



Rapport de Projet Web

#ERP

Plate-forme Web de Gestion de Junior-Entreprise

Romain A. ALFRED
Amadou BA
Paul MAZET
Antoine NADJAR

Avril 2020 à Mai 2020

1- Le Contexte

A la Junior-Entreprise Dièse, l'ensemble de l'activité de la Junior-Entreprise se retrouve dispatchée entre un drive et une multitude de classeurs. Une plate-forme plus sécurisée, centralisant les données et répondant aux besoins spécifiques d'une Junior-Entreprise telle que Dièse serait donc d'une grande utilité pour les étudiants mais surtout le client pour qui la plate-forme web reste toujours une référence.

Proposer un progiciel de gestion (ERP) sous la forme d'une plate-forme Web qui permettra d'organiser l'activité d'une Junior-Entreprise du domaine des technologies de l'information a donc été notre objectif.

Dans ce projet il va falloir gérer, les conseillers techniques (élèves intervenants sur des projets), les Junior-Entrepreneurs, les clients, mais aussi les nombreux événements se déroulant au sein de la Junior-Entreprise tel que les réunions, les salons, les after-works, les présentations et les soirées ; et les projets.

2- #Development

Pour ce projet, l'équipe **#Development** était composée de quatre membres. Au sein de ce groupe les rôles et tâches ont été réparties en fonction des compétences et appétences de chacun.

Ainsi on retrouve l'organisation suivante :



Chef de Projet

Tâches : rédaction du cahier des charges, modélisation du projet, conception de la base de données, assistance en front-end et PHP



Développeur Fullstack

Tâches : conception et coordination du front-end et du back-end



Développeur Front-end

Tâches : assistance à la modélisation de la base de données, conception du front-end



Développeur Back-end

Tâches : conception des fonctions PHP

Le **Chef de Projet** du groupe a été choisi en fonction de sa capacité à comprendre au mieux les besoins du métier, appartenant lui-même à la Junior-Entreprise Dièse. C'était donc un interlocuteur privilégié avec la Junior-Entreprise elle-même. L'objectif étant d'arriver à la conception d'un outil réellement opérationnel et utilisable.

Du fait que c'est ce dernier qui a mis en place le cahier des charges, le rôle de modélisation de l'outil lui est donc revenu, que ce soit en terme de structure du site mais aussi de la base de données.

Bien entendu, ce sont des modélisations qui ont pu évoluer au fur et à mesure du projet en fonction des nouvelles contraintes et réflexions de toute l'équipe.

Le rôle principal du **Développeur Fullstack** au cours de ce projet a été d'être le coordinateur en chef des différents fichiers et des différentes fonctions PHP du back-end avec le front-end. Savoir quoi mettre en place et où le mettre en place.

Le rôle du **Développeur Front-end** a été de développer le front-end du site. C'est-à-dire l'aspect visuel, le code HTML et CSS.

Enfin le rôle du **Développeur Back-end** a été de développer le back-end. Dans le code, le back-end n'est pas strictement séparé du front-end, dans le sens où l'on peut retrouver des fonctions PHP et donc des requêtes SQL au sein même du fichier de code d'une page. Néanmoins dans la méthodologie de travail, cela a bien été deux aspects complètement différents. Le fait de ne pas séparer strictement les deux c'est fait pour faciliter la clarté et aussi du fait qu'il y avait peu de redondance au niveau des fonctions.

Bien entendu, ce projet est avant tout un moyen d'apprendre. Ainsi, personne ne s'est cantonné uniquement au rôle que l'on s'était définie. Ainsi comme indiqué ci-dessus dans "l'organigramme" de l'équipe **#Development**, les tâches ont été réalisées en collaboration.

3- Modélisation de la base de données

A partir de ces informations, nous avons donc modélisé une **base de données**. Cette dernière nous permettra en premier lieu de créer les différents types d'utilisateurs et d'événements, et les projets.

