|  |
| --- |
| TPI 2017 |
| TripTracker |
| Serena Sadek |

Table des matières

[1 Introduction 3](#_Toc485289059)

[1.1 Organisation 3](#_Toc485289060)

[1.2 Objectifs 3](#_Toc485289061)

[2 Analyse concurrentielle 4](#_Toc485289062)

[2.1 Facebook 4](#_Toc485289063)

[2.2 Travel Diaries 4](#_Toc485289064)

[2.3 MyTripJournal 4](#_Toc485289065)

[2.4 Travel Pod 5](#_Toc485289066)

[2.5 Positionnement par rapport à la concurrence 5](#_Toc485289067)

[3 Cahier des charges 6](#_Toc485289068)

[3.1 Définition de l’audience 6](#_Toc485289069)

[3.2 Définition du contenu 6](#_Toc485289070)

[3.3 Fonctionnalités 6](#_Toc485289071)

[3.4 Modèle conceptuel 7](#_Toc485289072)

[3.5 Maquette 8](#_Toc485289073)

[3.5.1 Page d’accueil 8](#_Toc485289074)

[3.5.2 Page principale 9](#_Toc485289075)

[4 Méthodologie 11](#_Toc485289076)

[4.1 Outils utilisés 11](#_Toc485289077)

[4.1.1 PHP 11](#_Toc485289078)

[4.1.2 MySQL 11](#_Toc485289079)

[4.1.3 JavaScript / JQuery 11](#_Toc485289080)

[4.1.4 AJAX 11](#_Toc485289081)

[4.1.5 Bootsrap 11](#_Toc485289082)

[4.1.6 Google Map API 11](#_Toc485289083)

[4.2 Charte Graphique 12](#_Toc485289084)

[5 Analyse du contenu 13](#_Toc485289085)

[5.1 Arborescence 13](#_Toc485289086)

[5.1.1 Organisation 13](#_Toc485289087)

[5.1.2 Rôle 13](#_Toc485289088)

[5.1.3 Dépendances 13](#_Toc485289089)

[5.2 Usage d’éléments externes 13](#_Toc485289090)

[5.2.1 Bootstrap DateTimePicker 13](#_Toc485289091)

[5.2.2 Bootstrap File Input 14](#_Toc485289092)

[5.3 Fonctions remarquables 14](#_Toc485289093)

[5.3.1 Inscription 14](#_Toc485289094)

[5.3.2 Connexion 14](#_Toc485289095)

[5.3.3 Interface de création 15](#_Toc485289096)

[5.3.4 Gestion des Routes 15](#_Toc485289097)

[5.3.5 Enregistrement des informations 17](#_Toc485289098)

[5.4 Navigation 19](#_Toc485289099)

[5.4.1 Pagination 19](#_Toc485289100)

[5.4.2 Chargement des voyages 19](#_Toc485289101)

[5.4.3 Chargement des détails 19](#_Toc485289102)

[5.4.4 Cohésion des différents composants 19](#_Toc485289103)

[5.5 Modification 20](#_Toc485289104)

[5.5.1 Chargement des informations 20](#_Toc485289105)

[5.5.2 Traitement des informations 21](#_Toc485289106)

[5.5.3 Gestion des images 21](#_Toc485289107)

[5.6 Suppression des voyages 21](#_Toc485289108)

[5.7 Sécurisation de la Map 21](#_Toc485289109)

[6 Test de l’application 23](#_Toc485289110)

[6.1 Stratégie de test 23](#_Toc485289111)

[6.2 Aperçu d’un rapport de test 23](#_Toc485289112)

[7 Conclusion 24](#_Toc485289113)

[7.1 Justesse de la planification 24](#_Toc485289114)

[7.1.1 Planning initiale 24](#_Toc485289115)

[7.1.2 Planning réel 24](#_Toc485289116)

[7.1.3 Commentaire sur la planification 24](#_Toc485289117)

[7.2 Critique du résultat obtenu 24](#_Toc485289118)

[7.3 Améliorations possibles 24](#_Toc485289119)

[7.4 Intérêt personnel et gain de connaissance 24](#_Toc485289120)

[8 Bibliographie 25](#_Toc485289121)

[9 Table des figures 26](#_Toc485289122)

# 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

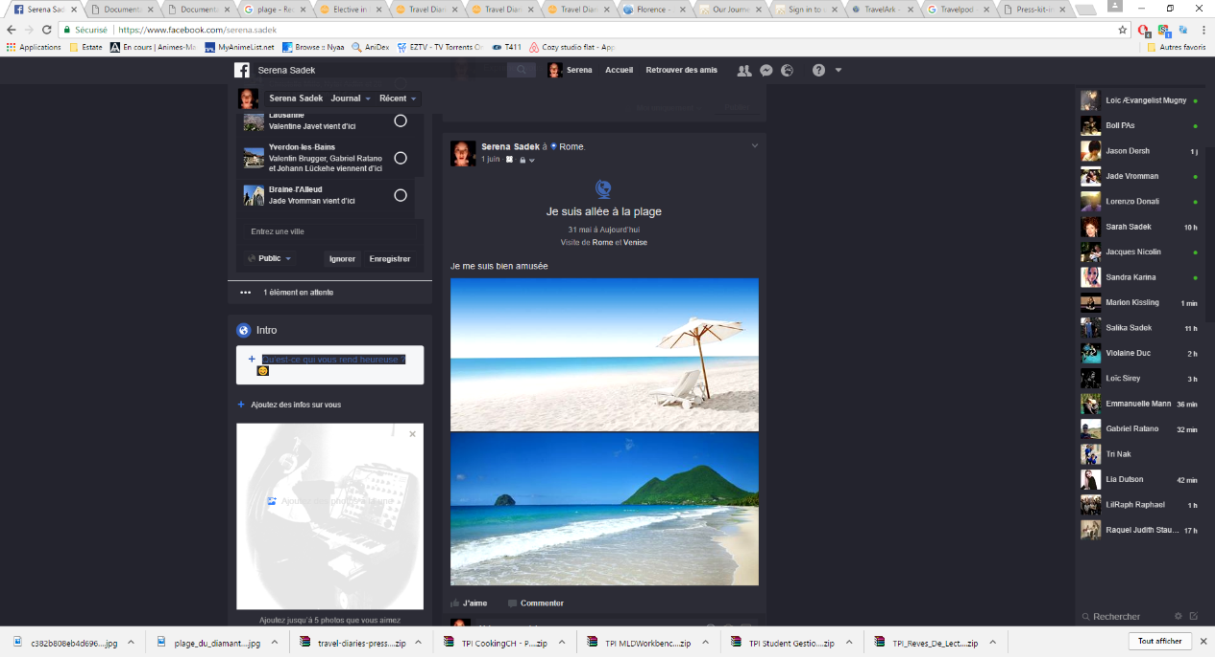


Figure  : Exemple de voyage que l’on peu publier sur Facebook

Dans cette analyse concurrentielle, je ne fais état que des sites spécialisés dans le même domaine que celui que je souhaite développer. 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 sites de voyages, 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. L’utilisateur ne connaissant pas la région sera obligé d’accéder à un site externe s’il veut en savoir plus sur l’endroit référencé sur une carte.

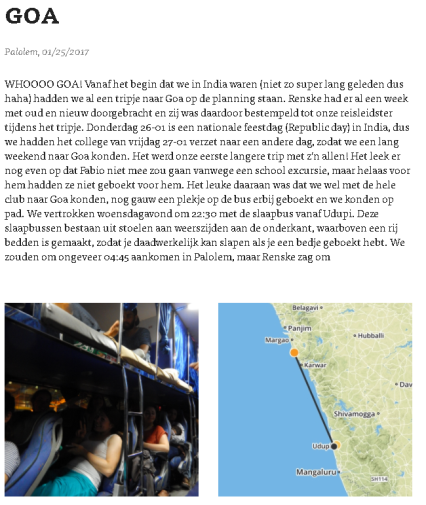


Figure  : Extrait d’un journal de voyage du site Travel Diaries

Ce site a également la particularité de gérer les informations dans leur ensemble, c’est-à-dire qu’images, cartes et texte sont placés les un à la suite des autres au gré de l’utilisateur.

