Rapport Projet WEB

ENTRAiiDe

Mai 2021

Clémence CLAVEL
Constant GAYET
Barnabé GEFFROY
Alexia HARIVEL

SOMMAIRE

Introduction

Dans ce rapport, nous essaierons de vous expliquer avec le plus de précision possible, en restant concis, les choix faits et les difficultés rencontrées lors de l'élaboration de notre site Internet.

Ayant choisi le projet libre, nous pensons nécessaire de réexpliquer le principe de notre site Internet "ENTRAiiDe". Tout est venu du constat qu'une grosse communauté des élèves de l'école vit en colocation sur Évry mais ne possède pas forcément tout le matériel électroménager, jeux de sociétés et autres biens qu'ils souhaitent utiliser occasionnellement. L'idée est de créer une plateforme qui mette en relation ces étudiants afin qu'ils puissent trouver d'autres IIEns pour échanger divers objets selon leur nécessité. Cela leur permettrait ainsi un gain de temps et d'argent.

Le principe est le suivant: un utilisateur va se connecter, il pourra alors choisir des objets 'à ajouter' s'il souhaite les prêter à sa liste de biens ou il pourra simplement rechercher un objet à emprunter en fonction de ses besoins/envies.

Lien du site : http://pgsql.pedago.ensiie.fr/~barnabe.geffroy/projet-web/public/

Lien du git : https://github.com/barnabegeffroy/projet-web/

1. Création de la base de données

1.1 Table Utilisateur

Cette table comporte les informations d'un Utilisateur lorsqu'il s'inscrit sur le site :

- Un `id` (identifiant de l'utilisateur) de type `SERIAL`, qui est la clé primaire de cette table. L'avantage du type `SERIAL` est qu'à chaque inscription d'un utilisateur, un id va lui être donné automatiquement.
- Un `prenom` de type `VARCHAR`, qui est le prénom de l'utilisateur et qui va donc être affiché sur son profil.
- Un `pseudo` de type `VARCHAR`, qui est entré par l'utilisateur s'il le désire afin qu'il soit affiché sur son profil.
- Un `nom` de type `VARCHAR`, qui est le nom de l'utilisateur et qui va donc être affiché sur son profil.
- Un `email` de type `VARCHAR`, qui est l'adresse mail de l'utilisateur. Elle est unique pour chaque utilisateur et va donc lui permettre de s'identifier sur le site.
- Un `telephone` de type `VARCHAR`, l'utilisateur entre son numéro de téléphone s'il le désire afin qu'il soit affiché sur son profil.
- Un `password` de type `VARCHAR`, l'utilisateur entre un mot de passe qui va lui permettre de s'identifier sur le site.

1.2 Table Annonce

Cette table comporte les informations propres à chaque annonce postées sur le site :

- Un `id` (identifiant de l'annonce) de type `SERIAL`, qui est la clé primaire de cette table. L'avantage du type `SERIAL` est qu'à chaque création d'une annonce, un id va lui être donné automatiquement.
- Un `titre` de type `VARCHAR` qui est le titre de l'annonce et qui sera affiché en haut de celle-ci.
- Un `idUtilisateur` de type `INTEGER` qui est l'identifiant de l'utilisateur qui a posté cette annonce.
- Un `datePublication` de type `DATE` qui est la date de poste de l'annonce
- Une `description` de type `VARCHAR` qui est la description entrée par l'utilisateur s'il le souhaite de l'objet qu'il prête.
- Une `photo` de type `BOOLEAN` qui vaut `True` si l'utilisateur a mis une photo dans son annonce et `False` sinon. Une seule photo peut être partagée par annonce.
- Un `lieu` de type `VARCHAR` qui est l'adresse correspondant au lieu où se trouve l'objet prêté
- Une `lat` de type `FLOAT` qui est la latitude correspondant à l'adresse de l'annonce
- Une `lng` de type `FLOAT` qui est la longitude correspondant à l'adresse de l'annonce

1.3 Table Favoris

Cette table décrit les annonces mises en favoris par un utilisateur, afin qu'il les retrouve plus rapidement, elle comporte :

- Un `idUtilisateur` de type `INTEGER`, qui est l'identifiant de l'utilisateur concerné.
- Un `idAnnonce` de type `INTEGER`, qui est l'identifiant de l'annonce favorite.
- La clé primaire est le couple (`idUtilisateur`, `idAnnonce`)

1.4 Conversation

Cette table qui gère les conversations entre deux utilisateurs par rapport à une annonce donnée comporte :