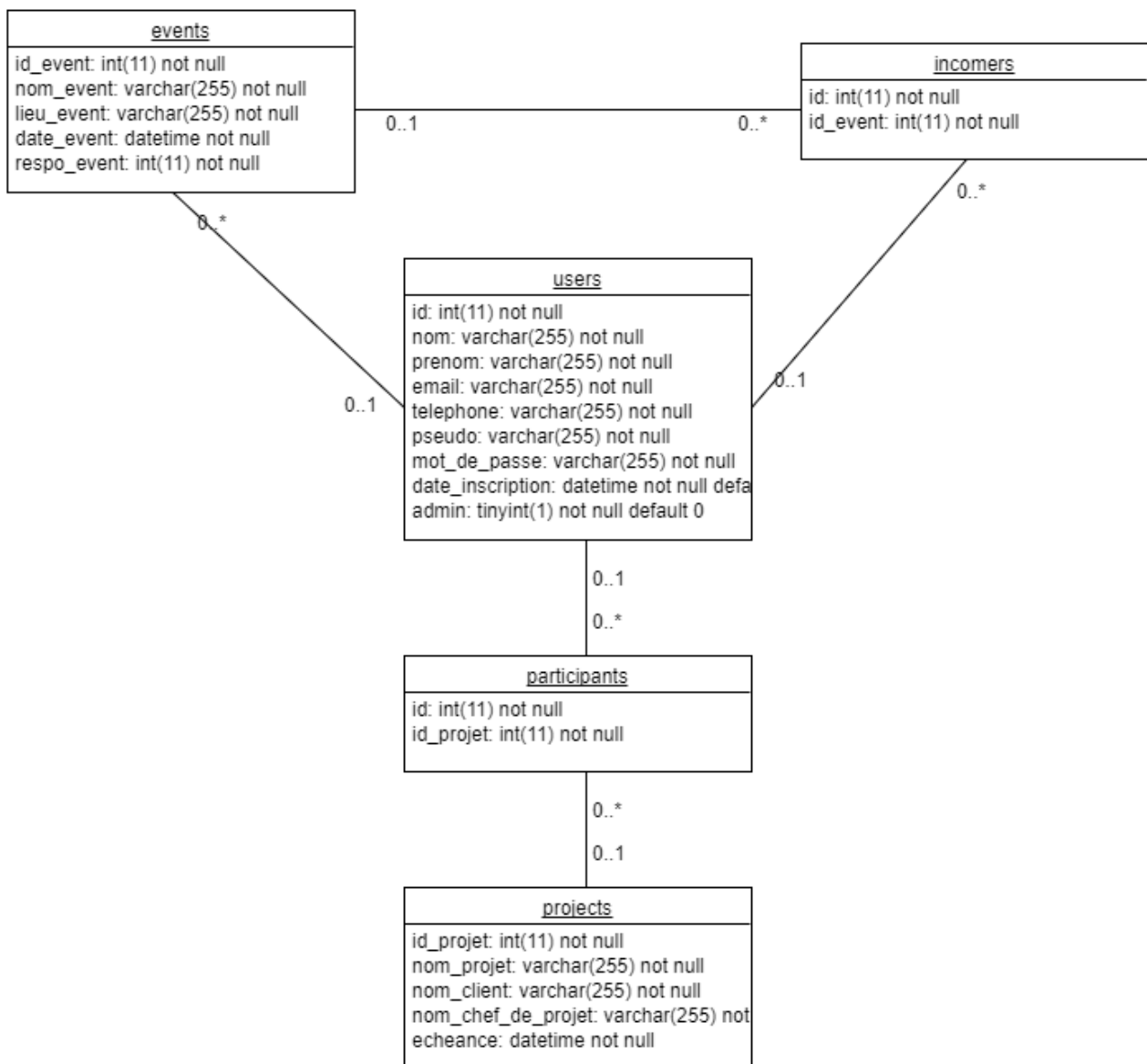
La base de données nous servira de plus à stocker les données, ce qui est très intéressant pour une entreprise en général.

Effectivement cela permet de garder une trace des actions effectuées, mais aussi potentiellement de mettre en place des statistiques en exportant les données. Tant de données qui peuvent aussi s'avérer utiles lors d'un audit.

On obtient ainsi la modélisation suivante : au début, une modélisation qui séparait les différents types d'utilisateurs avait été envisagée : chefs de projet, administrateurs (administrateurs de la Junior-Entreprise), clients et conseillers techniques. Cela aurait nécessité un héritage : Administrateur héritant de chef de projet, et chef de projet et conseillers techniques héritant de Dièseman (membre de la Junior-Entreprise Dièse), et enfin Dièseman et client héritant d'utilisateur.

Néanmoins cela nécessitait de « remonter » une suite d'héritage à chaque fois que l'on voulait récupérer des informations sur un utilisateur, dans les fonctions PHP. L'organisation ci-dessus a donc été préférée pour sa simplicité.