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 plus proche 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 elle n’offre pas la possibilité de zoomer.



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

On accède aux autres articles sur une autre page. Le contenu des articles est uniquement textuel, mais une section est réservée aux images.

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.

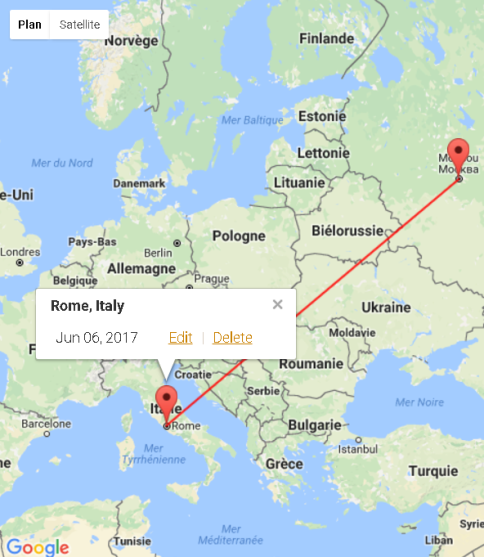


Figure  : Navigation dynamique et tracés, une fonctionnalité de Travel Pod

Malheureusement, 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 complets et le plus proche de l’approche que je souhaiterais avoir, avec une carte interactive, offrant la possibilité de naviguer. 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.

## 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 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 la plupart du temps 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.

# Cahier des charges

## 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.

## 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 créer et éditer ses étapes, mais doit avoir entré une localisation pour y ajouter une autre information. La localisation peut être entrée manuellement sous forme d’adresse, ou en cliquant sur la map.

## 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 étapes 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.
* Lorsqu’on clique sur le titre d’une étape ou son marqueur correspondant, les détails apparaissent dans la section correspondante.
* Si l’étape contient des images, elles apparaissent sous forme de carrousel
* Lorsqu’on ouvre la section de navigation, la section de détail se réduit, et réciproquement.
* Lorsque l’utilisateur commande l’ajout d’un nouveau voyage, tous les autres sections se ferment.
* 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’étape sélectionnée est en surbrillance.
* 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 importer un contenu textuel
* 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 depuis l’interface de création

## Modèle conceptuel

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

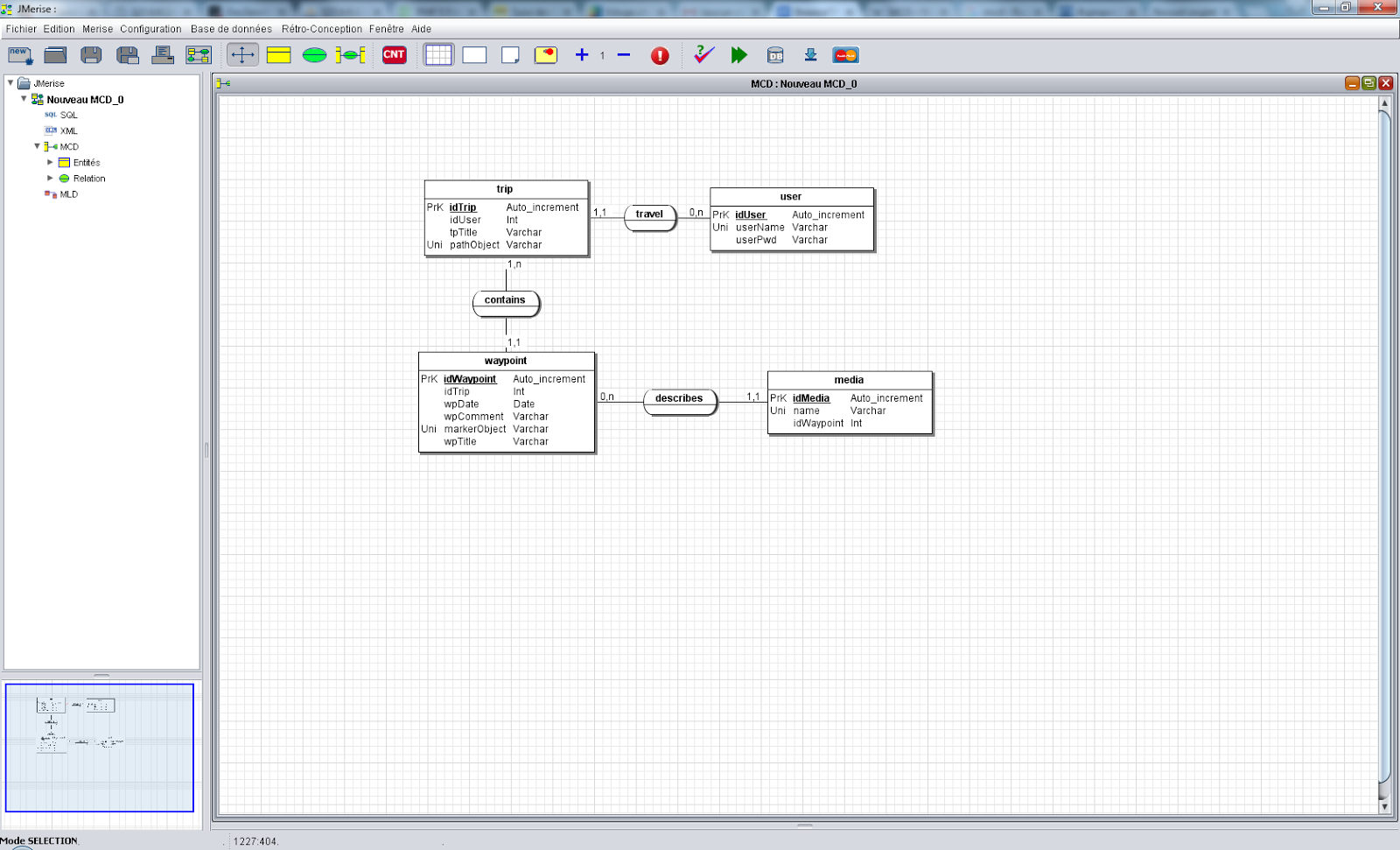


Figure  : Modèle conceptuel du projet

## Maquette

### Page d’accueil

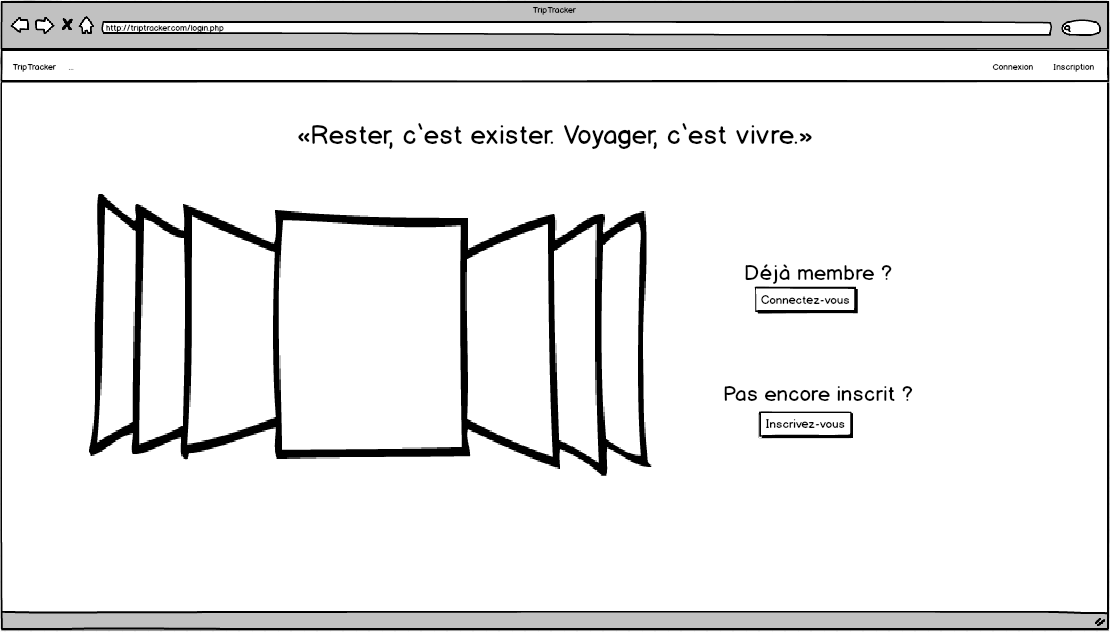


Figure  : Maquette de la page d’accueil.

