BRANCHET Thibaud
DELANOU Quentin
FREMONT Florian

Rapport projet IHM

-

Site de Covoiturage

-

TuturFinder

Se référer au fichier INSTALLATION.txt pour déployer l’application

2016-2017

Introduction

La L3 MIAGE nous a offert l'opportunité d'étudier le Développement IHM et Client-Serveur. De plus, afin de consolider nos connaissances dans ce domaine, nous avons dû réaliser un projet qui englobe les différentes méthodes et aspects de ce développement vus lors des cours magistraux ou travaux dirigés.
Nous sommes donc un trinôme qui se compose de : Thibaud Branchet, Quentin Delanou ainsi que Florian Fremont.

Notre projet était le suivant : réaliser un site de covoiturage comportant certaines fonctionnalités spécifiques citées dans le sujet. Le tout devant être implémenté avec l'aide du framework Symfony. Nous avons donc choisi de réaliser un site de covoiturage, que nous avons nommé *TuturFinder*, en essayant d'implémenter le plus de fonctionnalités possibles. La prise en main du framework Symfony a été compliquée, et certaines fonctionnalités n'ont pas été mises en place à raison d'un manque de temps dû à la complexité des débuts d'apprentissage de Symfony, nous expliquerons les problèmes rencontrés par la suite. Cependant, nous avons implémenté bon nombre de fonctionnalités et ajoutées certaines qui n'étaient pas demandées. Nous vous les détaillerons dans ce rapport par la suite.

Nous aborderons dans un premier temps les fonctionnalités ajoutées, c'est à dire celles qui sont présentes dans notre application de covoiturage. La deuxième partie se focalise sur les fonctionnalités que nous n'avons pas réussi à implémenter et pourquoi.

1. Fonctionnalités ajoutées

a. Template Bootstrap

Pour réaliser notre projet, nous avons choisi de réutiliser le très connu Template Bootstrap déjà performant pour faciliter la réalisation de la vue de notre application, tout en rendant celle ci responsive. De ce fait, nous avons pu directement nous concentrer sur des aspects plus spécifiques à notre site. De plus, certains plugins ont été incorporés à *TuturFinder* pour gagner en efficacité. Nous souhaitions en effet obtenir un site au design épuré, sans laisser de détails superflus.

Nous avons donc opté pour le thème Bootstrap suivant (SB Admin), en ne conservant que la base du Template. Voici donc les différentes pages ainsi créées :

i. Index (Accueil, name="myhomepage")

Cette page est la deuxième page que l'utilisateur rencontre lors de son arrivée sur notre site. En effet, il est d'abord dirigé vers une page de connexion (gérée par RolesBundle:Utilisateur) avant de parvenir à l'index. Si l'utilisateur souhaite visiter le site sans être connecté, il peut poursuivre en *annonyme* et de ce fait son affichage ne sera pas le même que celui d'un utilisateur connecté. Celui ci se voit alors privé de certaines fonctionnalités comme l'ajout d'annonce par exemple.

On retrouve alors quatre différents blocs (respectivements bleu, orange, vert et rouge) qui offrent les services suivants :

- **bleu :** redirection vers la page *Publier une annonce*, explicitée dans la partie qui suit. Le nombre *5 000* correspond au nombre d'utilisateurs inscrits. Comme notre projet n'est qu'un prototype, nous avons laissé cette valeur en "dure" mais le but est de le modifier à chaque création de compte sur notre site. C'est une fonctionnalité qui n'a pas été implémentée mais elle pourrait être pertinente pour une possible suite de ce projet et simple à mettre en place
- **orange :** redirection vers la page *Consulter les annonces*, explicitée dans la partie correspondante. Comme le bloc **bleu**, il indique un nombre correspondant aux nombre d'annonces publiées sur le site. C'est aussi une fonctionnalité qui n'a pas été implémentée mais elle pourrait être pertinente pour une possible suite de ce projet.
- **vert :** lors du clique affiche via la boîte de dialogue JQuery la météo de Nantes en temps réel. Se reporter à la partie sur les Flux RSS pour plus de détails.
- **rouge :** lors du clique affiche via la boîte de dialogue JQuery le trafic du grand Ouest en temps réel. Se reporter à la partie sur les Flux RSS pour plus de détails.