Illustration du Modèle Entités-Associations



La différenciation entre les différentes sortes d'événements avait aussi été envisagée, mais le modèle ci-dessus permet de faire ressortir les attributs communs. Préalablement, l'administrateur était une classe fille d'utilisateur, mais un simple attribut « admin » de la classe « users » suffisait.

4- Le Front-end

Pour le **front-end**, on a opté pour un style assez simple et épuré. Le but étant principalement la conception d'un outil comprenant l'ensemble des fonctionnalités essentielles. Ainsi on retrouve un home, et différentes pages pour les catégories

importantes :

- Une page membre pour visualiser les utilisateurs de l'outil
- Une page projet pour visualiser la liste des projets en cours, leurs échéances, mais aussi les informations relatives au chef de ce projet
- Une page événement pour visualiser les événements, les informations utiles à son sujet, ainsi que les coordonnées de son organisateur
- Une page compte permettant à un utilisateur d'accéder à ses informations le concernant
- Un espace administrateur : ce dernier permet à l'administrateur de gérer la plateforme

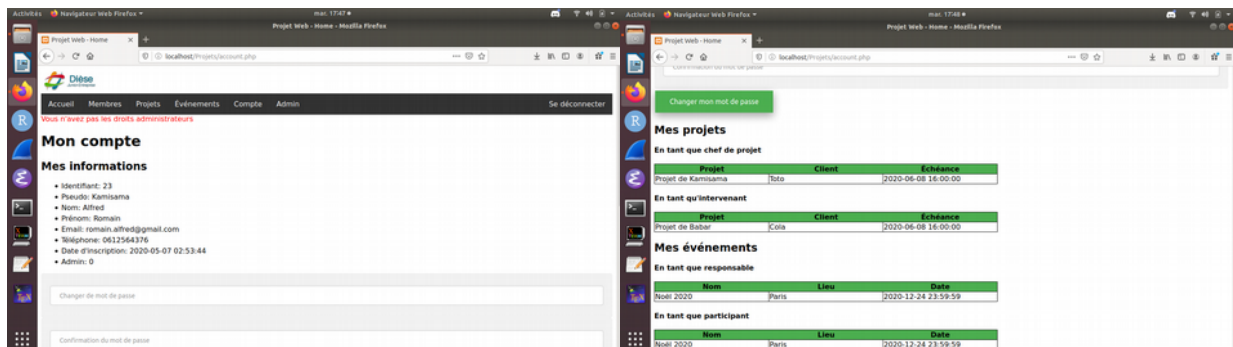
5- Description des fonctionnalités à travers les fichiers

Dans cette partie du rapport, on s'efforcera de décrire au mieux chaque fichier du projet pour faciliter l'utilisation de la plateforme. Ou bien si des problèmes d'utilisation de l'outil surviennent, une explication de chaque fichier pourrait aider à résoudre le problème. *Les fichiers sont ici décrits dans l'ordre alphabétique, pour faciliter la recherche.*

5.1- account.php

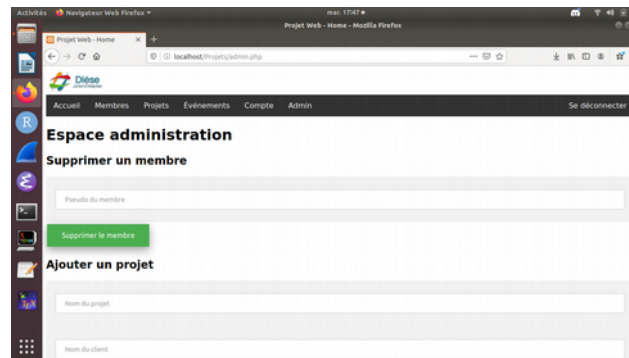
Un utilisateur peut accéder à ses informations grâce à l'onglet « Compte ». Le fichier **account.php** permet d'afficher toutes les informations utiles sur l'utilisateur : ses projets en tant que chef de projet ou en tant qu'intervenant, les événements auxquels il participe ou desquels il est responsable. L'utilisateur peut aussi s'il le souhaite changer son mot de passe dans cet onglet. Un message de confirmation de changement de mot de passe sera alors affiché si le mot de passe est valide.

Un utilisateur ne peut accéder à l'onglet « **Compte** » uniquement s'il est connecté, s'il ne l'est pas un message lui indiquera de se connecter et l'utilisateur sera redirigé sur la page de connexion. Le fichier **admin.php** gère toutes les requêtes faites en conséquence.