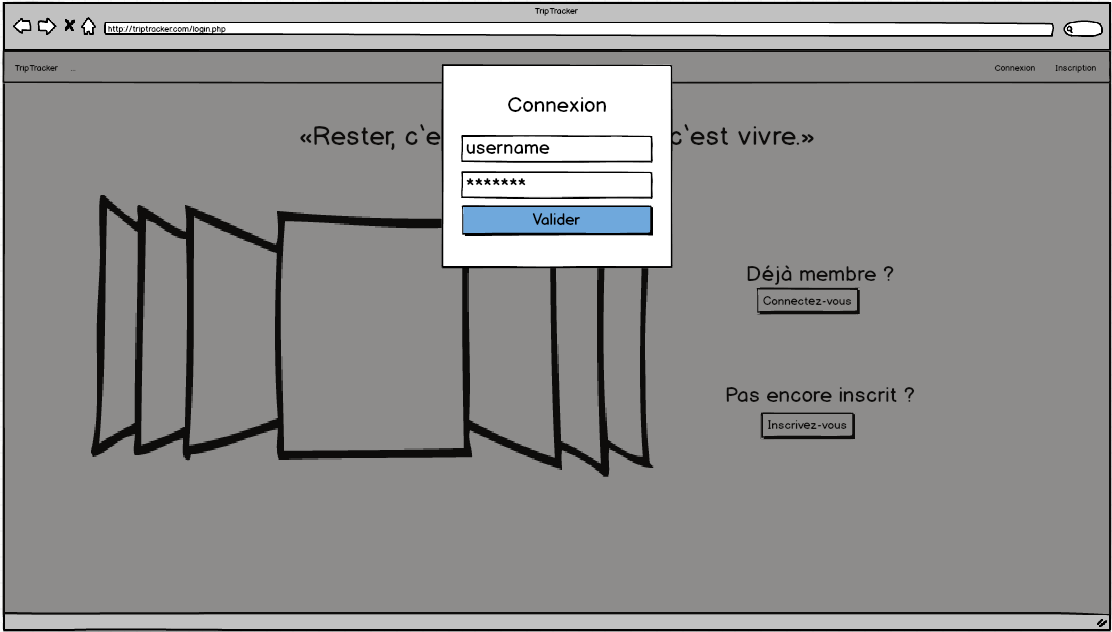


Figure  : Maquette de la page d’accueil avec sa modale de connexion

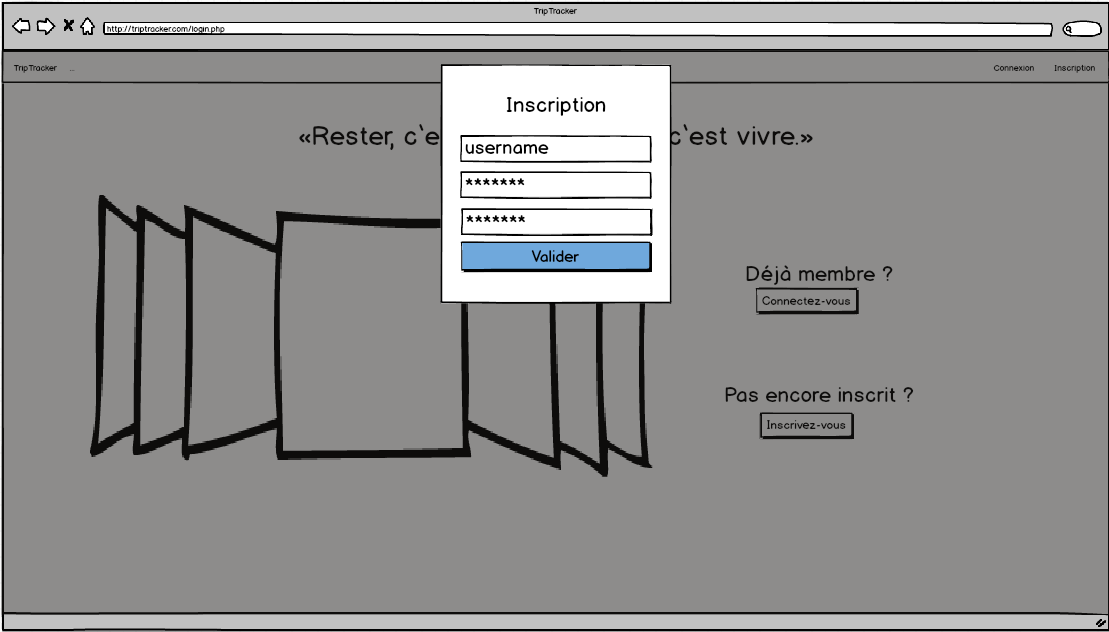


Figure  : Maquette de la page d’accueil avec sa modale d’inscription

### Page principale

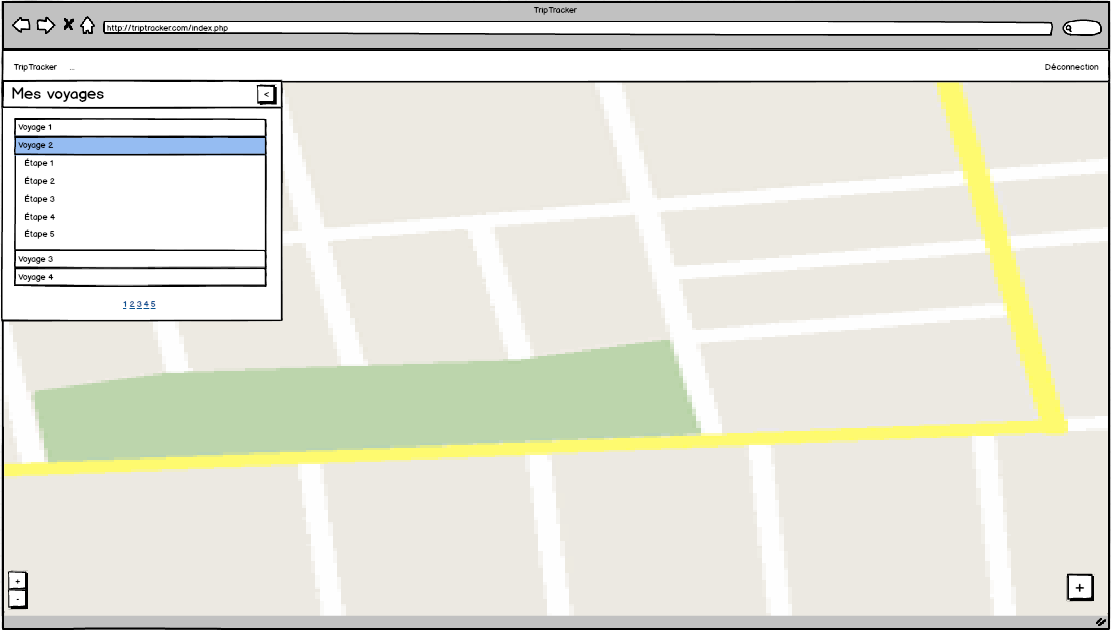


Figure  : Maquette de la page principale, lors de la navigation parmi les voyages et étapes