ii. Publier une annonce

Cette page est l'une des pages que nous avons créé en priorité car elle est la clef de voûte de notre projet. En effet tout notre projet se base sur l'accès à des annonces, reserver des places d'une annonce et de ce fait il était primordiale pour faire des tests d'ajouter des tuples d'annonces dans notre base de données. Pour cela rien de plus simple, il faut un formulaire recueillant les attributs d'un tuple de notre table Annonce c'est à dire le titre de l'annonce, les villes d'arrivées et de départs etc. Nous ne lui demandons effectivement pas ses informations propres, celles-ci sont récupérées facilement à l'aide des variables de session car il faut bien évidemment être connecté pour faire cette opération. Si l'utilisateur n'est pas connecté (donc que c'est un visiteur anonyme) on le renvoie sur la page de connexion en lui affichant un message d'alerte lui indiquant qu'il faut qu'il soit connecté pour faire cette annonce. Celui-ci peut alors se créer un compte.

iii. Barre de navigation

Les barres de navigations se composent de deux blocs : vertical gauche et horizontal, ces blocs étant fixe pour toutes les pages, nous avons modifié nos base.html.twig afin d'appeler les blocs de ce fichier pour afficher la navigation des pages. Il nous a fallu dupliquer ces base.html.twig afin qu'ils s'adaptent au type d'utilisateur (utilisateur connecté ou anonyme).

- Celui de gauche redirige simplement l'utilisateur vers les pages du site.
- Celui du haut permet à l'utilisateur de se déconnecter s'il s'est authentifié (sinon un onglet lui propose un chemin afin de se connecter). Il permet aussi de changer de thème et de basculer entre la langue française et anglaise (se référer à la partie correspondante). Aussi, le logo du site et le slogan redirigent juste vers l'accueil (index) du site.

iv. Rechercher les annonces et réserver un covoiturage

La recherche des annonces et la réservation d'un covoiturage se fait en trois étapes bien distinctes et donc par trois fichiers différents. Cette recherche peut se faire qu'il soit connecté ou non à notre site.
La première étape consiste à récupérer les critères de recherche de l'utilisateur à l'aide d'un simple formulaire.

A la base nous avions laissé le choix de recherche sur beaucoup de critères (tel que prix maximum par place, nombre de places disponibles minimum …) mais nous avons vu que cela était peu utile et nous nous sommes restreints à la ville de départ. En effet dans la plupart des sites de covoiturage, l'utilisateur choisit seulement une ville de départ et par heure et peut trier par la suite la liste des annonces par prix, nombre de place etc.

Dans notre cas il verra la liste des annonces respectant ses critères (deuxième étape) et pourra pour chacune d'entre elles voir une description plus détaillée et le profil de l'annonceur (3eme étape). La dernière étape se fait seulement si l'utilisateur s'est connecté au préalable ce qui semble évident pour pouvoir réserver une place. Si ce dernier n'est pas connecté et essaie de réserver un covoiturage, il est renvoyé sur la page de connexion.
Ce scénario est le plus important de notre projet car il permet de mettre en application toutes nos fonctionnalités précédemment implémentées comme l'ajout ou la recherche d'annonces.