5.2- admin.php

Un utilisateur peut avoir des droits administrateurs qui lui procurent quelques options supplémentaires. Il peut se rendre sur l'onglet « **Admin** » qui permet de gérer la base de données. Sur cet onglet un administrateur peut supprimer un membre, ajouter un projet, ajouter un événement ou encore ajouter un intervenant à un projet après avoir rempli le formulaire correspondant. Si ces formulaires ne sont pas valides, l'administrateur se verra retourner un message d'erreur. Un utilisateur peut se rendre sur l'onglet « **Admin** » uniquement s'il possède les droits administrateurs, s'il ne les possède pas un message d'erreur sera affiché et il sera redirigé sur l'onglet « **Compte** ». Le fichier **admin.php** gère tous ces formulaires.

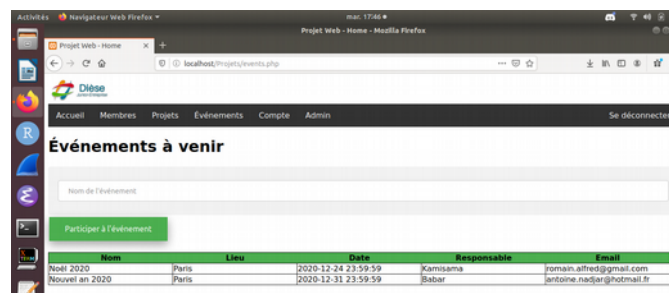


5.3- db.php

Lorsqu'une connexion à la base de donnée est requise on fait appel à db.php qui crée cette connexion. Les événements et les projets dont la date et l'échéance ont périmées sont supprimés de la base de donnée à chaque fois qu'on fait appel à db.php. Cela assure une mise à jour de la **base de donnée**.

5.4- events.php

Un utilisateur peut se rendre sur l'onglet « **Événement** » afin de voir les différents événements à venir organisés par l'association et quelques informations les concernant. Si cet utilisateur est connecté il peut choisir de participer à un événement en entrant le nom de l'événement auquel il souhaite participer. Si cet **événement** n'existe pas un message d'erreur est affiché. Une fois que l'utilisateur décide de participer à l'**événement** la base de donnée est mise à jour. Le fichier **events.php** gère l'affichage et les requêtes passées à la base de données.



5.5- footer.php

Le fichier **footer** permet la **fermeture** des différentes balises ouvertes présentes dans le fichier header.php. Il permet de plus la **fermeture** de la connexion à la base de donnée.

5.6- functions.php

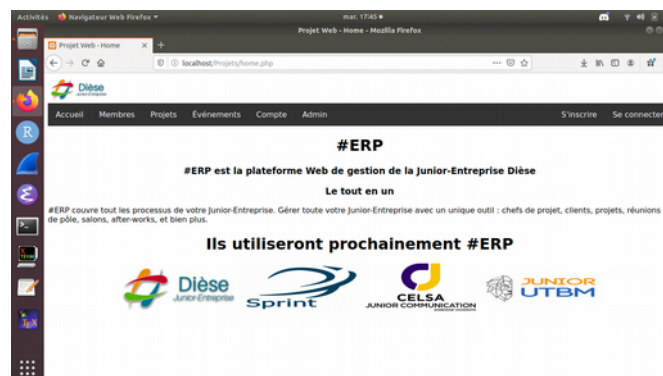
Ce fichier contient une fonction « **loggedOnly** » qui lorsqu'elle est appelée renvoie un message d'erreur à l'utilisateur s'il n'est pas connecté et le redirige vers la page de connexion.

5.7- header.php

Le fichier **header.php** contient tout le code redondant du site : la connexion à la base de donnée, le menu de navigation, le style ...Lorsqu'un utilisateur se connecte le menu de navigation change et ne propose plus l'inscription et la connexion à l'utilisateur mais la déconnexion.

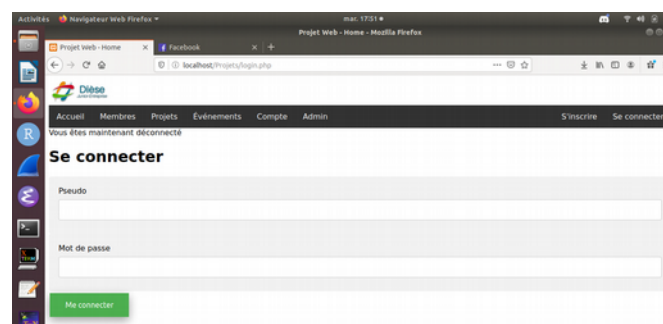
5.8- home.php

L'utilisateur peut se rendre sur l'onglet « **Home** » qui affiche l'enjeu du site. Ce fichier ne contient pas de web dynamique.



