi que ses trois panneaux. C’est la vue de la page personnelle de l’utilisateur. |
| index.php | Page web contenant de manière statique la navigation supérieure, et affichant différentes vues selon si l’utilisateur est connecté ou non. |

### Dépendances

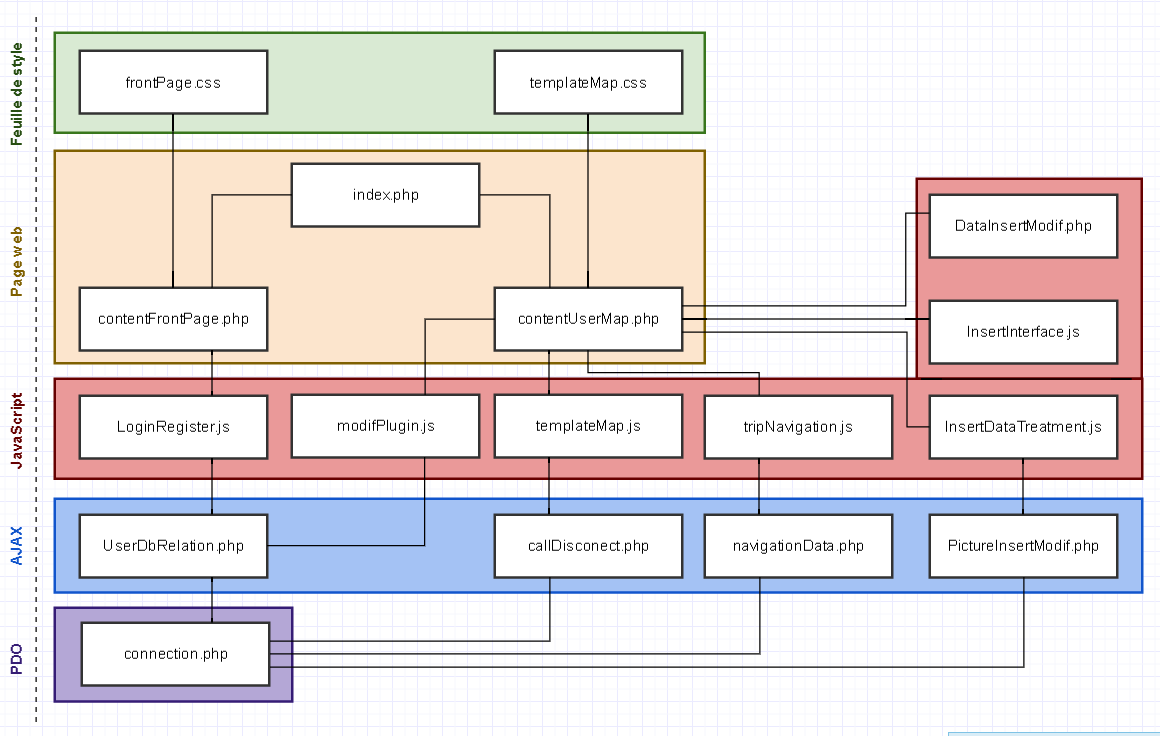


Figure  : Représentation graphique des dépendances

## Usage d’éléments externes

### Bootsrap DateTimePicker

Ce plugin de Bootstrap, développé par Eonasdan, et est disponible sur son compte GitHub. Il a l’avantage d’être bien organisé dans son contenu, mais aussi d’être très bien documenté, accompagné d’une série d’exemples fonctionnel et d’une communauté d’utilisateurs actif, répondant à de nombreuses questions sur des forums comme StackOverflow.

Cette version améliorée du DateTimePicker est initialisée à partir de JQuery, et on peut lui spécifier de nombreux paramètres tels que le type et le format de donnée temporel que l’on veut récupérer. En plus d’être facile d’utilisation, ce composant est réalisé avec Bootstrap, tout comme le reste de l’application : cela lui permet d’avoir un style correspondant parfaitement aux autres composants.



Figure  : Exemple fourni par l’auteur du DateTimePicker

### Bootstrap File Input

Tout comme le DateTimePicker, l’input de type File n’est pas nativement amélioré dans la version de Bootstrap que j’utilise. Néanmoins, certains plugins (comme celui détaillé au point précédent) bénéficient d’une grande polyvalence et d’une excellente documentation. Le plugin utilisé dans cette application, pour l’input de type file, s’est très vite imposé : il semble être un incontournable et est très souvent cité dans les forums de développement. Il permet, à l’aide de quelque ligne de code, d’obtenir un composant ergonomique, permettant le drag and drop, et surtout permettant l’upload de fichier via AJAX.

De plus, il permet de pré visualiser dans l’input des fichiers contenu dans le serveur. Cette fonctionnalité en fait le candidat parfait, car avec celle-ci, il n’est pas nécessaire d’utiliser d’autres composants lors de la modification de voyage existant. Mais avec tous ses avantages, il reste un inconvénient : les images ne sont pas traitées en même temps que le reste de champs, vu que chaque input lance son propre call AJAX.

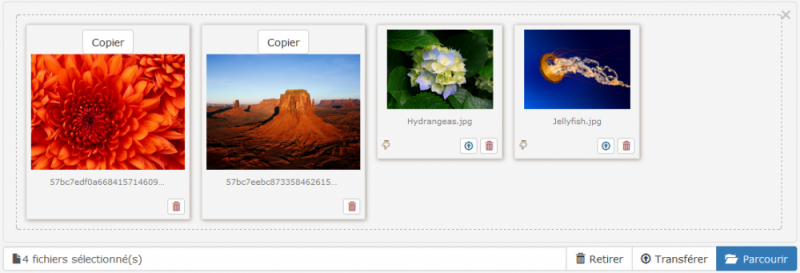


Figure  : Exemple du File Input

## Fonctions remarquables

### Inscription

Le traitement de l’inscription se lance lorsque l’utilisateur valide le petit formulaire se trouvant dans la modale de connexion. Une vérification de la complétion des trois champs va d’abord être effectuée. Si l’un des champs n’est pas rempli, il est mis en évidence et le traitement s’arrête là.

Sinon on vérifie la concordance du mot de passe et de la confirmation. Un message d’erreur sera affiché en cas de non concordance, et sinon les informations seront envoyées au serveur à la base de données à l’aide du Call AJAX. Les informations récupérées par le PHP sont filtrées. Une fonction vérifie ensuite l’existence d’un utilisateur du même nom. Si aucun utilisateur de ce nom n’existe, L’utilisateur est inscrit dans la base de données. Pour chaque cas de figure cités, le serveur retourne un code. Selon le code retourné, un message d’erreur ou de réussite va être affiché. Si l’utilisateur a été correctement inscrit, ce dernier va être redirigé vers sa page personnelle.

### Connexion

La connexion se passe de la même manière que l’inscription : validation par un bouton, vérification de la complétion des champs… Elle ne comporte juste pas de traitement côté client. Les informations sont directement envoyées au serveur via AJAX, puis filtrées. Le serveur interroge la base de données sur l’existence d’un utilisateur dont le nom est tel que l’utilisateur l’a entré. Si un tel utilisateur existe, le serveur compare une version hachée en SHA1 du mot de passe entré par l’utilisateur à celle retournée par la base. Dans ces trois, ou l’utilisateur n’existe pas, son mot de passe ne correspond pas ou ses informations sont correctes, le serveur retourne trois codes différents. Le client, selon le code retourné, affiche un message d’erreur ou de succès, et dans ce dernier cas redirige l’utilisateur vers sa page personnelle.