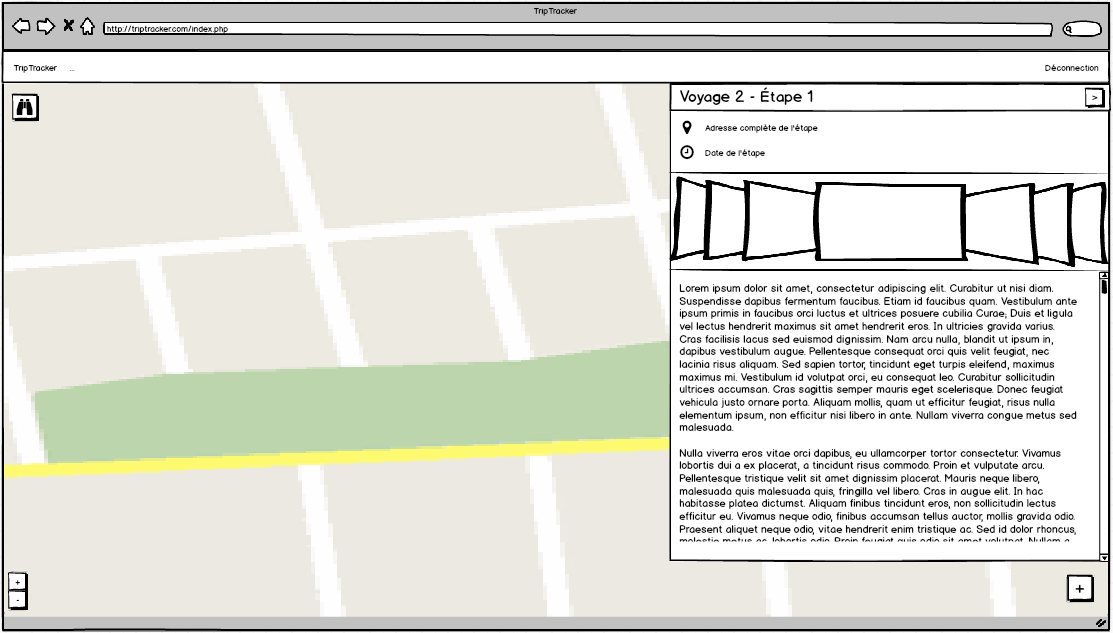


Figure  : Maquette de la page principale, lors de l’aperçu d’une étape

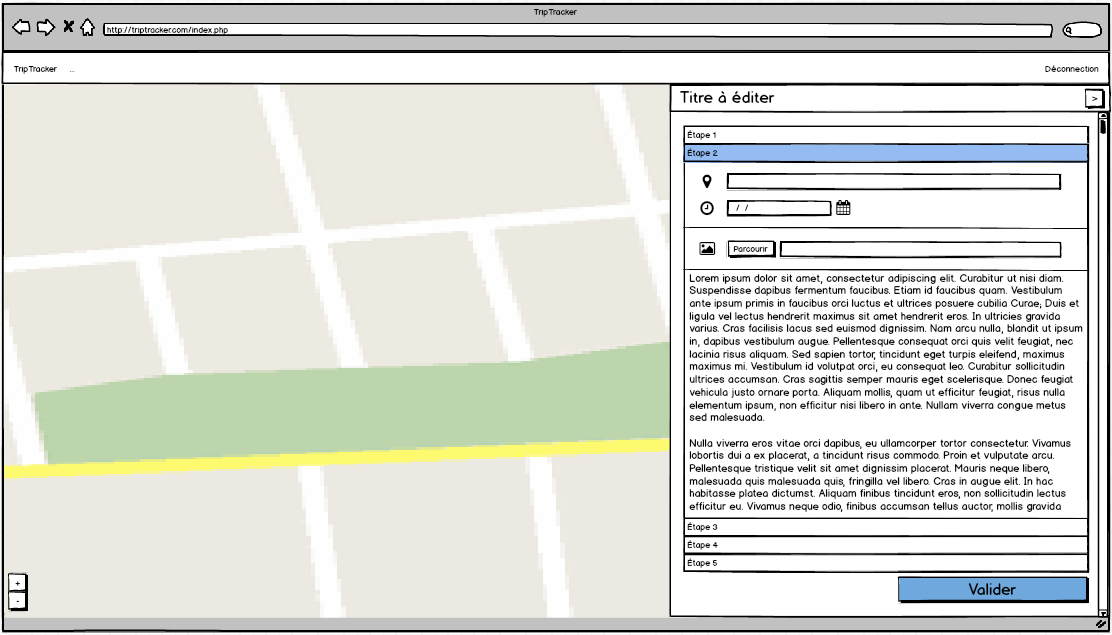


Figure  : Maquette de la page principale, lors de l’ajout ou de la modification d’un voyage

# Méthodologie

## Outils utilisés

Bien qu’à n niveau technique de CFC, le choix des outils se fasse plus par restriction que par séléction, il n’en est pas moins possible de tirer profit

### 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 dispensée, orientés respectivement vers la gestion de session, la connexion à une base de donnée à l’aide d’objet PDO et la gestion et le stockage de média.

Bien que j’aie entendu parler d’autres méthodes pour arriver au même résultat (notamment avec des Framework Basés sur JavaScript), ma maitrise de ces outils est bien insuffisante pour me reposer dessus dans un travail de cette ampleur. J’ai donc choisi d’utiliser PHP pour certains aspects (les trois cités ci-dessus) du site.

### MySQL

Ce langage servant à l’interaction avec les SGBD est, on peut dire, un incontournable. A vrai dire, je n’ai personnellement jamais entendu parler d’un langage tenant le même rôle qui ne soit pas une énième version de SQL.

La base de donnée étant l’un des éléments centraux fournissant son contenu au site web, à l’aide d’objet PDO, il m’est donc nécessaire d’avoir un usage récurent de MySQL.

### JavaScript / JQuery

Bien qu’aucun de ces deux langages ne soit dispensé sous forme de module, ce sont sans doute ceux avec lesquels j’ai le plus travaillé au cours de ma troisième années, et même peut-être ma deuxième. Ils ont l’avantage d’offrir un résultat rapide, dynamique, plus proche de ce qu’on pourrait attendre d’un site qui aurait sa place sur le marché.

Bien que développer avec ces langages soit plus compliqué qu’avec du PHP (gestion des erreurs plus difficile, problèmes liés au caractère asynchrone de leur fonctionnement et code plus difficile à structurer), ce sont eux qui font la promesse d’un résultat qui corresponde au mieux à mon cahier des charges. JQuery est également mon langage préféré…

### AJAX

Ouvrant les portes des requêtes asynchrones, c’est la solution aux temps de chargement qui n’en finissent plus.

### Bootsrap

### Google Map API

Indigeste au premier abord, Google Map API est en réalité un outil extrêmement puissant. Bénéficiant de surcroit d’une communauté active, de mise à jour régulières et d’un grand nombre de code sources, il permet en réalité d’arriver à un résultat bluffant sans trop de peine. Mais cette API n’a un unique gros défaut : comme tous les langages, Framework ou API puissants, des ralentissements se font vite sentirez si on ne se repose que sur ses capacité de calculs.

J’ai travaillé sur toute cette année scolaire sur un projet de groupe basé sur Google MAP API, et bien qu’il m’a donné du fil à retordre, j’ai fini par bien l’aimé (au point de baser mon travail de diplôme dessus). D’une part, il a le potentiel d’ajouter un aspect très attractif, dynamique et ergonomique au cahier des charges que je me suis donné. Mais une des choses qui m’a poussé à choisir un projet basé dessus est la promesse d’un résultat bluffant qu’on obtient si on réussit le chalenge qu’implique la conception.

## Charte Graphique

# Analyse du contenu

## Arborescence

### Organisation

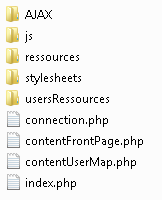


Figure  : Organisation des fichiers du projet

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.