- Un `id` (identifiant de la conversation) de type `SERIAL`, qui est la clé primaire de cette table. L'avantage du type `SERIAL` est qu'à chaque création d'une conversation entre deux utilisateurs, un id va lui être donné automatiquement.
- Un `id1` de type `INTEGER` qui est l'identifiant du premier utilisateur (de celui qui à créé l'annonce).
- Un `id2` de type `INTEGER` qui est l'identifiant du second utilisateur, celui qui est intéressé par l'annonce
- Un `conv_idAnnonce` de type `INTEGER` qui est l'identifiant de l'annonce pour laquelle les deux utilisateurs conversent

1.5 Table Message

Cette table décrit les messages échangés par les utilisateurs, elle comporte :

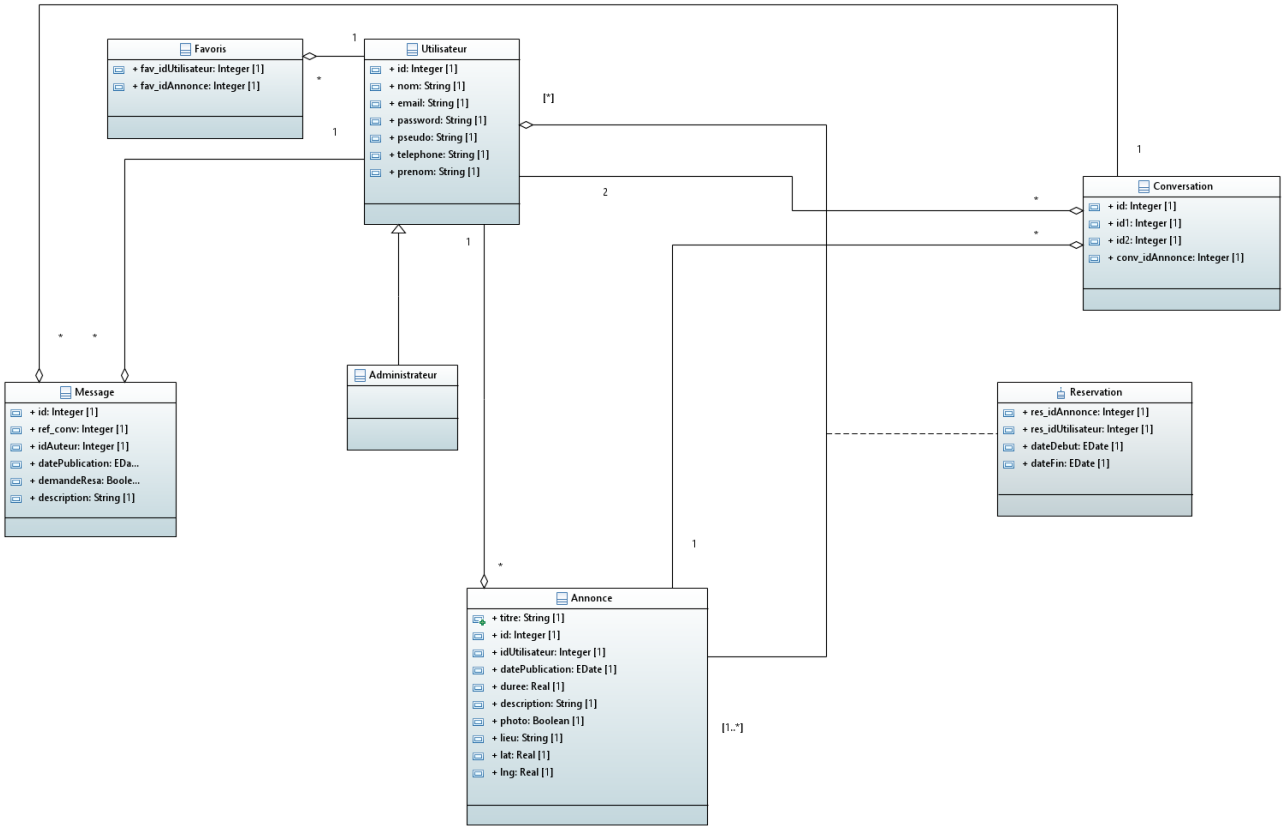
- Un `id` de type `SERIAL`, qui est l'identifiant du message. L'avantage du type `SERIAL` est qu'à chaque envoi d'un message, un id va lui être donné automatiquement. C'est la clé primaire de cette table.
- Un `ref_conv` de type `INTEGER`, qui est l'identifiant de la conversation où se situe le message.
- Un `id_Auteur` de type `INTEGER`, qui est l'identifiant de l'utilisateur qui envoie le message
- Une `datePublication` de type `DATE`, qui est la date à laquelle le message est émis.
- Une `demandeResa` de type `BOOLEAN`, qui indique par `True` si le message est une demande de réservation et par `False` sinon.
- Une `description` de type `VARCHAR`, qui est le contenu du message.