C'est au cours de l'implémentation de cette fonctionnalitée que nous nous somme aperçu que nous avions besoin d'une autre entitée : Réservation. Cette entitée "association" fonctionne comme les tables associations des base de donnée relationnelles, c'est à dire qu'elle permet pour une annonce, plusieurs réservations. En effet, nous voulions à la base enregistrer des "array" d'utilisateur pour une annonce (correspondant aux passagers des annonce), mais cela ne respectait pas la règle de l'atomicité des bases de données relationnelles, ainsi, notre ORM Doctrine enregistrait ses array en tant que chaîne de caractère. Bien qu'il aurait été possible de manipuler les transactions avec des chaînes de caractère (à l'aide des méthodes implode() et explode() natives à PHP), nous avons donc choisie la première solution citée, plus "élégante".

v. Consulter les annonces et les réservations d'un utilisateur

Ces fonctionnalités sollicitent les entitées Réservation et Annonce. Dans le cas des annonces postées on regarde simplement l'utilisateur qui poste les annonces et pour les reservation ont regarde si l'utilisateur apparaît dans la table Réservation, pour chaque id_user égal à l'id de l'utilisateur (dans la variable de session) on affiche l'annonce associée. Nous avons choisi pour la même raison que lors de la recherche d'annonce, d'afficher toutes les annonces/réservations sous forme de petits blocs avec des brèves informations pour une meilleure lisibilité globale. C'est alors que nous avons vu les réels atout de Twig, comme faire des boucles dans le HTML (ce qui réduit énormément le code) ou bien faire le calcul du nombre de place disponible directement dans le fichier HTML.

vi. Pages de connexion et d'inscription

La première page sur laquelle nous arrivons lorsque théoriquement nous recherchons notre site est la page de connexion ou d'inscription. Cette page permet trois fonctionnalités basiques mais primordiales :

1. La connexion des utilisateurs grâce à leur adresse e-mail et un mot de passe. Ce compte doit donc exister au préalable. Par la suite l'utilisateur aura la possibilité de se déconnecter depuis la barre de navigation en haut à droite.
2. L'inscription d'un utilisateur afin de lui associer un compte pour qu'il puisse utiliser la première fonctionnalité : la connexion. Après avoir créé son compte et s'être connecté il peut accéder à son profil et ajouter et/ou modifier ses informations personnelles, tout comme publier une annonce.
3. La dernière fonctionnalité est la connexion anonyme. Cette dernière permet un accès à notre site et certaines de ses fonctionnalités tel que la recherche des annonces disponibles selon des critères mais reste limité dans ses actions. En effet lors d'une telle connexion, l'utilisateur n'a pas la possibilité de voir ses annonces, ses réservations, réserver un covoiturage ni accéder aux settings de son profil.

vii. Profil et Settings de l'utilisateur

La dernière page que nous avons créée est celle accessible seulement pour un utilisateur connecté et depuis la barre de navigation supérieure à droite. Cette dernière permet à l'utilisateur de voir son profil mais il a aussi la possibilité de modifier ses informations personnelles comme son nom ou prénom ou encore d'ajouter des informations qui n'étaient pas présentes (car non indispensables) lors de l'inscription.

b. RSS (Météo et Trafic)

En parallèle des créations de nos pages html , nous nous sommes penchés sur le cas des flux RSS qui était important d'intégrer dans notre site.

Nous nous sommes vite rendu compte qu'un flux RSS sur la météo et le trafic était important dans un site de covoiturage. Nous avons donc choisi d'intégrer deux widget RSS afin permettre à l'utilisateur d'accéder depuis l'accueil à la météo de Nantes et au trafic routier de l'ouest de la France. Ces derniers apportent des informations sur des lieux qu'on a choisi de mettre fixent. Une suite possible de notre site serait que l'utilisateur puisse choisir les villes/régions sur lesquelles il souhaite avoir ces informations.

c. Calcul d'itinéraire (Google API)

Bien qu'une fonctionnalité permettant de calculer l'itinéraire ne nous soit pas demandée dans le cadre de notre projet, il nous a paru indispensable d'en intégrer une.