### Rôle

|  |  |  |
| --- | --- | --- |
| **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 table « user » de 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). |
| 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. |
| 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 l’interface de création/modification (focus, génération d’onglet), par exemple. |
| modifPlugin.js | Permet à l’interface de création d’être utilisée pour la modification en chargeant le contenu du voyage de la base, et en générant des composants pré-remplis. |
| templateMap.js | Veille au bon fonctionnement du template : gère l’ouverture et la fermeture de panneaux, l’initialisation et la sécurité de la carte. |
| tripNavigation.js | Charge le voyage de l’utilisateur et génère les composants leur correspondant. Il gère également les interactions entre l’utilisateur et les composants dynamiques. |
| stylesheet | frontPage.css | S’occupe de la 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 à laquelle l’utilisateur accède lorsqu’il est connecté. |
| [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. |
| contentUserMap.php | Vue affichant contant la carte ainsi que ses trois panneaux. |
| index.php | Page web contenant de manière statique la navigation supérieure, et affichant différentes vues selon si l’utilisateur est connecté ou non. |

### Dépendances

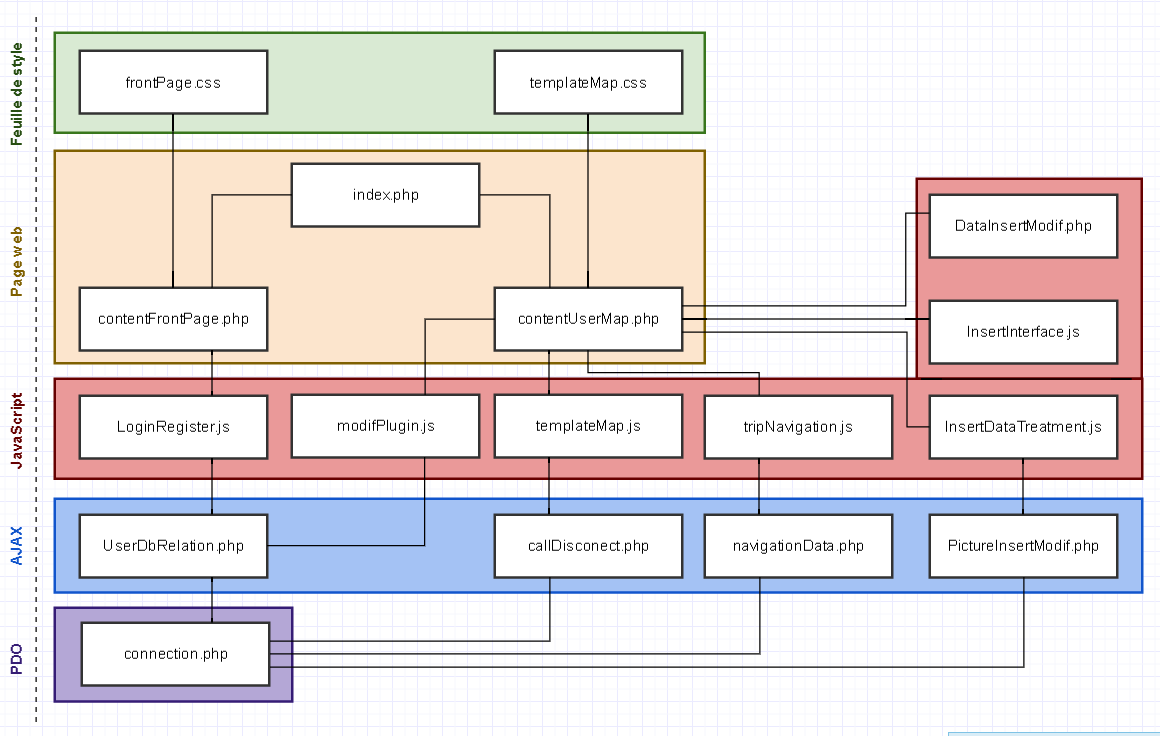


Figure  : Représentation graphique des dépendances

## Usage d’éléments externes

### Bootsrap DateTimePicker

Ce plugin de Bootstrap a été 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 pouvant apporter les réponses à certaines questions.

Cette version améliorée 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 simplicités, d’obtenir un composant ergonomique, permettant le drag and drop, et surtout permettant l’upload de fichier via AJAX.

## Fonctions remarquables

### Inscription

L’inscription s’effectue à l’aide d’un call AJAX. Une fois le nom d’utilisateur et le mot de passe récupéré, on vérifie qu’aucune entrée n’est vide (sa longueur est alors égale à zéro).

Si ce n’est pas le cas, et que le mot de passe et la confirmation correspondent, on envoie les informations à la base de données à l’aide du Call AJAX. Les informations récupérées par le PHP sont filtrées pour éviter les injections SQL. 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 PHP retourne un message au JavaScript, ainsi qu’un booléen indiquant la réussite de l’inscription. Si la requête auprès de la base échoue, c’est le message d’erreur récupéré par le Try/Catch qui sera retourné.

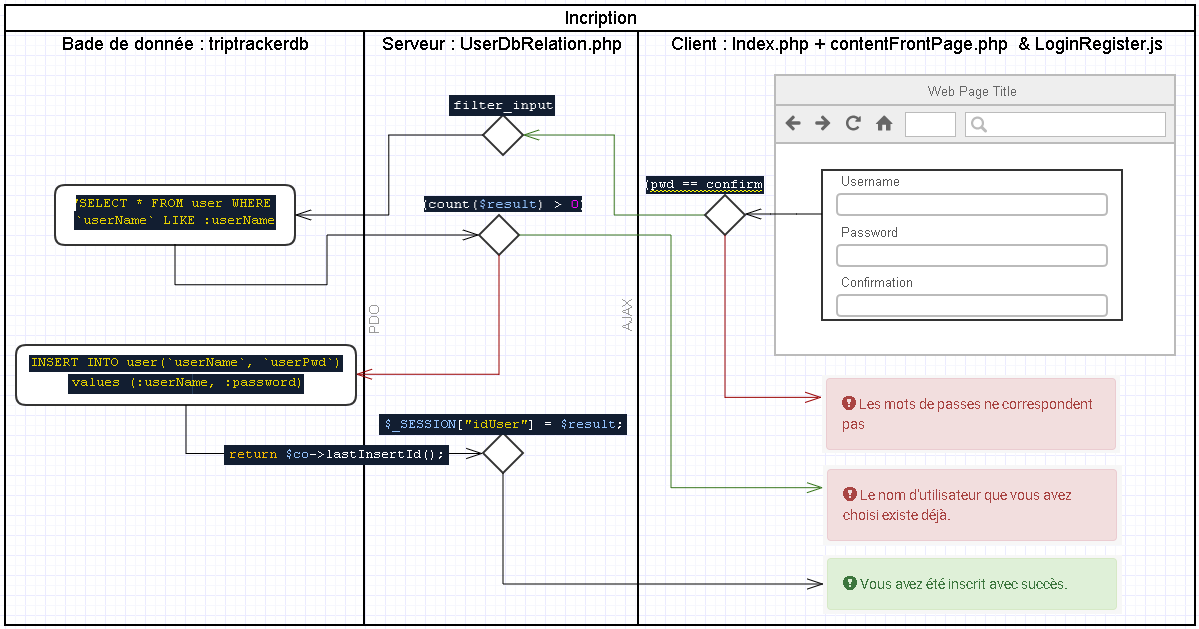


Figure  : Digramme représentant globalement le processus d’inscription

Lorsque le Call AJAX a récupéré sa réponse, il génère, dans le formulaire, une fenêtre d’information contenant le message retourné. La fenêtre d’information est rouge ou verte selon le booléen retourné par le Call. Si le booléen est à True, la page va s’actualiser au bout d’un instant et le nouvel utilisateur va se retrouver sur sa page personnelle.

### Connexion

La connexion s’effectue, tout comme l’inscription, à l’aide d’un Call AJAX. Les informations sont récupérées, filtrées pour éviter les injections, puis envoyées au serveur à l’aide d’un Call. Le serveur va vérifier si l’utilisateur existe et dans ce cas, si le mot de passe est correct. Il va cette fois retourner une unique valeur, respectivement « true » si l’utilisateur existe mais que son mot de passe est incorrecte, « false » si l’utilisateur n’existe pas et une valeur numérique correspondant à l’identifiant de l’utilisateur si toute les informations sont correctes.