1.6 Table Reservation

Cette table décrit les réservations effectuées par les utilisateurs, elle comporte :

- Un `res_idAnnonce` de type `INTEGER`, qui est l'identifiant de l'annonce réservée.
- Un `res_idUtilisateur` de type `INTEGER`, qui est l'identifiant de l'utilisateur qui réserve.
- Une `dateDebut` de type `DATE`, qui est la date du début de la réservation.
- Un `dateFin` de type `DATE`, qui est la date de fin de la réservation.

1.7 Schéma UML de notre base de données



2. Structure du projet

Nous avons utilisé `git-ftp` pour pouvoir mettre à jour facilement le site sur Pedago et voir en temps réel l'avancée de notre site.

2.1 Structure du code

Étant tous débutants en Web, nous avons mis un certain temps à comprendre comment réaliser une architecture correcte du projet et comment bien s'organiser pour ne pas se perdre dans tous les fichiers.

La structure de notre code a été la première étape à laquelle nous nous sommes attelés de manière à pouvoir travailler efficacement par la suite. Nous avons pour cela suivi le modèle donné dans le premier TP de PHP.

Ainsi, le dossier `data` contient les fichiers liés à notre base de données.

Le dossier `public` contient les fichiers PHP accessibles par l'utilisateur (page web et gestion des formulaires).

Le dossier `src` contient plusieurs sous-dossiers. Le sous-dossier `Assets` contient les scripts JavaScript et les fichiers CSS. Le sous-dossier `Model` contient :

- un répertoire `Entity` contenant les classes métiers,

- un répertoire `Factory` qui permet de gérer le lien avec la base de données (l'utilisateur tpcurseurs a accès à notre base de données pour ce projet),
- un répertoire `Repository` pour les requêtes à la base de données afin de créer des objets,
- un répertoire `service` qui fournit les différents services
- et un répertoire `Hydrator` qui nous permet de créer des entités à partir des données récupérées auprès de la base de données.

Le sous-dossier `View` contient les images à afficher ainsi que le contenu HTML à générer sur les pages. `View` contient également plusieurs sous-dossiers dans lesquels les fichiers PHP sont triés par thématiques (fonctionnalités utilisateur, espace administrateur…).

2.2 Connexion/Inscription

Clémence Clavel et Alexia Harivel se sont chargées de cette partie.

En reprenant le modèle donné en TP et en l'adaptant, nous avons créé des utilisateurs avec toutes les informations nécessaires pour notre base de données. Des fonctions PHP permettent également de tester la connexion sur n'importe quelle page grâce à la variable `$SESSION['user_id']` et ainsi de s'assurer que l'utilisateur s'est identifié lors d'un accès aux pages nécessitant une connexion.
La connexion se fait via la page `./public/login.php` et l'inscription `./public/signup.php`, accessibles dans la barre de navigation si l'utilisateur n'est pas connecté.

2.3 Profil utilisateur/Modifier son profil

Cette partie a été réalisée par l'ensemble des membres du groupe.

Lorsque l'utilisateur est connecté, il a accès à la page `./public/account.php` accessible via la barre de navigation. Il peut y voir ses informations, les modifier et supprimer son compte.
Modifier ses informations personnelles fait appel à deux pages distinctes : une pour modifier son mot de passe, l'autre pour modifier les autres informations.
Supprimer son compte se fait avec un formulaire de confirmation qui apparaît quand on clique sur le bouton "Supprimer son compte". Ce formulaire a été réalisé grâce à ce tutoriel.

2.4 Les annonces

Cette partie a été réalisée par Barnabé Geffroy

2.4.1 Ajout/modification

Pour la création d'une annonce il suffit de remplir un formulaire sur `./public/newAnnounce.php`. Le titre et l'adresse sont obligatoires. On peut également ajouter une image. Le fichier est vérifié pour s'assurer qu'il n'est pas trop volumineux et que c'est bien une image. Nous n'avons pas réussi à faire l'upload des images sans faire un `chmod 777` sur le dossier d'arrivée.
La modification d'une annonce se fait en utilisant également `./public/newAnnounce.php`. Nous avons fait ce choix pour éviter d'écrire deux formulaires quasi identiques. La suppression de l'image est ajoutée lors de la modification d'une image.