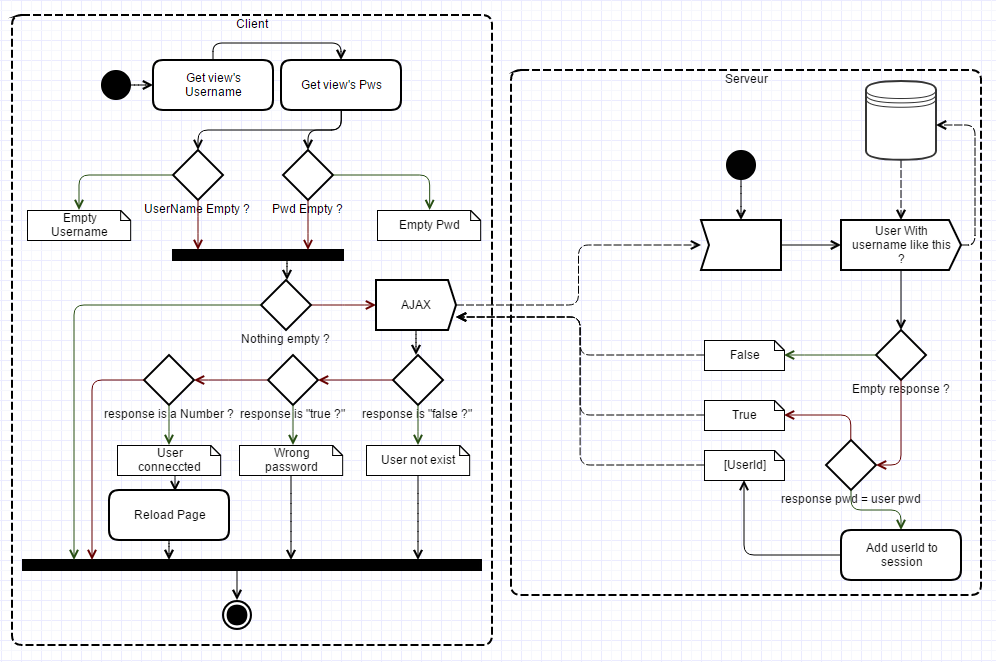


Figure  : Algorigramme de connexion

### Interface de création

#### Génération dynamique

Lorsque l’utilisateur se trouve sur l’interface de création, il se trouve en présence de trois boutons, dont un qui est l’objet de ce point : le bouton « ajouter une étape ». Ce bouton permet à l’utilisateur d’ajouter à l’affichage un nouvel panneau, correspondant à une nouvelle étape. Ce bouton à pour action d’ajouter dans un espace HTML (ici un div possédant l’id « Insertion Content ») un composant contenant tous les champs permettant à l’utilisateur d’en ajouter les détails. Tous les champs de ce nouvel onglet dont nous auront besoin lorsqu’il faudra valider le formulaire seront nommés de façon unique, à l’aide d’un identifiant textuel qui définit leur rôle, et d’une variable globale (count), qui différencie ces champs des champs des autres panneaux. Par exemple, le champ de titre du premier panneau généré sera nommé « title0 », et celui du cent-vint-quatrième panneau sera nommé « title123 ». A l’instar de ses champs, le panneau lui-même est nommé de la même manière.

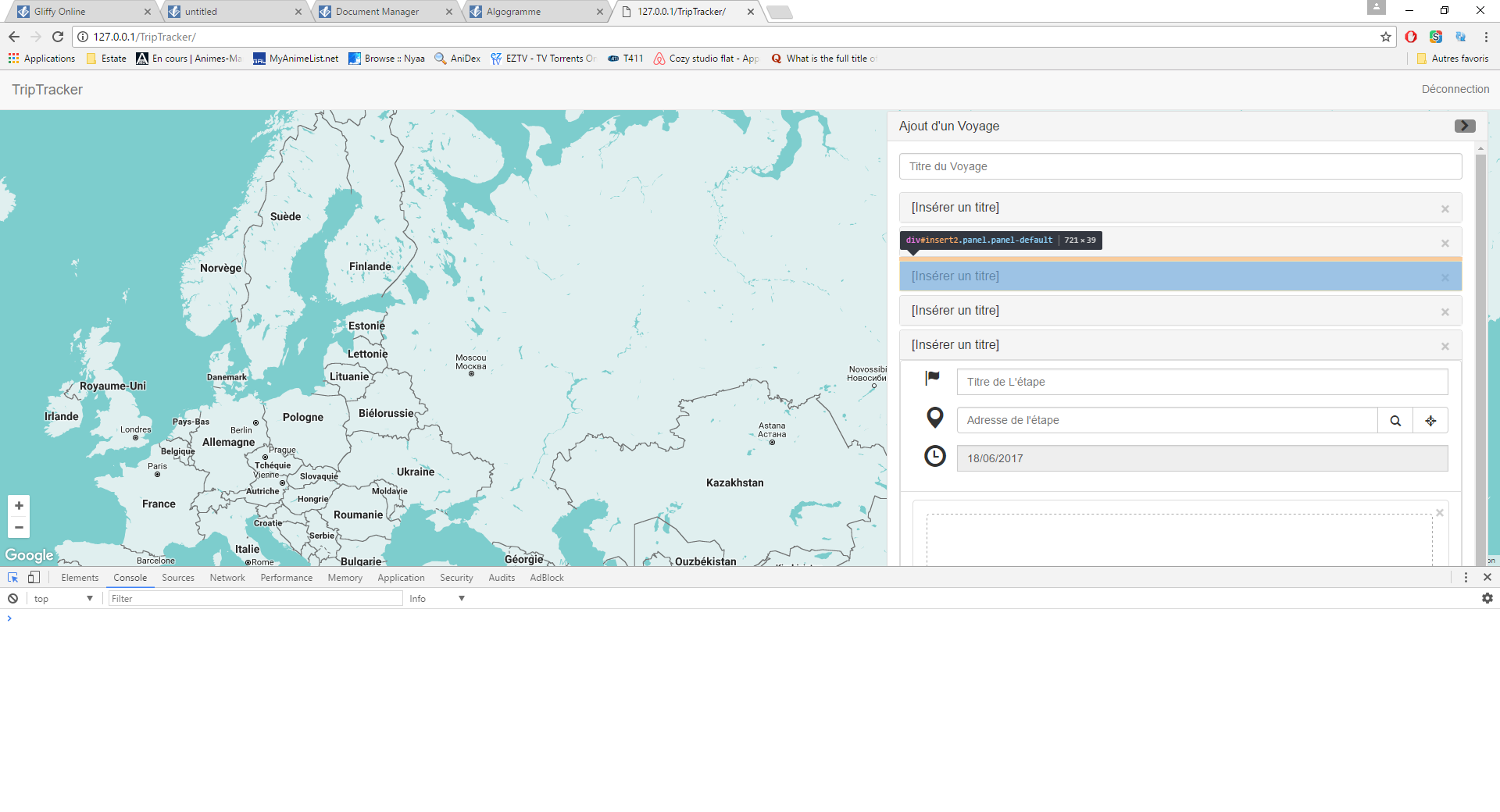


Figure  : Panneau généré dynamiquement

Les éléments externes nécessitent une initialisation. Pour le DateTimePicker, la date du jour est récupérée et le composant est initialisé avec cette date au format « jj/mm/aaaa ». Le File Input est initialisé avec un grand nombre de paramètres, notamment l’adresse à laquelle il va envoyer ses images lors de l’upload et les informations qui les accompagnent.