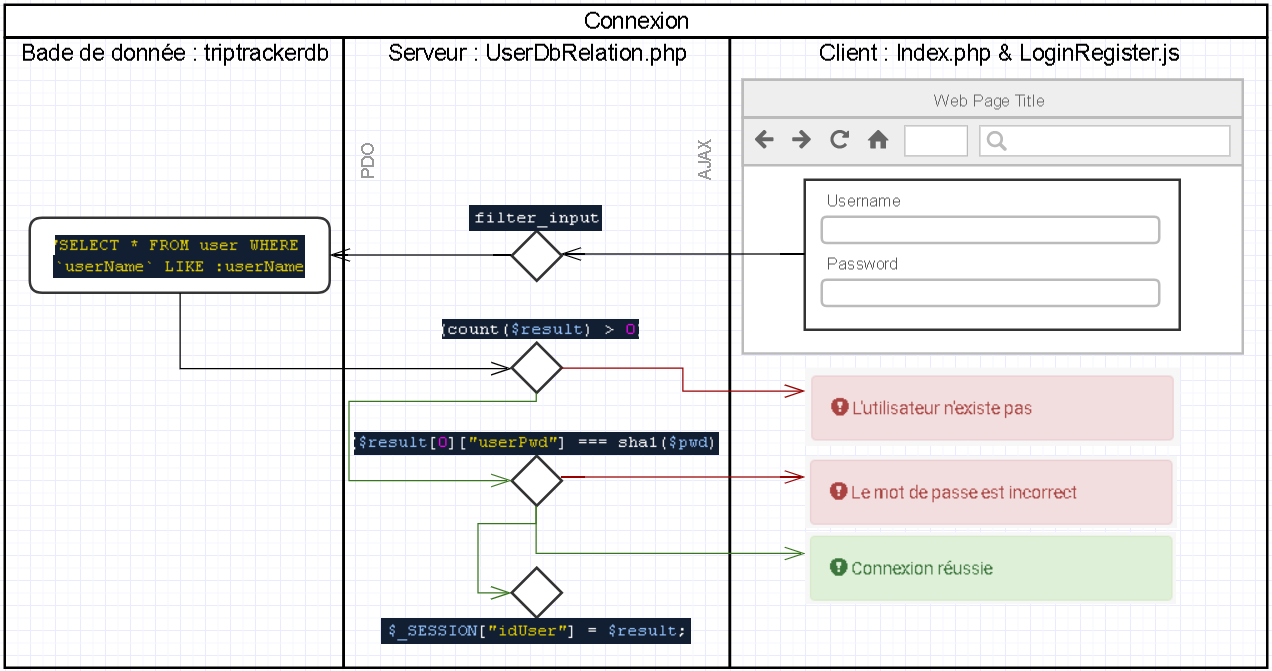


Figure  : Digramme représentant globalement le processus de connexion

Après avoir réceptionné l’unique valeur, c’est cette fois le JavaScript qui va générer les fenêtres d’information, et qui, en cas de réussite lors de la connexion, va actualiser la page. Tout comme l’inscription, si les informations de l’utilisateur sont correctes, l’identifiant de l’utilisateur a été ajouté coté serveur à la session, et sera redirigé dans son espace personnel lors de l’actualisation.

### Interface de création

#### Génération dynamique

Lorsque l’utilisateur se trouve sur l’interface de création, il se trouve en présence de deux bouton, 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 champ 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 également nommé de la même manière.

#### 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 de nom et d’un autre d’indice numérique. Lorsque cette action est déclenchée, on retire la particule correspondant au nom pour récupérer le numéro du panneau. On recompose l’identifiant du panneau, puis on le supprime. Le numéro du panneau sert aussi, à cette étape, à commander la suppression de tous les tracés et les marqueurs en relation avec cette position. Pour plus d’information sur ce dernier point, voir le point dédié au stockage des informations, dans la gestion des routes.

#### 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 nul.

### 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). 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 géocoder la latitude et la longitude récupérée par l’évènement click. On crée un marqueur dont la modalité de stockage est détaillée dans le point concernant le stockage des informations. Puis, on récupère l’adresse de ce point retournée par le géocoder, et on l’insère dans les champs de recherche d’adresse correspondant au focus.

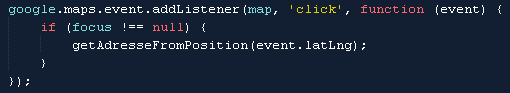


Figure  : Évènement click de la carte

#### Recherche par adresse

Lorsque l’utilisateur click sur la loupe, à droite du champ d’adresse d’une étape, on fait appel à la même fonctionnalité de Google Map API détaillée dans le point précédent : le géocoder. 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 complète reçue, qui est une version corrigée de l’adresse que l’utilisateur a entré.

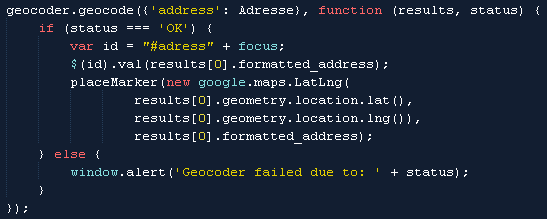


Figure  : Géocoder Google Map prenant en paramètre une adresse

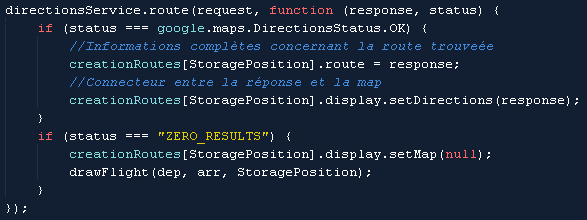


Figure  : Service de route de Google Map

#### 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 de l’information 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 se trouvant à la droite du 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 au géocoder par le même procédé décrit dans le point sur la recherche d’adresse, pour compléter le champ d’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, et pour plein d’autres fonctionnalités 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 vrai, quel que soit le contexte.

Dans cette construction, les règles sont la suivante. Soit deux tableaux, le tableau des marqueurs et le tableau des tracés reliant tous les marqueurs connexe 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 tracés reliant A à B ou A < B est stocké à la position B-1.

Comme le tableau va comporter des « trous », il convient de faire la différence le trou en question et un tracé non défini car la position des deux marqueurs qu’il relie ne sont pas défini. Il en va de même pour les marqueurs, pour différencier ceux qui ne sont pas 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.

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 défini (non nuls) lorsqu’on parcoure le tableau des marqueurs 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 une multitude d’option, comme par exemple le moyen de transport utilisé. Pour maximiser l’accessibilité des emplacements entre eux, je paramètre mes requêtes avec le moyen de transport le plus polyvalent, la marche. Néanmoins, certains emplacement très isolés ou ne se trouvant pas sur le même contient resteront inaccessibles les un par rapport aux autres. Pour ce cas particulier, ils seront reliés par une ligne courbe, comme le vol d’un avion. Logiquement, même si deux destinations ne sont pas accessibles l’une par rapport à l’autre par la terre, elles le seront par les airs.

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 et que celle-ci est que le marqueur d’une étape adjacent à celle-ci est également défini, on trace la route entre ces deux points à l’aide des coordonnées géographiques du marker. Lorsqu’un marqueur est supprimé et que les deux étapes adjacentes possédaient également un marqueur défini, on relie ces deux étapes entre elles par le même procédé.

Les modalités de stockage des tracés sont décrites dans le point concernant le  stockage des informations géographiques.

### 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. Avec toute ces valeurs, ainsi que l’identifiant, on en fait un tableau associatif.

#### Récupération du tracé

Une fois la certitude que toutes les informations du voyage sont correctement entrées, le tracé doit être récupéré. Le tracé étant stocké en un objet unique, il est nécessaire de prendre chaque tracé présent sur la carte et de les combiner les uns à la suite des autres. Cette étape nécessite une bonne connaissance des objets retournée par le service de route de Google Map API. Je me suis donc largement inspiré d’un code dont la source se trouve en annexe.

Le paramètre leg de l’objet contient plusieurs objets, qui contiennent eux-mêmes des listes de points géographiques. En extrayant chaque liste de points de chaque leg de chaque Polyline, 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 du parcours, on obtienne la syntaxe d’un tableau JSON valide.