2.4.2 Affichage

L'affichage d'une liste d'annonces est présente sur de nombreuses pages : la page d'accueil, "Mes favoris", "Mes annonces", "Mes réservations", "Mes entraiides". Nous avons donc décidé d'écrire une fonction PHP `loadAnnounce` (présente dans `template.php`) qui permet d'afficher une annonce. Ainsi chaque page utilise une boucle `foreach` et affiche les annonces nécessaires sans avoir à ré-implémenter la mise en forme de chaque annonce. Par exemple dans "Mes Favoris" (`./public/myFavs.php`), la ligne

```
$favs = $announceRepository->findAllFavs($authenticatorService->getCurrentUserId());
```

récupère la liste des favoris de l'utilisateur et affiche une à une les annonces grâce à la fonction `loadAnnounce` et une boucle `foreach`.

En ce qui concerne l'affichage d'une seule annonce, il est réalisé sur `./projet-web/public/announce.php?id=n` où `n` est l'identifiant de l'annonce. Cet affichage est différent en fonction des droits de l'utilisateur.

2.5 Administrateurs

Cette partie a été réalisée par Constant Gayet.

Nous avons fait le choix d'accorder des fonctionnalités aux quatre premiers utilisateurs (nous quatre). Ces administrateurs peuvent modérer le contenu du site sans passer par la base de données. Ils peuvent donc supprimer les annonces et les utilisateurs.

La suppression des annonces se fait sur la page de l'annonce en question. Si l'utilisateur est administrateur, un bouton "supprimer l'annonce" est disponible. Bien entendu, si c'est sa propre annonce ce bouton n'apparaît pas car il y a déjà le bouton "Supprimer mon annonce".

2.6 Messagerie

Cette partie a été réalisée par Clémence Clavel.

Nous n'avons pas trouvé de modèle de chat facile à implémenter dans notre projet. Nous nous sommes donc concertés pour choisir la manière dont nous allions réaliser cette messagerie. Nous avons décidé d'utiliser deux tables : une gérant les conversations et une autre gérant les messages. Cette dissociation permet d'afficher toutes les conversations d'un utilisateur dans `./public/chat.php` sans faire une requête sur tous les messages, qui demanderait un temps d'exécution plus long.

Depuis la messagerie l'utilisateur peut alors ouvrir une conversation (`./public/salon.php?idConv=n` où n est l'identifiant de la conversation). On ne fait donc des requêtes que lorsque l'utilisateur a besoin de ces données. Les conversations ne sont accessibles que par les utilisateurs enregistrés dans la conversation.

La création d'une nouvelle conversation se fait l'envoi d'un message d'un utilisateur à l'auteur d'une annonce (un entraiideur) depuis cette annonce (`./projet-web /public/announce.php?id=n`). Si une conversation existe déjà, le message est ajouté à cette conversation.

Ce modèle de messagerie n'est pas idéal (pas de notification, pas de mise à jour instantanée) mais fournit les services nécessaires au fonctionnement de notre site web.

2.7 Réservations

Cette partie a été réalisée par Barnabé Geffroy

Plusieurs contraintes s'imposent lors de l'ajout d'une réservation. Il ne peut y avoir de réservations le même jour. Une réservation ne peut pas dépasser le nombre de jours maximum indiqué par l'entraiideur. Il faut également que l'utilisateur puisse voir les jours de disponibilité de l'article.

Nous avons pensé à réaliser un calendrier comme sur le site de AirBnb qui permet de sélectionner une période et de voir en même temps les disponibilités de l'article. DateTimePicker de bootstrap permet cette fonctionnalité. En s'aidant de plusieurs modèles, nous avons réussi à combiner un calendrier qui affiche les dates réservées en rouge et un calendrier permettant de sélectionner une période de prêt. Seuls les utilisateurs connectés et qui ne sont pas l'entraiideur de l'annonce peuvent réserver.

La réservation va envoyer un message (créer une conversation si besoin) à l'entraiideur. Dans la base de données, la colonne "demande" du message est initialisée à TRUE. Cela permet de réaliser un affichage particulier pour l'entraiideur. Celui-ci voit en effet le contenu du message (demande de réservations avec les dates) et deux boutons : Accepter et Refuser. Si l'entraiideur appuie sur Accepter, une réservation est créée. Cette réservation apparaît dans "Mes entraiides" pour l'entraiideur et dans "Mes réservations" pour l'autre utilisateur. Si l'entraiideur appuie sur Refuser, la demande est refusée. Dans les deux cas, le message est mis à jour. `Demande` passe à FALSE et le contenu indique si la demande a été acceptée ou refusée.