5.9- login.php

Un utilisateur peut se connecter dans l'onglet « **Se connecter** » s'il est enregistré. Une fois connecté le membre est redirigé vers l'onglet « **Compte** » et un message de succès de connexion est affiché. Si le formulaire de connexion n'est pas valide des messages d'erreurs sont affichés.

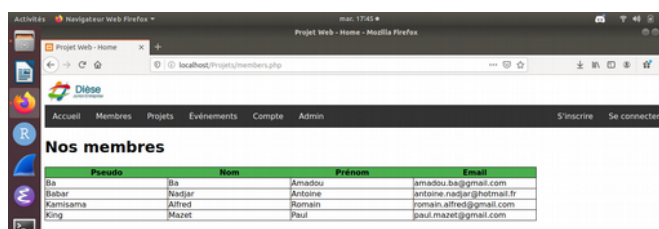


5.10- logout.php

Un membre peut choisir de se déconnecter via l'onglet « **Se déconnecter** » . Dans ce cas il est redirigé vers l'onglet « **Se connecter** » et un message de succès de déconnexion est affiché.

5.11- members.php

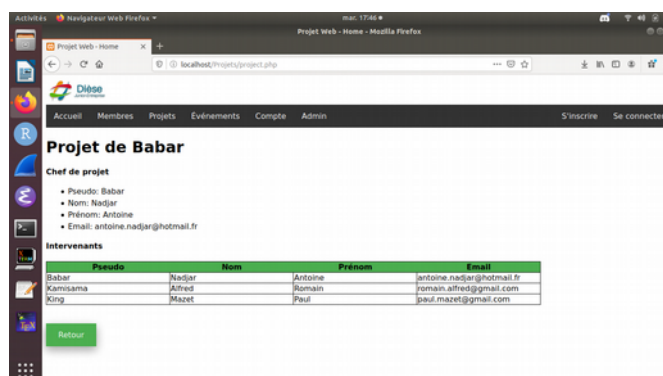
Un utilisateur peut avoir besoin d'accéder aux informations de membre de l'association. Dans ce cas il peut se rendre sur l'onglet « **Membres** » qui affiche des informations utiles sur les membres. Le fichier **members.php** gère l'affichage et les requêtes passées à la base de données.



Pseudo	Nom	Prénom	Email
Ila	Ila	Amadou	amadou.bal@gmail.com
Babar	Nadjar	Antoine	antoine.nadjar@hotmail.fr
Kamisama	Alfred	Romain	romain.alfred@gmail.com
King	Mazet	Paul	paul.mazet@gmail.com

5.12- project.php

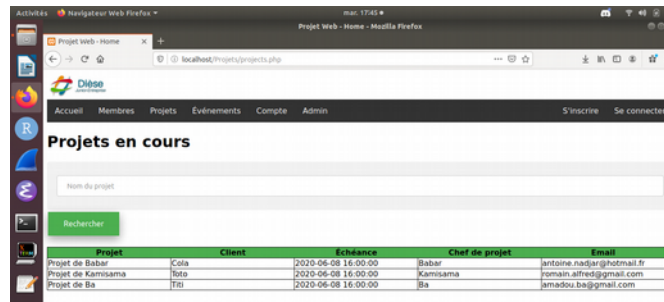
Le fichier **project.php** gère la demande passé par le formulaire du fichier **projects.php** et affiche le détail d'un projet : les informations du chef de projet et les informations des différents intervenants sur ce projet. S'il a fini de consulter un projet il peut retourner à la page « Projets » via l'onglet « Retour » présent dans le corps de la page.



Pseudo	Nom	Prénom	Email
Babar	Nadjar	Antoine	antoine.nadjar@hotmail.fr
Kamisama	Alfred	Romain	romain.alfred@gmail.com
King	Mazet	Paul	paul.mazet@gmail.com

5.13- projects.php

Un utilisateur peut consulter les différents projets en cours classés par date d'échéance. Pour cela il doit se rendre sur l'onglet « **Projets** » Des informations sur le chef de projet sont aussi disponibles. S'il le souhaite il peut avoir le détail d'un projet en particulier en recherchant le nom d'un projet. Si le nom n'est pas valide un message d'erreur sera affiché sinon l'utilisateur sera redirigé vers une autre page géré par **project.php**. Le fichier **projects.php** gère l'affichage des différents projets. Le formulaire de recherche est lui traité par **project.php**.