#### Envoi des informations

Une fois les informations récupérées, validées et le tracé récupéré, 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.

Côté serveur, c’est d’abord le voyage qui sera ajouté à la base de données. Son tracé, qui est en général trop volumineux pour être stocké dans le champs d’une table, sera stocké dans un fichier texte auxquels on attribue un nom aléatoire. Le nom sera récupéré le champs « pat Object » prendra cette valeur. Le tableau de contenu décodé sera ensuite parcouru. Chaque étape du voyage sera ajoutée à la base de données et son id sera récupéré. Lors de son ajout, un cookie portant comme clé le numéro que portait l’étape dans l’interface de création et comme valeur son id une fois inséré dans la base de données. Ce cookie dure dix seconde, et servira lors de l’insertion des images.

L’ensemble des actions effectué côté serveur se fait sous la forme de transaction. Cela signifie que si une erreur survient durant la procédure, toute les modifications apportées au serveur et à la base de données seront annulées. Cette méthode permet, d’éviter les entrée et fichiers solitaire qui polluent la base de données et le serveur.

#### Envoi des images

Le composant que j’utilise pour permettre à l’utilisateur de sélectionner des images est un input avec une surcouche, dont le fonctionnement est détaillé plus précisément dans le point dédié aux usages d’éléments externes. Ce composant permet, entre autres, d’envoyer les images via AJAX à une adresse spécifiée à sa création. L’inconvénient de cette méthode est qu’une fois le composant déclaré, il n’est pas possible d’en changer les paramètres sans le réinitialiser. Mais, bien qu’il soit nécessaire de connaitre l’identifiant d’une étape pour lui lier une image, on ne connait l’identifiant de l’étape qu’une fois après l’avoir inséré dans la base. Si l’insertion les informations dans la base est déclenche, c’est que l’utilisateur a validé le formulaire et donc qu’il a déjà inséré ses images : il n’est donc pas possible de réinitialiser les sélections d’images sous peine de perdre des informations.

C’est pour palier à ce défaut que des cookies de très courte durée sont initialisés lorsque les étapes sont insérées dans la base. Ces cookies permettent au serveur de retrouver les identifiant des étapes lorsqu’après avoir enregistré une image en générant un nom aléatoire, il doit insérer son nom dans la base de données pour la lier à son étape.

#### Sécurité

Pour éviter que des information invalide soit insérées, ou pire, que des injections SQL puissent être effectuées, le site se doit d’être sécurisé 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 dernière existe.

Pour le call AJAX, on utilise un input post, pour éviter que l’utilisateur n’ait 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.

## Navigation

### Pagination

Pour des questions de lisibilité, la navigation est décomposée en pages de cinq voyages chacune, et toute la fonction concernant le chargement des informations est construit autour de cette pagination. Tout d’abord, l’application interroge la base de données sur le nombre de voyage que l’utilisateur a à son actif, puis elle génère des liens vers ses pages. Cette décomposition permet également d'instaurer un code de couleur pour maintenir la clarté et la lisibilité.

### Chargement des voyages

Après avoir généré les liens des pages, ce sont les informations des voyages qui sont chargés. Le chargement s'effectue à l'aide d'AJAX, et en fonction de la page sélectionnée. A l'initialisation c'est la première page qui est chargée par défaut. Les informations retournées par le serveur se présentent sous la forme d'un tableau indexé. Chaque élément est un voyages, lui-même un tableau indexé contenant les informations minimes au chargement des détails.

Quant aux tracés, ils ont également chargés sous la forme d'un constructeur depuis la base de données. Ils se trouvent dans l'un des paramètres du voyage. Ils sont tracés à l'aide de polylines Google Map, construites directement à partir du constructeur en l'état. Une fois construits, la map principale leur est ajoutée en paramètre, et c'est ce qui permet de les visualiser sur la carte. Un évènement leur est aussi attribué au click, permettant d'accéder au contenu du voyage correspondant.

Des marqueurs sont également générés pour indiquer les étapes. Ils sont également stockés dans une variable globale, en deux dimension cette fois : la première dimension gradue les voyage, la seconde les étapes. Comme pour les tracés, on ajoute au marqueur un évènement permettant de les lier à l'étape leur correspondant.

### Chargement des détails

Lorsqu’une étape d'un voyage est sélectionnée, le panneau de navigation se ferme et le panneau de droite s'ouvre. 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 média qui lui correspondent. Pour les informations, les champs composants statiques de l'onglet sont simplement modifiés. Pour les média, 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 est entièrement masquée.

### Cohésion des différents composants

Malgré le code de couleur, il est relativement difficile de s'y retrouver parmi les différents composants. Ainsi, en orientant le focus de la navigation comme de la map sur l'élément sélectionné pourrait permettre un peu plus de clarté. Ainsi, comme pour les autres blocs de panneaux présents sur l'application, un seul panneau peut-être ouvert à la fois. Lorsqu'on commande l'ouverture d'un panneau en cliquant dessus, le panneau s'ouvre et l'éventuel autre panneau ouvert se ferme. De son côté, la carte se centre sur le tracé du voyage à l'aide d'un bound. Un bound est un groupe de points que l'on peut imposer à la map comme ses délimitations. Elle va alors placer ajuster son centre et son zoom en conséquence. Les tracés des autres voyages vont disparaître, et les marqueurs relatifs au voyage vont apparaître. L'ensemble de ces sélections d'objet se fait avec le numéro du voyage dans la page (de 0 à 4).

Figure  : Fonction permettant de focaliser la map sur un tracé

Une fois les marqueurs affichés, l'utilisateur pourra afficher les détails d'une étape, soit en cliquant sur la maquer, soit en cliquant sur le détail contenu dans le panneau du voyage. Si le marker contient l'identifiant de l'étape, le lien ne contient que sa position par rapport aux autres markers du voyage. Il doit donc interroger le marker pour récupérer l'id, avant de faire appel au serveur. Lorsque le panneau s'ouvre, le zoom augmente et la carte se centre sur le marker de l'étape.

Un retour en arrière doit également être mis en place. Lorsqu'on ouvre à nouveau le panneau de navigation (ce qui entraine la fermeture de celui de droite), la map s'ajuste de nouveau au voyage contenant l'étape que nous venons de visualiser. De même, lorsqu'on réduit le panneau d'un voyage, la map s'ajuste de manière à ce que tous les voyage soient visibles, toujours avec un bound de Google map.

## Modification

### Chargement des informations

Une instance de modification est lancée (lors du clic sur le pinceau, à côté du titre du panneau, dans la navigation. Cet évènement déclenche en premier lieu de l’interface sur laquelle les insertions sont effectuées. Parallèlement, les informations du voyage à modifier sont chargées grâce à un call Ajax. Le titre est mis à jour, et pour chaque étape est générée un onglet et un marqueur par le même processus que lors de la création, à l’exception du fait que les champs sont pré-remplis. Les routes sont également tracées, non pas à l’aide du tracé du voyage mais grâce à un géocoder. Pour différencier les étapes chargées des nouvelles étapes, l’id de l’étape est stocké dans l’objet du marqueur correspondant à l’étape. Lorsque le marqueur est mis à jour, l’id est transmis de l’ancien marqueur au nouveau marqueur. L’id du voyage est, lui, stocké dans une variable globale

Les images sont également chargées dans l’input, 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.

### Traitement des informations

Lorsque les informations sont validées, les informations sont envoyées au serveur par un call AJAX. Les informations envoyées sont récupérées et validée avec les mêmes fonctions que celle utilisées lors du processus d’ajout, à ceci près qu’un champ « id » est ajouté au voyage et aux étapes chargées lors de la modification.

Le serveur récupère les informations, puis charge les informations actuelle du voyage grâce à son id. Ces informations servent de référence. Il parcourt ensuite les étapes envoyées par le client, une par une. Si l’étape ne possède pas de paramètre « id », l’étape est ajoutée à la base, par le même procédé que celui utilisé lors de l’ajout d’un voyage. Si l’étape en possède un id, ses informations sont mises à jour dans la base de donnée 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 la valeur nulle ont été supprimées par l’utilisateur. Elles sont donc supprimées, ainsi que leur média, fichier et référence. Le voyage, lui, est simplement mis à jour, et son constructeur est réécrit dans le fichier texte lui correspondant.