#### Suppression d’étape

Chaque étape possède, à droite de son titre, une petite croix. Celle-ci commande la suppression de cette étape, et est comme d’autres champs du panneau, identifié à l’aide d’une particule textuelle et d’un d’indice numérique. Lorsqu’un click est intercepté sur l’un de ses boutons, on en retire la particule numérique pour obtenir l’index de l’étape. Le panneau correspondant à cet index ainsi que ses éventuels marqueur et tracés sont supprimés.

#### Focus

Le focus est géré à l’aide d’une variable globale, initialement nulle. Lorsque l’utilisateur sélectionne un panneau correspondant à une étape pour en afficher le contenu, le numéro de ce panel est récupéré à l’aide de son identifiant et le focus prend cette valeur. Lorsqu’un panneau est fermé ou supprimé lorsqu’il était ouvert, la variable focus est à nouveau nulle.

#### Gestion des Routes

##### Sélection d’emplacement

Lorsque le focus est défini et que l’interface de création est active, la carte réagit au click. Cette mise en place se fait à l’initialisation de la carte, en y ajoutant manuellement un évènement comme ci-dessus. Lorsque cet évènement est déclenché, il fait appel à une fonctionnalité de Google Map API : le géocoder. Cette fonctionnalité permet d’entrer un élément géographique (une adresse ou une mesure de latitude et longitudes), et de récupérer à partir de cette information un objet contenant un grand nombre d’informations concernant cet emplacement. Ici, on passe en paramètre du géocodeur la latitude et la longitude récupérée par l’évènement click. On crée un marqueur qui va être affiché sur la carte, mais également stocké pour en récupérer les informations par la suite. L’adresse récupérée par le géocodeur est insérée dans les champs contenant l’adresse de l’étape sélectionnée.

##### Recherche par adresse

Lorsque l’utilisateur click sur la loupe, à droite du champ d’adresse d’une étape, on fait appel au même service de Google Map API que celui détaillée dans le point précédent : le géocodeur. Mais cette fois, c’est l’adresse qui est passée en paramètre, et c’est le géocoder qui nous permet de récupérer la position exacte. Lorsqu’on obtient la réponse du géocoder, on crée un marqueur à l’aide des latitudes et longitudes reçues, et on met à jour le champ d’adresse à l’aide de l’adresse reçue, qui est une version corrigée de l’adresse que l’utilisateur a entrée.

##### Géolocalisation

Le second des trois services de Google Map API utilisé dans ce programme est le service de géolocalisation. Ce dernier fonctionne sur un ordinateur à l’aide d’informations sur la connexion au réseau, et sur mobile à l’aide du service de géolocalisation satellite. Ainsi, lorsque l’utilisateur click sur le second bouton « Cible » champ d’adresse, c’est à ce service qu’on fait appel. Ce dernier ne retournant que la latitude et la longitude, on fait ensuite appel à un géocodeur, prenant en paramètre des coordonnées géographiques, pour récupérer l’adresse.

##### Stockage des informations géographiques

L’enjeu résidant dans le stockage d’informations géographique est le suivant : les étapes sont nommées par incrémentation, au fur et à mesure de leur création. Ce qui signifie qu’à partir du moment où une étape est supprimée, la numérotation n’est plus linéaire. Or pour récupérer les informations d’une étape, pour supprimer ses éléments qui sont en relation, ou pour plein d’autres opérations encore, il est impératif de connaitre avec exactitude tous l’emplacement des composants liés à une étape. Les règles régissant leur emplacement doivent donc être applicables, quel que soit le nombre d’ajout ou de suppression effectués.

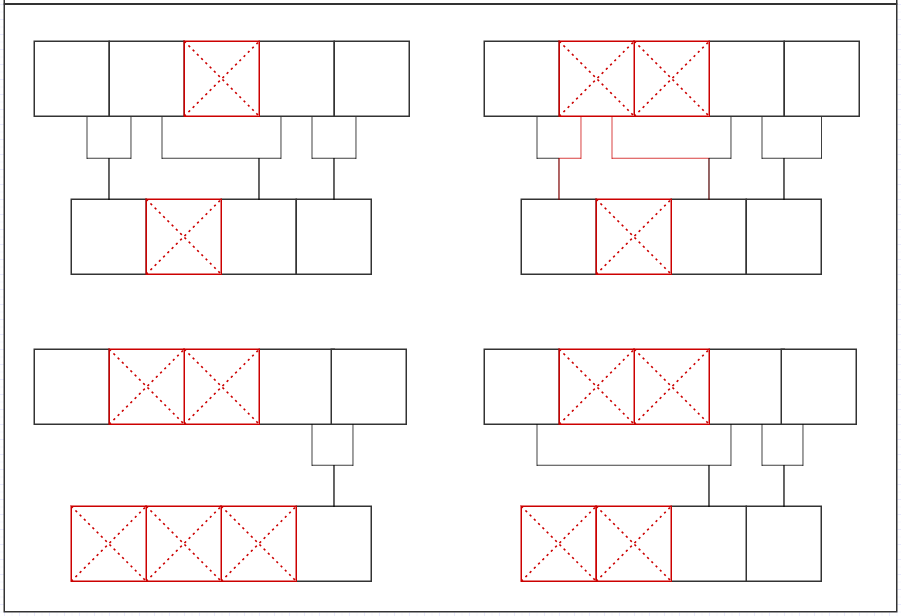
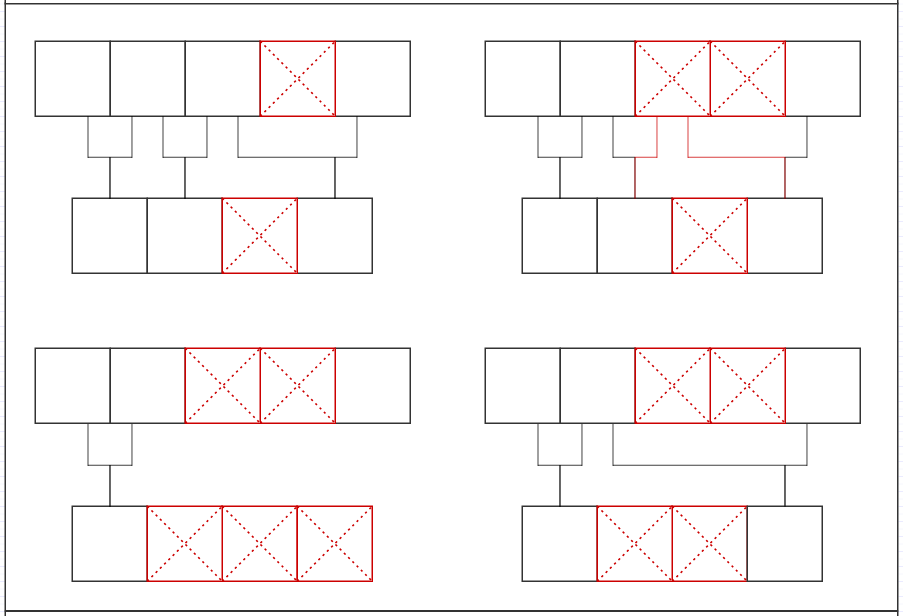
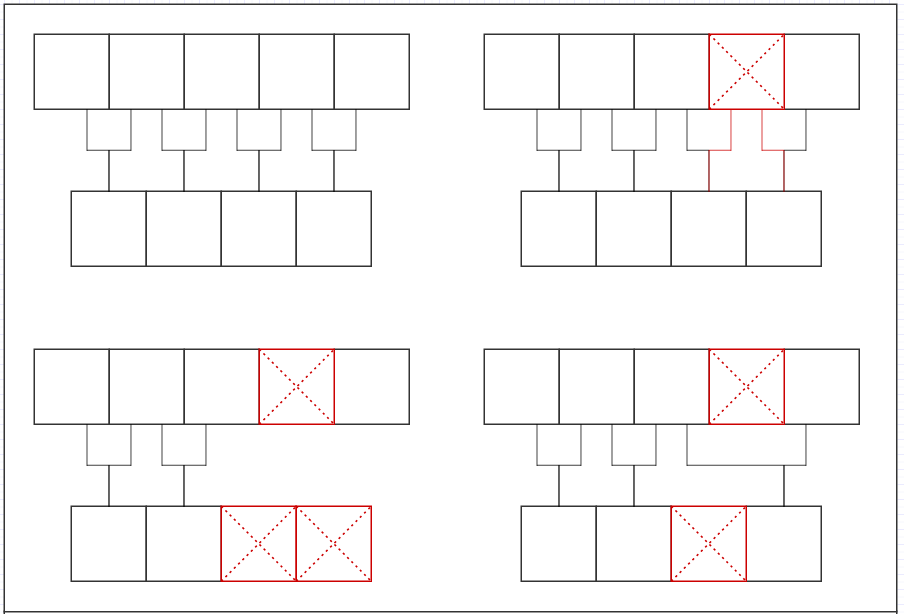


