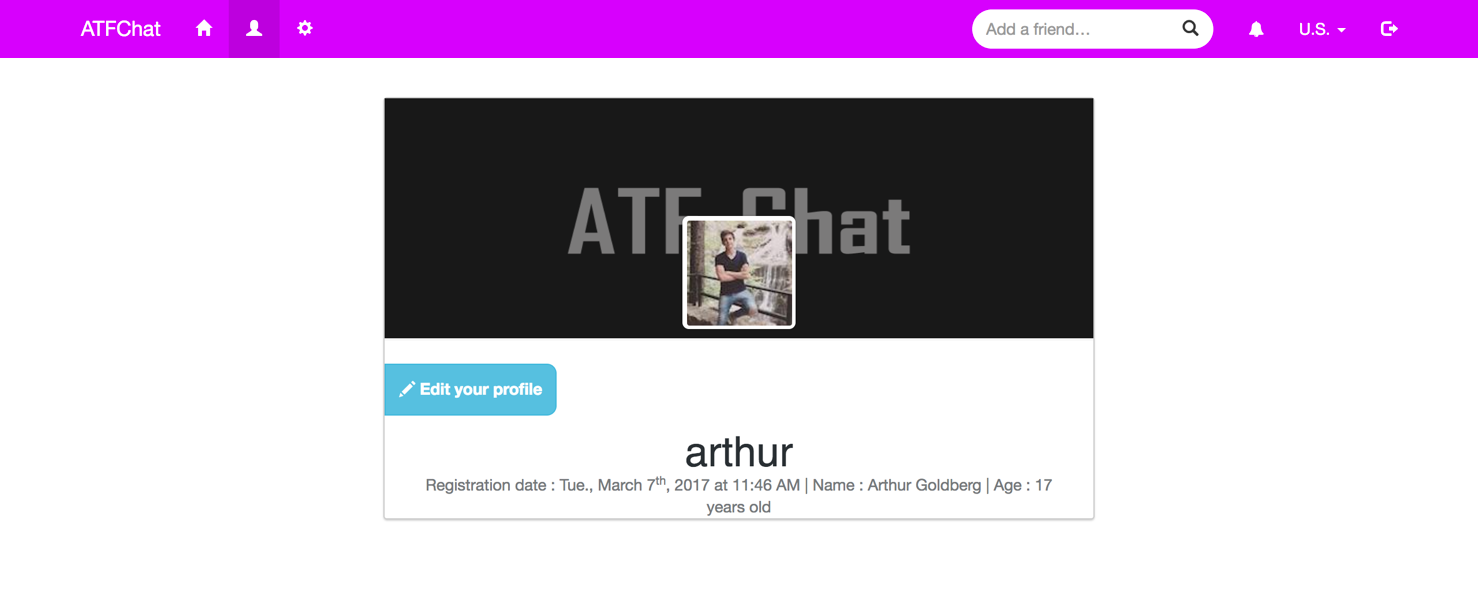
ATF CHAT

Dossier pour le Baccalauréat d’Informatique et Sciences du Numérique

Tao SCHREINER

<http://atfchat.scola.ac-paris.fr/>

16 mai 2017



Sommaire

1. **Introduction**
2. **Guide d’utilisation**
   1. Par quels moyens peut-on retrouver ses amis
   2. Utilisation du système de messagerie instantanée
   3. Comment accéder à la version anglophone de la messagerie instantanée
   4. Comment peut-on modifier ses informations personnelles
3. **Structure du projet**
   1. Quels sont les technologies utilisées
   2. Apprentissage des connaissances nécessaires au projet
   3. Répartition des taches
   4. Structure globale de la messagerie instantanée
   5. Présentation de l’Architecture MVC
   6. Présentation de la cryptographie utilisée
   7. Stratégie adoptée et description de la partie faite individuellement
4. **Conclusion**
5. **Bibliographie**
6. **Annexe**

5.1 Système d’envoi et de réception des messages

5.2 Système de cryptographie

* 1. Système de gestion du membre
  2. Cahier des charges
  3. Planning prévisionnel