Un autre problème s'est également présenté. Si plusieurs utilisateurs font une demande sur les mêmes dates, il faut qu'une seule soit acceptée. Nous avons donc ré-utilisé la logique de vérification des dates disponibles. Ainsi au chargement d'une conversation par l'entraiideur, la demande est automatiquement rejetée si elle ne vérifie plus les critères de disponibilités. Si l'entraiideur met à jour la durée maximale de prêt de son article, toutes les demandes seront refusées de la même manière.

2.8 Affichage d'une carte et auto-complétion de l'adresse

Cette partie a été réalisée par Constant Gayet.

Dans un premier temps, nous avons essayé d'utiliser une API Open Source : Algolia Places API. Cependant, n'étant plus disponible à l'utilisation, nous nous sommes tournés vers les services de Google : Maps API et Places API.

Tout d'abord, Places API nous permet de mettre en place l'auto-complétion de l'adresse lorsque l'utilisateur crée une nouvelle annonce. Cette même API nous permet également de récupérer la latitude et la longitude correspondant à l'adresse indiquée, grâce à la méthode getPlace(), que l'on stocke dans la base de données (erreur à ce niveau, voir partie 3). Ensuite, Maps API nous permet d'afficher une carte centrée sur l'ENSIIE sur laquelle figure un marqueur à l'endroit indiqué. L'utilisateur peut donc facilement situer le bien qu'il veut récupérer.

2.9 La barre de recherche

La barre de recherche a été réalisée par Constant Gayet.

La barre de recherche utilise la méthode `search()` (appartenant à la classe `AnnounceRepository`) qui nous permet de retourner une série de résultats grâce à une requête SQL sur l'ensemble des chaînes de caractères des annonces. Cela signifie que l'on peut obtenir des résultats portant sur la description des annonces par exemple, ce qui améliore la fiabilité de la recherche.

2.10 Design

Cette partie a été réalisée par Alexia Harivel.

Le design du site a été modelé à l'aide de Bootstrap qui est une collection d'outils utiles à la création du design de sites et d'applications web. Bootstrap contient notamment des codes HTML et CSS, des formulaires, mais aussi des boutons et d'autres outils de navigation. Concernant l'agencement du site, tout utilisateur peut, sur chaque page du site, voir le logo et avoir accès à la barre de recherche et un bouton pour lancer la recherche. Et chaque utilisateur inscrit et connecté, a également accès sur toutes les pages au bouton

‘Messagerie’, au bouton ‘Ajouter une annonce’ et au bouton ‘Espace Utilisateur’. Ce dernier contient six sous-boutons : ‘Mon Compte’, ‘Mes favoris’, ‘Mes annonces’, ‘Mes entraiides’, ‘Mes réservations’ et ‘Se déconnecter’.

Concernant les couleurs, nous avons choisi d’utiliser principalement des couleurs neutres pour le site, c’est-à-dire du blanc, du noir et du gris afin qu’il soit agréable à consulter. Nous avons également créé un logo pour le site.

3. Conclusion

Les principales limites de notre site internet sont les suivantes :

- En termes de sécurité, l’upload des images nous a obligé de faire un `chmod 777` sur le dossier d’arrivée des images. Cela présente une faille de sécurité.
- L’optimisation pourrait être revue : le chat ne s’actualise pas tout seul, les recherches pourraient être plus précise (par lieu et/ou catégorie par exemple)
- Lors de l'exécution de maps.js, nous n’arrivons pas à enregistrer la latitude et la longitude dans la base de données. En réalité, nous devons appeler une fonction initMap() lors du chargement de l’API google Maps, à laquelle nous ne pouvons pas entrer d’arguments, ce qui nous pose problème. Par ailleurs, nous aurions sûrement pu résoudre ce problème avec une meilleure gestion de notre temps.
- La barre de recherche est fonctionnelle mais pourrait être améliorée : elle permet de rechercher un mot ou une suite de mots, peu importe les majuscules. Cependant, la recherche d’une suite de mots dans le désordre, par exemple, ne fonctionne pas.
- Par ailleurs, nous aurions voulu mettre en place un système de catégories permettant de rechercher et cataloguer les annonces par catégories.
- Nous aurions pu également mettre en place une notification par mail lorsque l’utilisateur reçoit une nouvelle demande de prêt ou un nouveau message.

Pour conclure, nous avons axé le développement du site vers quelque chose de fonctionnel, proposant un service réaliste au vu de nos compétences en développement web. Bien sûr, il offre des perspectives d’améliorations sur des points précis. Cependant, nous avons revu notre ambition de base à la baisse puisque nous devions réaliser ce projet en temps limité. Finalement, malgré l’oubli des options telles que la notation des utilisateurs et annonces, notre site fournit tous les services nécessaires à son utilisation.