Figure  : Suppression de marqueur et attribution de tracé dans trois cas

Dans la construction que j’ai choisie, les règles sont la suivante. Soit deux tableaux, le tableau des marqueurs et le tableau des tracés reliant les marqueurs connexes deux par deux. A l’exception du cas ou le tableau des marqueurs est vide, la longueur du tableau tes tracés est la longueur du tableau des marqueurs, moins un. Comme on les place dans l’ordre, le tracé 0 relie les points 0 et 1, le tracé 1 relie les points 1 et 2, etc. Réciproquement, on peut dire que le tracé reliant A à B ou A < B est stocké à la position B-1.

Mais vu que la possibilité est laissée à l’utilisateur de laissé une étape provisoirement sans position, il convient de différencier les tracés et marqueurs pas encore défini de ceux correspondant à une étape supprimée. Ainsi les éléments pas encore défini sont indiqués par une chaine de caractères « none », et les « trous » sont nuls. Les éléments remplis contiennent les objets Google Map API liés à leur affichage sur la carte, ainsi que ceux contenant leurs informations.

A partir de cette définition, il est possible de stocker et de retrouver tous les tracés, quel que soit le nombre de marqueurs supprimés. Prenons donc un marqueur de position P. Les marqueurs avec lesquels il est en relation sont les premiers marqueurs définis (non nuls) lorsqu’on parcoure le tableau des marqueurs de la position P-1 à 0 et de la position P+1 à la fin du tableau. De la même manière, les tracés actuels auxquels P est lié sont les premiers tracés définis en parcourant le tableau des tracés de P-1 à 0 et de P à la fin du tableau.

##### Traçage des routes

Le traçage des routes se fait à l’aide du troisième service de Google Map API utilisé dans ce programme : le service de route. Ce dernier sert à chercher un itinéraire entre deux localisations. Il offre de nombreuses options, notamment le moyen de transport utilisé. Pour maximiser l’accessibilité des emplacements entre eux, les requêtes sont paramétrées avec le moyen de transport le plus polyvalent : la marche. Néanmoins, certains emplacements très isolés entre eux resteront inaccessibles les uns par rapport aux autres. Pour ce cas particulier, ils seront reliés par une Polyline (tracé sur la Map) dite « géodésique ». Cette ligne particulière apparait courbe car c’est une ligne qui, au lieu d’être rectiligne sur un plan, serait rectiligne sur la sphère qu’est la terre. Logiquement, c’est le tracé qu’un engin volant emprunterait pour aller du point A au point B, inaccessible les uns par rapport aux autres par la terre…

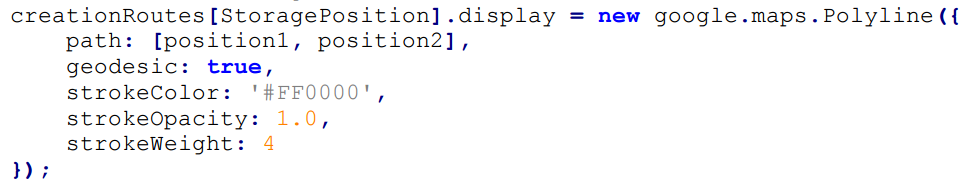


Figure  : Instance d’une Polyline géodésique

Le traçage de la route se fait dans deux cas de figure : le premier lors de la définition d’un marqueur, le second lors de la suppression d’une étape. Lorsqu’on définit la position d’une étape une étape adjacente à celle-ci possède un marqueur défini, on trace la route entre ces deux points à l’aide des coordonnées géographiques de leur marqueur respectif. Lorsqu’un marqueur est supprimé et que les deux étapes maintenant adjacentes possèdent chacun un marqueur défini, on relie ces deux étapes entre elles par le même procédé.

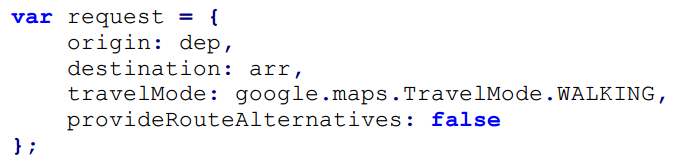


Figure  : Paramétrage des routes dans l’application

#### Enregistrement des informations

##### Récupération du contenu

Lorsque l’utilisateur valide son voyage, il faut en premier lieux récupérer les différentes étapes générées dynamiquement, ainsi que leur contenu. Pour se faire, on utilise la variable globale servant à la numérotation des éléments dynamiques. Pour chaque valeur de 0 à cette dernière, on vérifie si le panneau correspondant existe toujours. Si c’est le cas le contenu des champs de titre, de date, et de commentaire sont récupérés. Si un marqueur est défini pour cette étape, sa latitude, sa longitude et son adresse sont également récupérés. Toutes ces valeurs récupérées sont placées dans un tableau associatif destiné à contenir toute les informations du voyage à l’exception de son tracé.

##### Validation du contenu

Une fois les informations récupérées, celle-ci sont vérifiées. La complétion du titre du voyage est d’abord vérifiée, puis ce sont les étapes qui une à une sont analysées. Chaque étape doit posséder un titre, un marqueur défini, un commentaire valide et une date égale ou postérieurs à celle de l’étape précédente. Au fur à mesure que les informations sont analysées, les champs leur correspondant sont indiqués comme ayant une erreur. Pour des questions de visibilité, les panneaux des étapes contenant des erreurs sont également mis en surbrillance. Si lors de la validation, aucune erreur n’est détectée, le tracé va être récupéré. Sinon, le traitement s’arrête là.