1. Introduction

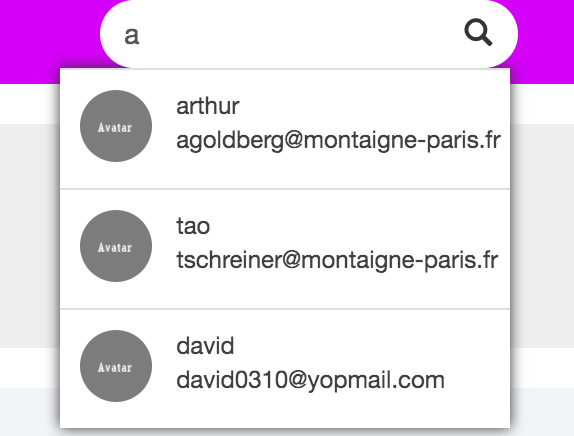
Durant notre année de Terminale S Arthur Goldberg et moi-même avons décidé de programmer un système de messagerie instantanée cryptée dans le cadre de notre projet pour notre spécialité Informatique et Sciences du Numérique. Nous avons choisi de réaliser ce projet car c’était pour nous un véritable défi où nous devions partager nos connaissances afin de réussir à le mener à bien. Il était aussi important pour nous de trouver un sujet en lien avec la sécurité informatique, et une messagerie instantanée nous a semblé répondre parfaitement à ce critère. Afin d’avoir une idée précise de ce que nous devions créer nous avons regardé autour de nous les sites-web et applications similaires à notre projet tels que WhatsApp, Facebook Messenger, et autres.

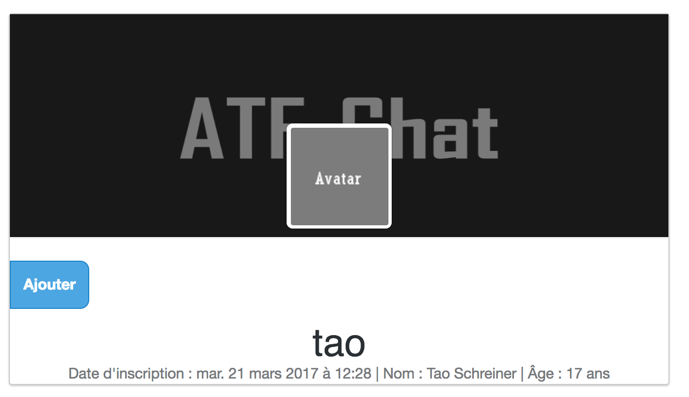
2. Guide d’utilisation

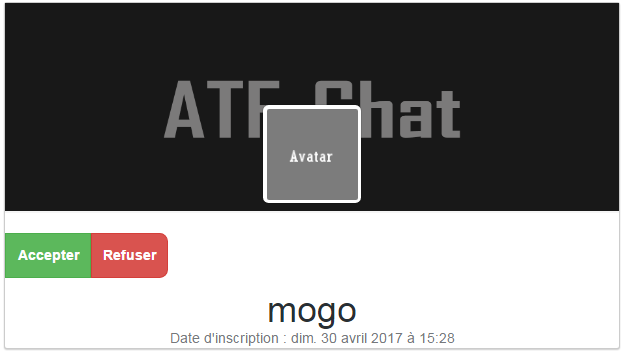
2.1 Par quels moyens peut-on retrouver ses amis

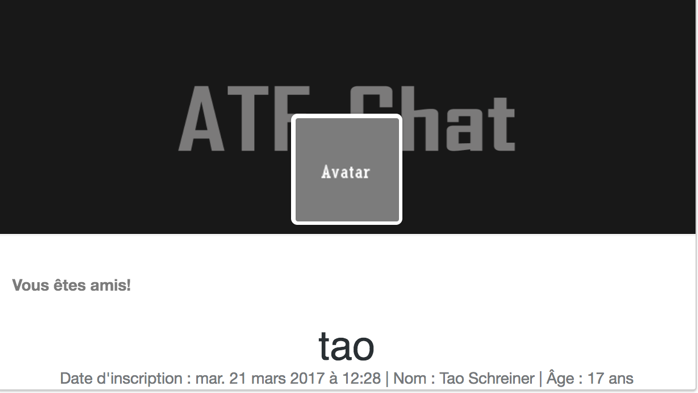
Afin de retrouver l’un de ses amis, il faut taper son pseudonyme dans la barre de recherche. Une fois trouvé, on est redirigé vers son profil. Ce profil permet de vérifier si c’est bien la personne que nous recherchons. On peut trouver sur son profil un bouton permettant de l’ajouter à notre liste d’amis. Il faut ensuite attendre l'approbation de votre (futur) ami et, une fois celle-ci obtenue, vous avez la possibilité de discuter ensemble.

Si c’est vous qui recevez une demande d’amitié, celle-ci arrivera dans vos notifications et vous aurez la possibilité de répondre à ladite demande.