Nous avons donc opté pour l'utilisation de l'API Google Map. Elle fonctionne alors parfaitement dans une page HTML indépendante de notre projet mais l'idée était bien évidemment de l'intégrer à *TuturFinder*. Après énormément de tentatives pour l'intégrer, nous y sommes parvenus en l'ajoutant à la fin de la page *publier une annonce*. L'utilisateur peut alors calculer l'itinéraire (temps du trajet, routes à emprunter, visuel sur la carte) en remplissant les champs *ville de départ* et *ville d'arrivée* et de cliquer sur le bouton *Calculer l'itinéraire*. En faisant abstraction de l'autocomplétion utilisée (cf partie AJAX) pour ces deux champs, il est possible par exemple de spécifier une ville qui n'apparaît pas dans la liste déroulante de l'autocomplétion. Ainsi, on pourrait par exemple calculer un trajet entre New York et Miami.
Il nous a donc fallu modifier l'API Google Map pour qu'elle puisse s'adapter à notre application pour la rendre la plus fonctionnelle possible. Beaucoup d'étapes ont alors été nécessaires pour finaliser cette fonctionnalité. Une suite possible de notre projet pourrait être de réutiliser les données calculées par Google pour déterminer une heure d'arrivée, et estimer un prix de base du trajet en fonction des péages et frais d'essence du voyage.

d. Sélection d'un thème par l'utilisateur

Une des dernières fonctionnalités qui nous étaient demandées était d'ajouter à notre site la possibilité à l'utilisateur de choisir différents thèmes de présentation. Cette sélection de thème se fait dans notre base de navigation en haut à droite. Nous avons choisi de laisser le choix entre deux thèmes, celui que nous avions choisi de base (Flatly) plus un second qui est Lumen. Nous aurions pu en ajouter une infinité mais nous nous sommes limités à seulement deux car cela requiert du travail supplémentaire et redondant. Ce travail repose sur la compréhension des aspects de choix de CSS(via bootswatch), puis sur la gestion de la variable de session qui enregistre le choix du thème, et enfin la gestion de la base.html.twig à utiliser en fonction de cette variable de session. C'est lorsque nous avions bien compris l'utilisation du moteur de template Twig que nous avions réussi à implémenter cette fonctionnalité.

e. Autocomplétion des villes (AJAX)

La dernière fonctionnalité que nous avons intégrée est celle utilisant AJAX. Nous avons choisis de l'utiliser pour l'autocomplétion des villes dans la page *publier une annonce*. Nous avons eu pas mal de difficultés pour l'intégrer au projet Symfony. En effet, ce système utilise à la fois AJAX mais aussi un fichier en JSON dans lequel sont stockées les villes. Le problème était via Symfony de faire correspondre le bon chemin pour le fichier JSON. Finalement, nous avons opté pour intégrer le script Javascript dans la page HTML afin d'utiliser les balises TWIG pour récupérer le chemin du fichier JSON.

Ce fichier JSON a alors été créé par nous même afin de répondre à nos besoins. Nous avons donc une petite liste de villes mais il pourrait s'étendre à un nombre bien plus important de villes. L'idée était surtout d'utiliser AJAX pour permettre cette autocompletion. L'autocomplétion est quant à elle gérée par un plugin d'autocomplétion utilisant AJAX. Il utilise de ce fait JSON et correspondait parfaitement à nos besoins pour le projet. Une des suites possibles consisterait à utiliser l'autocomplétion à plusieurs endroits dans notre site pour aiguiller au maximum l'utilisateur, lui permettant un gain de temps non négligeable.

2. Fonctionnalités à ajouter

a. Sélection des langues par l'utilisateur

La première fonctionnalité qu'on aurait voulu implémenter est celle de la sélection des langues par l'utilisateur dans un but d'internationalisation de notre site. Cette fonctionnalité n'a pas pus être ajouter dans le cadre de notre site web car cette dernière était très fastidieuse à réaliser de part la traduction mot à mot de tout notre site. Il nous manquait donc du temps et nous nous sommes plus concentrés sur la stabilité des autres fonctionnalités qui étaient plus importantes selon nous. Cependant nous nous sommes tout de même pencher sur celle-ci et nous pensons donc avoir une idée de marche à suivre pour l'ajouter à notre projet.

b. Utilisation de FosUserBundle

Le Framework Symfony repose sur la création, la gestion et la réutilisation des Bundles. C'est pourquoi notre toute première idée du projet fût de rechercher les bundles que nous aurions pu utiliser. Nous avons donc commencé par intégrer le Bundle FosUserBundle.