##### Récupération du tracé

Une fois la certitude que toute les informations textuelles et géographiques sont correctes, le tracé peut être récupérer. Le tracé étant stocké sur le serveur en un objet unique, il est nécessaire de récupérer tous les tracés définis et de les combiner. Cette étape nécessite la récupération des composantes du tracé. L’image présentée ci-contre indique l’emplacement où l’on peut trouver tous les points composant le trajet, dans l’objet retourné par le géocodeur. Dans le cas où les tracés ne sont que des Polyline géodésiques, leurs seuls points (ceux de départ et d’arrivée) sont disponibles à l’autre emplacement indiqué sur l’image.

Le paramètre leg de l’objet contenant plusieurs objets, qui contiennent eux-mêmes des listes de points géographiques, les routes nécessitent une autre étape pour en extraire toute les informations. En extrayant chaque liste de points de chaque « leg » de chaque route et le les points du « path » des Polylines géodésiques, le tout dans l’ordre, on obtient une liste de points permettant à un constructeur de Polyline Google Map de reconstruire le tracé prévisualisé sur la carte. C’est précisément ce constructeur qui va être stocké sur le serveur. Ainsi, au fur et à mesure qu’on extrait les points, on les ajoute à une chaine de caractères accompagnée des séparateurs nécessaires pour qu’à la fin de la lecture, on obtienne la syntaxe d’un tableau JSON valide.

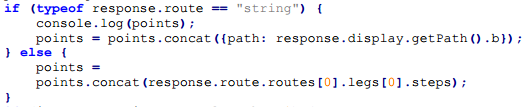


Figure  : Concaténation des composant d’une Polyline en haut et d’une route en bas

##### Envoi des informations

Une fois les informations récupérées, validées, le tracé récupéré et converti en constructeur, les informations sont prêtes à être envoyées au serveur. Le tableau contenant toute l’information est encodé sous forme de string, et est envoyé par un call AJAX. Le tracé sous forme de string, le titre, ainsi qu’une variable booléenne nommée « insert », indiquant la nature de la demande, est également ajouté aux paramètres du call.

Côté serveur, c’est d’abord le voyage qui sera ajouté à la base de données. Son tracé, qui est trop volumineux pour être stocké dans les champs d’une table, sera stocké dans un fichier texte, sur le serveur, auxquels on attribue un nom aléatoire. Le nom sera récupéré et référencé dans le champ « pathObject » du nouveau voyage. Après avoir été décodées, les autres informations sont insérées une à une. D’abord, le voyage est inséré, puis chaque étape du voyage est ajoutée à la base de données en récupérant sa clé primaire auto-incrémentée. Lors l’ajout d’une étape, un cookie portant comme clé le numéro que portait l’étape dans l’interface de création et comme valeur son identifiant dans la base de données est généré. Ce cookie durera dix seconde, et servira lors de l’insertion des images.

Les dates seront, avant d’être insérées dans la base de données, modifiées, le format de la date étant « jj/mm/aaaa » chez le client et « aaaa-mm-jj » dans la base de données. La date est découpée en trois particules, l’ordre de ses particules est inversé, puis une chaine de caractère est reconstruite en ajoutant un tiret entre les particules. C’est cette chaine qui sera inséré dans la base de données.

L’ensemble des actions effectué côté serveur se fait sous la forme de transaction. Si une erreur survient durant la procédure, toutes les modifications à la base de données seront annulées. Cette méthode permet, d’éviter que des informations incomplètes soient insérées : ces dernières pourraient poser des problèmes lorsqu’on tenterait de les afficher.

##### Envoi des images

Le composant utilisé pour la sélection d’image permet, entre autres, d’envoyer les images via AJAX à une adresse spécifiée lors de son initialisation. L’inconvénient de cette méthode est le suivant : il n’est pas possible de passer dans la requête l’identifiant de l’étape tout juste créée, sous peine de réinitialiser le composant. C’est pour palier à ce défaut que des cookies de très courte durée sont initialisés lorsque les étapes sont insérées dans la base.

Ainsi, lorsque l’insertion des informations dans la base est confirmée, le client envoie les images au serveur à leur tour. L’appel au serveur n’est pas une fonction que j’ai moi-même développée, elle est simplement appelée. L’envoi de l’image individuellement pour chaque composant. Chaque composant possède justement dans les paramètres de son appel au serveur l’identifiant de son composant. Le serveur peut ainsi récupérer grâce à ce numéro l’identifiant réel de l’étape dans la base. Les images peuvent ainsi être stockée dans le serveur et référencée dans la base.



Figure  : Commande d’upload d’un input

### Navigation

#### Pagination

Pour des questions de lisibilité, la navigation est décomposée en pages de cinq voyages chacune. Une restriction de cinq voyages chargés à la fois permet également de diminuer le temps de traitement en ne traitant pas trop d’informations à la fois.

Le nombre de pages est récupéré grâce à un appel au serveur, qui lui-même interroge la base de données sur le comptage du voyage de l’utilisateur. Le serveur retourne le nombre de page nécessaire, et le client génère des liens en conséquence, pour permettre la navigation.

#### Chargement des voyages

Les voyages sont chargés par groupe de cinq, selon leur pagination. Ces cinq voyages sont sélectionnés par le serveur grâce au numéro de page passé en paramètre et de l’identifiant de l’utilisateur stocké dans la session. Ce dernier retourne un tableau à plusieurs dimensions, contenant les voyages, qui contiennent eux-mêmes les informations sommaires de leurs étapes.

Une fois les informations récupérées, cinq panneaux sont générés selon un code de couleur, labélisés par les titres des cinq voyages. Dans ces cinq panneaux, représentant respectivement les cinq voyages, sont générés des liens vers leurs étapes.

Le serveur retourne également les tracés des cinq voyages. Ceux-ci sont affichés dans un code de couleur similaire à celui des panneaux, afin de les reconnaitre facilement. Ces tracés sont stockés dans une variable globale Des marqueurs sont également générés à partir des coordonnées géographiques de chaque étape. Ils stockés dans une autre variable globale, en deux dimension cette fois : la première dimension gradue les voyages, la seconde les étapes.

#### Chargement des détails

Lorsqu’une étape d'un voyage est sélectionnée, le panneau de navigation se ferme et un panneau, à droite, apparait. A l'aide de l'id de l'étape stockée à la fois dans le marqueur lui correspondant et le lien dans la navigation, on récupère les informations détaillées de l'étape ainsi que les médias qui lui correspondent. Pour les informations, les composants statiques du panneau sont simplement modifiés. Pour les médias, dans le cas où l'étape en possède, un carrousel est entièrement généré. Dans le cas où cette dernière n'en possède pas, la section dédiée au média masquée.