### Gestion des images

## Suppression des voyages

Les voyages sont supprimés par un call AJAX, 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.

## Sécurisation de la Map

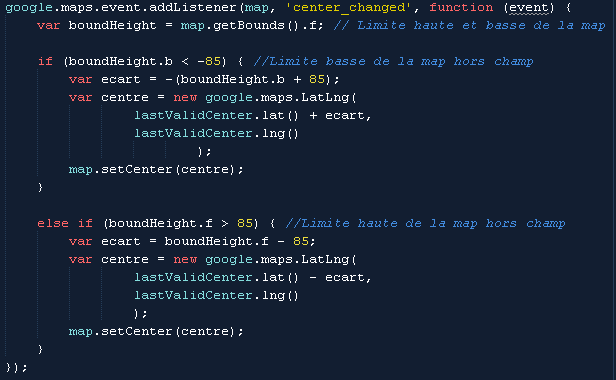
Étant donné que l’une des actions lors de la création ou la modification consiste à cliquer sur la Map pour définir une localisation, ils seraient préférables que l’utilisateur ne puisse pas, malgré les sécurités déjà présentes, accéder aux zones grises en haut et en bas de la carte. La première contrainte 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 le seul niveau de sécurité natif. Bien avec n’importe quel niveau de zoom, on peut toujours déplacer la carte à sortir hors des limites.

Figure  : code de la fonction sécurisant la carte

Plusieurs évènements sont déclenchés lors du déplacement de la carte, mais celui qui permet le plus grand contrôle est l’évènement déclenché par une changement du point central de la carte. Lors de cet évènement, on récupère 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 sa contrition n’est pas respectée, le centre de la carte est décalé en fonction de l’écart constaté, afin que le nouveau placement valide à nouveau les deux conditions.

Il n’est pas nécessaire de placer un autre évènement prenant en compte le cas où le zoom change, car bien que le centre reste à première vue le même lors d’une tel action, la fonction attribuée à l’évènement de changement de zoom est quand même appelée.

# Test de l’application

## Stratégie de test

## Aperçu d’un rapport de test

# Conclusion

## Justesse de la planification

### Planning initiale

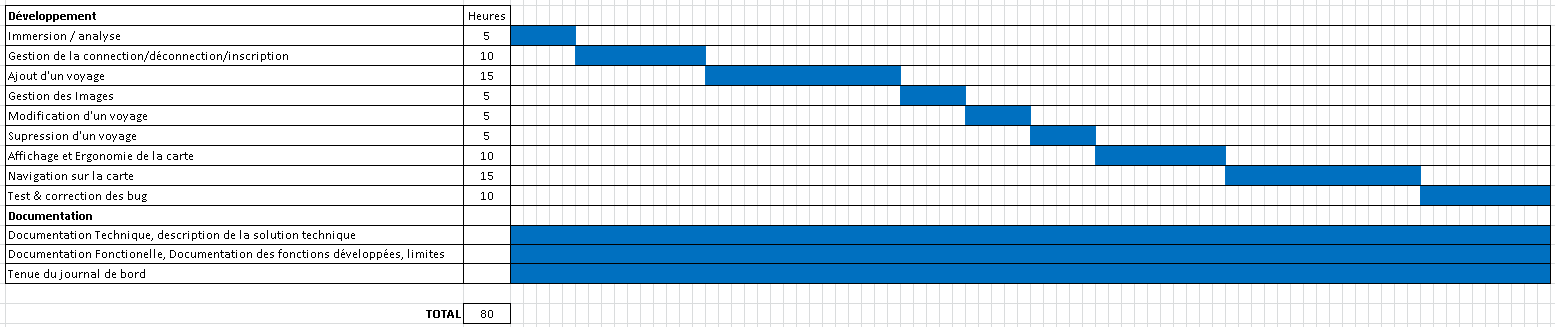


Figure  : Planning prévisionnel

### Planning réel

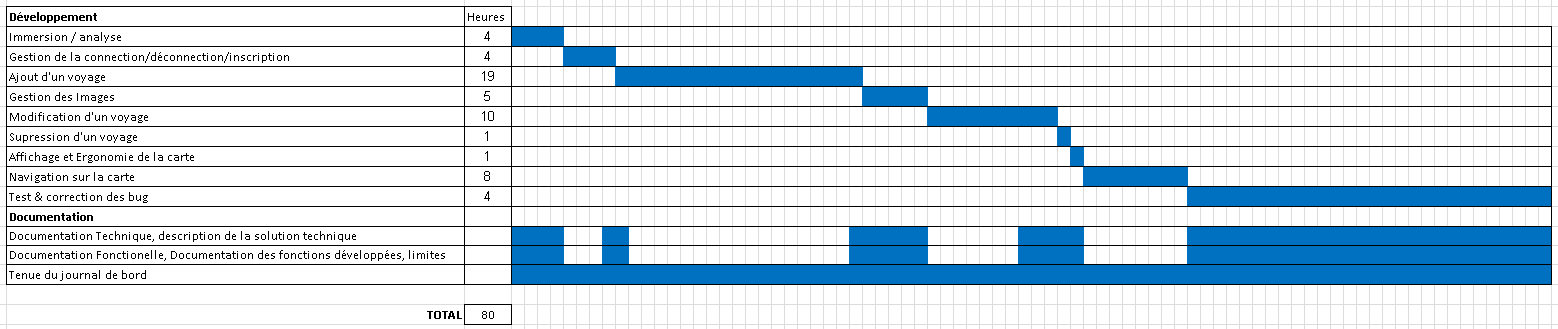


Figure  : Planning réel

### 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.

## Critique du résultat obtenu

## Améliorations possibles

## Intérêt personnel et gain de connaissance

# 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>

# Table des figures

[Figure 1 : Exemple de voyage que l’on peu publier sur Facebook 4](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302066)

[Figure 2 : Extrait d’un journal de voyage du site Travel Diaries 4](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302067)

[Figure 3 : Un voyage de MyTripJournal avec sa description et ses étapes 4](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302068)

[Figure 4 : Navigation dynamique et tracés, une fonctionnalité de Travel Pod 5](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302069)

[Figure 5 : Modèle conceptuel du projet 7](#_Toc485302070)

[Figure 6 : Maquette de la page d’accueil. 8](#_Toc485302071)

[Figure 7 : Maquette de la page d’accueil avec sa modale de connexion 8](#_Toc485302072)

[Figure 8 : Maquette de la page d’accueil avec sa modale d’inscription 9](#_Toc485302073)

[Figure 9 : Maquette de la page principale, lors de la navigation parmi les voyages et étapes 9](#_Toc485302074)

[Figure 10 : Maquette de la page principale, lors de l’aperçu d’une étape 10](#_Toc485302075)

[Figure 11 : Maquette de la page principale, lors de l’ajout ou de la modification d’un voyage 10](#_Toc485302076)

[Figure 12 : Organisation des fichiers du projet 13](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302077)

[Figure 13 : Représentation graphique des dépendances 14](#_Toc485302078)

[Figure 14 : Digramme représentant globalement le processus d’inscription 15](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302079)

[Figure 15 : Digramme représentant globalement le processus de connexion 15](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302080)

[Figure 16 : Évènement click de la carte 16](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302081)

[Figure 17 : Géocoder Google Map prenant en paramètre une adresse 17](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302082)

[Figure 18 : Service de route de Google Map 17](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302083)

[Figure 19 : Fonction permettant de focaliser la map sur un tracé 21](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302084)

[Figure 20 : code de la fonction sécurisant la carte 22](file:///C:\Users\SADEKS_INFO\Desktop\sadekTPI\Documentation\DocTechique.docx#_Toc485302085)

[Figure 21 : Planning prévisionnel 25](#_Toc485302086)

[Figure 22 : Planning réel 25](#_Toc485302087)