C'est après plusieurs essais (et donc après avoir recommencé le projet 0) que nous avons réussi à installer FosUserBundle dans notre projet Symfony. Cependant lorsque nous avons voulu l'utiliser, nous avions un problème du type "unserialize type exception". Ce problème nous a pris énormément de temps, et nous n'avions pas réussi à "lancer" le projet pendant un longue période (les mois de janvier à février). Nous avons donc pris la décision

d'intégrer notre propre bundle RolesUtilisateur, et de gérer nous même la gestion de nos rôles et de nos comptes utilisateurs.

c. Gestion des rôles

Cette non-utilisation de FosUserBundle a engendrée plusieurs problèmes. Le plus important étant la gestion des rôles, et plus spécifiquement la gestion des variables de sessions. Nous avons réussi à gérer tous les scénarios des deux rôles (utilisateur connecté ou anonyme), sauf un : lorsque l'utilisateur anonyme veut consulter les annonces, la variable de session de l'utilisateur précédent est récupérée et la page de recherche s'affiche en tant qu'utilisateur connecté. Le problème se situe dans le chemin nommé "myhomepage", cependant nous n'avons pas réussi à résoudre ce problème à temps.

Conclusion

Au cours de ce projet, nous avons affiné nos connaissances dans le domaine du web, notamment avec l'utilisation du PHP, javascript et Ajax. Mais aussi et principalement dans l'utilisation du Framework Symfony et du moteur de Template Twig.

Cependant nous avons fait plusieurs erreurs. La première fut de vouloir directement faire quelque chose de propre et "finit" dès le commencement du projet. Cette sous-estimation du temps d'apprentissage nous a fait recommencer plusieurs fois le projet, car nous avons éprouvé beaucoup de difficultés à paramétrer Symfony. Cela a eu pour effet de ralentir l'avancement du projet. Lorsque nous nous sommes lancés dans ce projet, nous avions énormément d'idées quant aux fonctionnalités à ajouter, ce qui nous a fait "partir" dans plusieurs directions, alors que nous aurions dû nous concentrer sur quelque chose de simple, dès le début du projet. Nous avons par exemple créé plusieurs entitées inutiles, que nous avions dû supprimer par la suite (comme une entitée document qui gérait les photos de Profil utilisateur).

Ce retard nous a cependant poussé à se répartir efficacement les tâches : Florian Fremont s'est occupé des pages HTML responsives, Thibaud Branchet s'est occupé des fonctionnalités javascript comme l'intégration d'ajax et des flux RSS au sein du projet de certaines pages HTML et Quentin Delanou s'est quand à lui occupé du paramétrage Symfony, routing et gestion des données. Cette répartition nous permettait de travailler indépendamment dans un premier temps, ce qui nous a permis d'implémenter de nombreuses fonctionnalités rapidement. L'intérêt était alors d'utiliser ce que l'un avait fait pour l'intégrer efficacement à Symfony. Cette répartition était obligatoire pour espérer finir le projet dans les temps. Cependant, nous sommes un peu frustré de n'avoir travaillé chacun que sur une partie du projet. Nous avons tout de même lors de la dernière semaine travaillé ensemble pour mieux comprendre les aspects de Symfony et ainsi mieux cerner notre projet.

Bien que le projet nous a pris de nombreuses heures de travail (plus d'une trentaines par personnes) nous avons trouvé ce projet riche et intéressant une fois la base du projet lancé (post paramétrage Symfony). Au final ce projet fut une très bonne expérience. Nous avons approfondi nos connaissance à l'utilisation d'outils importants dans le monde de l'entreprise spécialisée dans les services WEB.