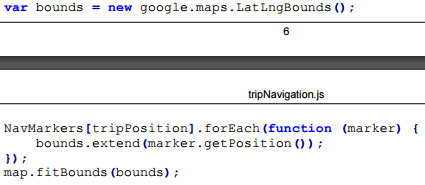
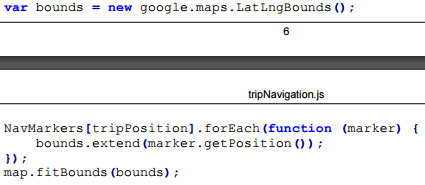
#### Cohésion des composants de navigation

Bien que la possibilité de zoomer et de bouger la carte paraissent très attractives, elles peuvent être aussi un obstacle à la clarté de l’information. Peu importe si un tracé s’affiche si la Map n’est pas cadrée dessus : l’utilisateur ne le verra tout simplement pas. De la même manière, si trop d’informations se mélangent sur la carte, il devient difficile de les distinguer. La pagination et le code de couleur contribuent déjà la clarté de l’affichage, mais ne sont pas suffisants. Un autre moyen de palier à ce manque de clarté est de restreindre l’affichage strictement aux éléments sélectionnés.

Cette stratégie est d’abord adoptée à l’échelle des voyages. Pour les panneaux correspondants aux voyages, un seul peut être ouvert à la fois : l’ouverture de l’un des panneaux entrainerait la fermeture de l’autre. Pour mettre en relation le panneau avec le tracé, lorsqu’un voyage est sélectionné, les tracés autres que celui de ce voyage disparaissent et la Map règle son centre et son zoom de façon à contenir tout juste toute les étapes du voyage. Les marqueurs de ses étapes, jusque-là existantes mais invisibles, sont affichés. Toutes les actions sont décrites sont d’ailleurs réciproque au tracé comme au panneau. La sélection de l’un ou de l’autre entraine ces mêmes changements.

Une fois les marqueurs affichés, l'utilisateur pourra afficher les détails d'une étape, soit en cliquant sur un marqueur, soit en sélectionnant une étape contenue dans un panneau. Si le marker contient l'identifiant de l'étape, le lien ne contient que l’index de son marqueur dans la variable globale ou ils sont tous stockés. Il doit donc retrouver le marqueur pour l’interroger sur l'id. Une fois l’information récupérée, les informations détaillées de l’étape peuvent être récupérées depuis le serveur, et le panneau contenant les détails peut être ouvert. Lorsque le panneau s'ouvre, le zoom augmente et la carte se centre sur le marquer de l'étape.

Figure  : Utilisation d’un Bound



Inversement, lorsqu’on retourne depuis l’affichage des détails au panneau de navigation, la Map se place de nouveau par rapport au voyage dans son entièreté. Et lorsque le panneau du voyage est fermé, elle se positionne par rapport à tous les tracés présents sur la page.

Cette adaptation de la carte est permise par le « bounds ». C’est un objet qui exprime une zone de la carte, et que l’on peut étendre en y incluant un nouveau point. La Map possède une fonction lui permettant d’être adaptée à un « bounds », et c’est en paramétrant ce bound avec les marqueurs de tout le voyage ou uniquement ceux d’un voyage précis que l’on peut adapter la carte à un tracé. Pour se centrer sur un unique marqueur, c’est uniquement le zoom qui est changé manuellement et le centre de la Map qui est adapté à la position du marqueur.

### Modification

#### Pré-remplissage des champs

Lorsque l’utilisateur demande à modifier un voyage, c’est l’interface de création qui est ouverte. L’id du voyage est stocké dans une variable globale, et les informations de ce voyage sont récupérées sur le serveur. Le champ de titre est alors complété et des panneaux sont générés pour chaque étape, en complétant les champs correspondants aux détails du voyage. Des marqueurs, auxquels on ajoute l’identifiant de l’étape, sont également crées. Cet identifiant sera conservé même quand le marqueur sera modifié par l’utilisateur. Les routes reliant les marqueurs entre eux sont tracées au fur et à mesure en utilisant le service géocodeur.

Les images sont également chargées, grâce à la fonctionnalité Preview de l’input modifié. Les images en Preview, lorsqu’elles sont supprimées par l’utilisateur, envoient leur Key (ici initialisée avec l’id des images) à un call AJAX spécifié dans les paramètres avant de disparaitre. Elles ne seront également pas envoyées au server lors de l’upload. Un input contenant des images prévisualisées requière un certain nombre de paramètre supplémentaires, qu’on peut voir dans l’image.

#### Traitement des informations

La récupération et la validation des informations n’est pas fondamentalement différente que lors de l’ajout d’un nouveau voyage. Les identifiant contenu dans les marqueurs en possédant sont simplement ajoutés aux autres informations. Une fois à l’étape de l’envoi au serveur, l’identifiant est ajouté aux informations envoyées et un autre paramètre explicitant que les informations sont modifiées est également inclus.

Le serveur récupère les informations envoyées par le client, puis charge les informations actuelles du voyage depuis la base de données. Ces informations servent de référence. Il parcourt ensuite les étapes envoyées par le client, une par une. Si l’étape ne possède pas d’identifiant, elle est ajoutée à la base. Dans le cas contraire, ses informations sont mises à jour dans la base de données et sa version de référence prend une valeur nulle. Lorsque toutes les étapes du client ont été traitées, on parcourt la valeur de références : celles qui n’ont pas pris une valeur nulle ont été supprimées par l’utilisateur. Elles sont donc supprimées, ainsi que leur média. Le voyage, lui, est simplement mis à jour, et son constructeur est réécrit dans le fichier texte lui correspondant.

#### Gestion des images

La gestion des images se fait en deux temps, lors de la modification. Lorsque l’utilisateur supprime une image existante, l’appel est directement fait au serveur. Au contraire, lorsqu’une nouvelle image est ajoutée, l’upload ne se fait que lors de la validation du voyage entier. Les images dont on commande la suppression sont mises « en attentes » dans la session. Cette fille d’attente est réinitialisée à chaque fois qu’un voyage est ouvert en modification, en plus d’être vidée à chaque fois qu’un voyage est modifié ou que sa modification est abandonnée. La suppression des images se fait donc en même temps que le traitement des autres informations, en récupérant id des images à supprimer dans la session.

Le cas de l’insertion d’image se passe exactement comme lors de l’ajout d’un voyage : vu que les images pré visualiser ne sont pas envoyées au serveur, les seules images qui y parviendront devront automatiquement être ajoutées. Ainsi, l’étape, traitée par le serveur lors de la modification, tout comme lors de la création, entraine la création d’un cookie de courte durée, pour permettre la récupération de l’id lors de l’upload d’image.

### Suppression des voyages

Les voyages sont supprimés par appel au serveur, après confirmation du client. La suppression s’effectue dans cet ordre : Les étapes du voyage sont parcourues. Pour chaque étape, chaque média voit son lien dans la base de données supprimé puis son image supprimée du serveur. L’étape est ensuite supprimée. Lorsque toutes les étapes ont été supprimées, le voyage est lui-même supprimé, ainsi que le fichier texte contrant le constructeur de ce voyage.

### Sécurité

#### Sécurité de la Map

Pour des questions d’esthétique mais aussi de sécurité, il serait préférable que l’utilisateur ne puisse pas sortir des limites de la carte, au nord et au sud. La première contrainte à ce niveau est imposée à la Map lors de sa création. Le niveau de zoom minimum est augmenté de manière à pouvoir contenir toute la carte sans déborder sur les bords. Malheureusement, le niveau de zoom est la seule sécurité native, et il est à lui tout seul insuffisant : avec n’importe quel niveau de zoom, on peut déplacer la carte à l’aide de la souris et sortir de ses limites. Il faut donc mettre en place un filtrage déclenché par de tel action.