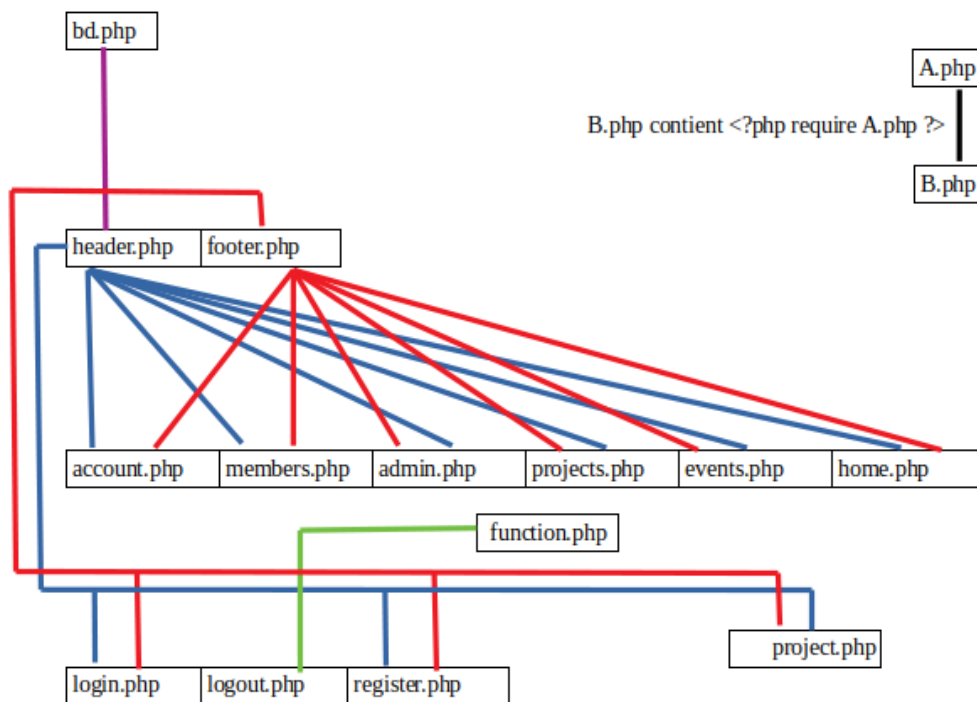
5.14- register.php

Un utilisateur peut choisir de s'inscrire en allant sur l'onglet « **S'inscrire** ». Il doit alors fournir plusieurs informations afin de finir son inscription. Si certaines des informations sont manquantes, l'utilisateur ne pourra pas s'inscrire et un message d'erreur sera affiché. Si l'adresse mail ou le pseudo entrés par l'utilisateur sont déjà enregistré dans la base de données il ne pourra pas s'inscrire et des messages d'erreurs seront communiqués à l'utilisateur. Deux membres ne peuvent pas posséder les même pseudos ou les même adresses électroniques. Le fichier **register.php** gère le formulaire d'inscription.

6- Liaison entre front-end, requêtes PHP et base de données

Le site est organisé sous forme d'un menu donnant accès à plusieurs onglets. Lorsque l'on se rend sur l'un des onglets, on souhaite pouvoir interagir sous le même format, indépendamment de l'onglet choisi. C'est pourquoi les fichiers PHP relatifs à chaque onglet requièrent les mêmes fichiers header.php et footer.php, en début et fin de script . Ce même fichier header.php nous permet d'établir une liaison avec la base de données car ce dernier requiert un fichier bd.php .

Architecture



7- Note de conclusion

A travers ce projet, l'on a donc tenté de mêler l'apprentissage du développement web, dans tous ses aspects, de la gestion de projet, à la modélisation, à la programmation en elle-même, à la conception d'un outil utile à une association de l'école. Effectivement, à la Junior-Entreprise Dièse, un ERP se fait attendre depuis plusieurs années. Concevoir un tel outil a donc à la fois été un challenge à enfin accomplir, et une mission d'utilité première pour cette association de l'école.

L'outil est bien entendu améliorable, mais se veut avant tout un prototype auquel de nouvelles fonctionnalités seront ajoutables aisément. Pour ce qu'il en est du design, celui-ci pourra par la suite être adaptable à n'importe quelle autre Junior-Entreprise.