Plusieurs évènements sont déclenchés lors du déplacement de la carte, mais celui qui permet le plus grand contrôle est l’évènement lié au changement du centre de la Map. Pour sécuriser, on récupère lors de cet évènement, la surface que couvre la carte à cet instant sous forme de Bound. Ce Bound contient deux paires de coordonnées : les latitude haute et basses et les longitudes hautes et basses. Nous nous intéresserons uniquement aux latitudes. Dans ce cas précis, pour que la carte ne déborde pas, la latitude basse ne doit pas être inférieure à -85 et la latitude haute ne pas être supérieure à 85. Si l’une de ces contritions n’est pas respectée, le centre de la carte est décalé en fonction de l’écart constaté, afin que le nouveau placement respecte à nouveau ces deux conditions.

Il reste un cas de figure ou l’utilisateur peut sortir de la carte : c’est en se plaçant à un zoom élevé au bord de la carte, puis en diminuant le zoom. Néanmoins, il n’est pas nécessaire de placer un autre évènement lié au changement de zoom, car bien que le centre reste à première vue le même lors d’une tel action, l’évènement lié au changement de centre est tout de même déclenché. Ainsi, dans ce cas de figure, lorsque l’utilisateur diminue le zoom, le centre s’approchera graduellement de l’équateur de manière à maintenir l’affichage dans les limites autorisées.

#### Sécurité des champs

Pour éviter que des information invalide soit insérées, ou pire, que des injections SQL puissent être pratiquées, le site se doit d’être sécurisé, dans la mesure du possible, sur tous les niveaux. La première vérification des données se fait lorsque les informations du formulaire sont récupérées. Pour les champs textuels, ils doivent simplement ne pas être vides. Pour la date, elle doit être validée par une expression régulière et doit également être identique ou postérieure à la date de l’étape précédente, si cette étape existe.

Pour le call AJAX, on ne passe pas l’information par l’url, enlever la possibilité à un utilisateur malveillant la possibilité d’entrer des informations manuellement via l’URL. Côté serveur, on utilise des bindParam pour sécuriser les informations une dernière fois, avant dès les insérer dans la base de données.

Vu que l’identifiant est récupéré peut être falsifié par un utilisateur malveillant, la moindre des choses est d’empêcher cet utilisateur d’opérer sur une information ne lui appartenant pas. Ainsi, pour chaque requête n’impliquant pas une insertion, un paramètre supplémentaire est ajouté : l’identifiant de l’utilisateur, stocké dans la session.

# Test de l’application

## Stratégie de test

Tester une application usant de nombreux éléments externes n’est pas une mince affaire, car aux failles de l’application elle-même s’ajoute les failles et particularités des éléments externes utilisés. À titre d’exemple, si on tente de rentrer une adresse vide dans un Geocoder Google Map, au lieu de nous envoyer une erreur, il va nous renvoyer la position d’un endroit perdu aux milieux des campagnes américaines. Pour cette raison, tester de façon exhaustive une application contenant une Google Map et usant de sa fonctionnalité est quasiment impossible. C’est pour cette raison que j’ai orientée mes tests plutôt vers la fonctionnalité que j’ai entièrement développé que celles usant d’éléments externes.

Mais même en orientant les tests vers les fonctionnalités exclusives du site, les tests restent très nombreux et un rapport de test complet offre plus de confusion que de clarté sur la question : le site est-il fonctionnel ? Ainsi, j’ai préféré conserver mes rapports de test complets en dehors de la documentation technique et fournir à cette dernière un « rapport de test restreint ». Ce dernier donne un meilleur aperçu du fonctionnement global du site, mais ne garantit pas son infaillibilité.

## Rapport de test restreins

# Conclusion

## Justesse de la planification

### Planning initiale

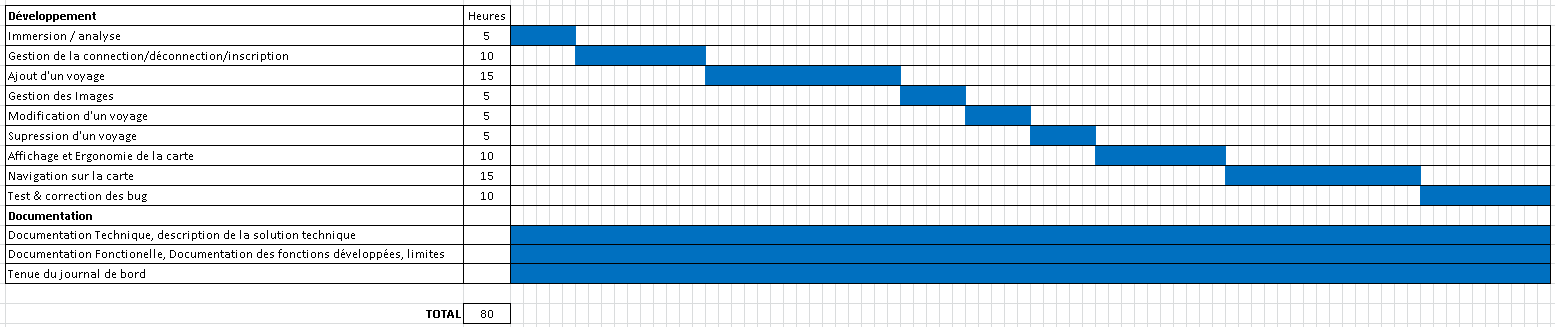


Figure  : Planning prévisionnel

### Planning réel

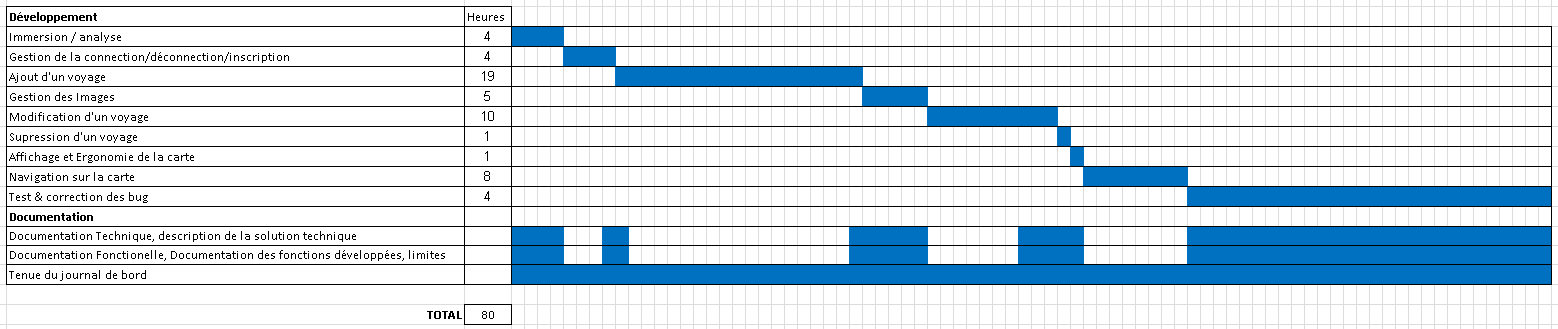


Figure  : Planning réel

### Commentaire sur la planification

Lorsque le travail a été planifié, c’est la carte de la sécurité qui a été jouée. En effet, plutôt que d’allouer le temps minimal ou approximatif à une tâche, j’ai plutôt opté pour le temps que prendrait cette tâche, avec tous ses contre temps, tous les problèmes qu’elle peut impliquer. On peut d’ailleurs voir qu’entre la planification initiale et la planification finale, il n’y a qu’une tâche de développement qui a dépassé cette estimation. Mais planifier large n’est pas plus mal qu’autre chose : en particulier dans le cadre de ce projet : l’usage d’une quantité importante de technologies différentes signifie potentiellement plus de problèmes à régler et de bug à résoudre à différents niveaux.

On peut également voir que les planifications initiales et finales par rapport à la documentation diffèrent. Alors que j’avais prévu de mettre à jour ma documentation tout au long du projet, je l’ai en réalité rédigé par session. En réalité, comme les quatre premiers points de la documentation ont été rédigés le premier jour, il ne restait plus qu’à compléter l’analyse de contenu et conclusion… Que j’ai préféré rédigé une fois la tâche lui correspondant terminée, pour éviter de perdre du temps à expliquer une méthode pourrait être changée par la suite.

## Modèle conceptuel

Pour des raisons de performances, le modèle conceptuel final diverge légèrement du modèle conceptuel initial. Les marqueurs, très volumineux et ne requérants pas d’appel au server lors de leur initialisation, n’avaient aucun intérêt à être stockés dans la base de données. Le champ « marker Object » a été remplacé par des champs « lat », « lng » et « address ».

## Critique du résultat obtenu

## Améliorations possibles

Bien que le site soit terminé selon le cahier des charges, il reste une quantité de fonctionnalité qu’on pourrait lui ajouter pour l’améliorer. Il y aurait par exemple l’aspect social que j’avais initialement voulu développer avec le reste de l’application, mais a finalement été retiré du cahier des charges sous peine de rendre celui-ci trop lourd. En effet, TripTracker étant construit sur un principe similaire à celui d’un blog, il ne lui manquerait plus que des interactions entre les utilisateurs pour en être un. Dans un premier temps, un utilisateur pourrait accéder au contenu d’un autre, avec sa permission. Les utilisateurs pourraient également réagir aux voyages des autres, en les commentant per exemple. La finalité de la mise en place d’interaction sociale pourrait même aller jusqu’à entrer des voyages communs à plusieurs utilisateurs…

Mais à mon avis, l’un des premiers points à améliorer dans TripTracker serait son design sur mobile. Pour un site parlant de voyager, rien de plus naturel que l’avoir dans sa poche !

Un autre aspect du site qui pourrait être développé est le système de route qu’il intègre. C’est finalement ce qui le démarque de la « concurrence » : Il se concentre plus sur le chemin parcouru, il se présente sur une carte au lieu de se présenter comme un simple journal. L’API Google Map offre de nombreuses fonctionnalités natives, comme la possibilité de choisir le mode de transport lorsqu’on veut relier un point A à un point B. Il permet également d’éditer manuellement le tracé, simplement à la souris.

La dernière finalité qui, à mon sens, pourrait rendre cette application attractive serait la possibilité de la mettre en relation avec le GPS d’un téléphone. A ce stade, l’application ne pourrait plus uniquement fonctionner sur un support web, et c’est pour ça que c’est évolution est la plus hypothétique. L’application alors largement orientée vers les smartphones. Elle permettrait d’enregistrer un tracé en temps réel, en récupérant régulièrement la localisation du smartphone.

## Intérêt personnel et gain de connaissance

Je dois avouer qu’une grande part de ma motivation résidait dans le fait qu’il n’existait pas d’application web similaire en ligne. Je trouve qu’il n’y a rien de plus déprimant que de développer une application tout en sachant qu’on est à des années-lumière de la concurrence. Comme je l’ai dit dans mon analyse concurrentielle, il n’est pas question de but lucratif. Il est juste question d’intérêt personnel.

# Bibliographie

<http://php.net/>

<http://api.jquery.com/>

<http://www.malot.fr/bootstrap-datetimepicker/demo.php>

<http://plugins.krajee.com/file-input>

<http://getbootstrap.com/css/>

<http://getbootstrap.com/javascript/>

<https://dev.mysql.com/doc/>

<https://developers.google.com/maps/documentation/javascript/>

https://developer.mozilla.org/fr/docs/Web/CSS/Reference

<https://stackoverflow.com/questions/1359018/in-jquery-how-to-attach-events-to-dynamic-html-elements>

<https://stackoverflow.com/questions/26750590/call-an-event-on-bootstrap-panel-expand>

<https://stackoverflow.com/questions/3818016/google-maps-v3-limit-viewable-area-and-zoom-level>

<https://stackoverflow.com/questions/11719495/php-warning-post-content-length-of-8978294-bytes-exceeds-the-limit-of-8388608-b>

<http://themapguyde.blogspot.ch/2014/08/bootstrap-map-viewer-templates.html>

<http://www.easyphp.org/>

<https://www.gliffy.com>

<https://triptracker.mybalsamiq.com/>

# Table des figures

[Figure 1 : Exemple de voyage que l’on peut publier sur Facebook 4](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546671)

[Figure 2 : Extrait d’un journal de voyage du site Travel Diaries 4](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546672)

[Figure 3 : Un voyage de MyTripJournal avec sa description et ses étapes 4](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546673)

[Figure 4 : Navigation dynamique et tracés, une fonctionnalité de Travel Pod 5](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546674)

[Figure 5 : Modèle conceptuel du projet soumis avec le cahier des charges 7](#_Toc485546675)

[Figure 6 : Arborescence des vues dans l’application 8](#_Toc485546676)

[Figure 7 : Maquette de la page d’accueil avec ses modales d’inscription et de connexion 8](#_Toc485546677)

[Figure 8 : Maquette de la page personnelle de l’utilisateur avec ses trois vues 9](#_Toc485546678)

[Figure 9 : Organisation à la racine du projet 11](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546679)

[Figure 10 : Représentation graphique des dépendances 12](#_Toc485546680)

[Figure 11 : Exemple fourni par l’auteur du DateTimePicker 13](#_Toc485546681)

[Figure 12 : Exemple du File Input 13](#_Toc485546682)

[Figure 13 : Évènement click de la carte 15](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546683)

[Figure 14 : Syntaxe de l’appel au service géocodeur Google Map 15](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546684)

[Figure 15 : Suppression de marqueur et attribution de tracé dans trois cas 16](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546685)

[Figure 16 : Instance d’une Polyline géodésique 17](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546686)

[Figure 17 : Paramétrage des routes dans l’application 17](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546687)

[Figure 18 : Concaténation des composant d’une Polyline en haut et d’une route en bas 18](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546688)

[Figure 19 : Commande d’upload d’un input 19](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546689)

[Figure 20 : Utilisation d’un Bound 20](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546690)

[Figure 21 : code de la fonction sécurisant la carte 22](file:///D:\School\TPI\CloneRepo\sadekTPI\Documentation\DocTechique.docx#_Toc485546691)

[Figure 22 : Planning prévisionnel 24](#_Toc485546692)

[Figure 23 : Planning réel 24](#_Toc485546693)