  
Barre de recherche d'amis

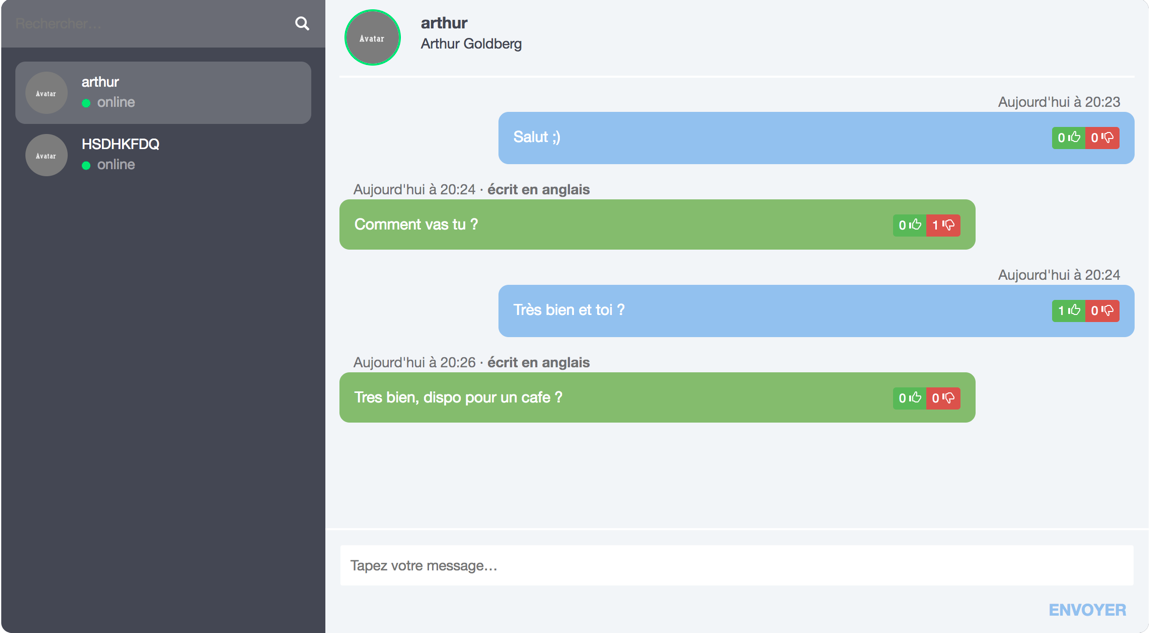
  
Profil avec bouton de demande d'ajout à la liste d'amis

  
Profil d'une personne vous ayant demandé en ami

  
Profil d'un ami

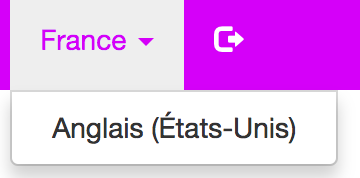
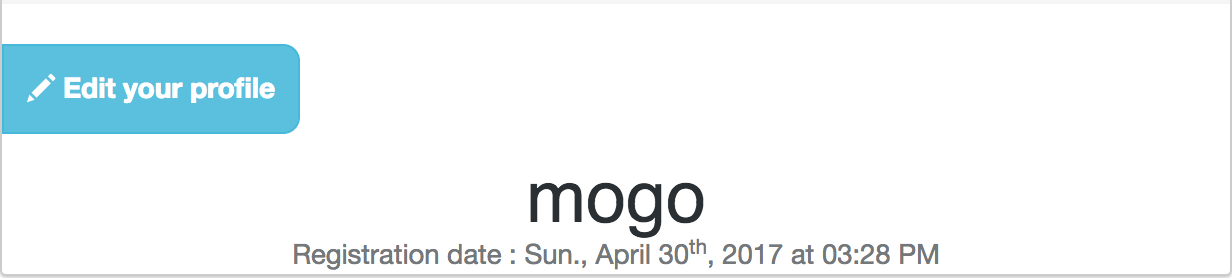
2.2 Utilisation du système de messagerie instantanée

Afin de discuter avec votre ami récemment ajouté, il suffit de le sélectionner dans votre liste de contact, une fois sélectionné il vous suffit d’écrire un message et d'appuyer soit sur la touche entrée soit sur le bouton *Envoyer*. De même il est indiqué sur la gauche de votre fenêtre de discutions vos conversations les plus récentes.



2.3 Utilisation de la version anglophone de la messagerie

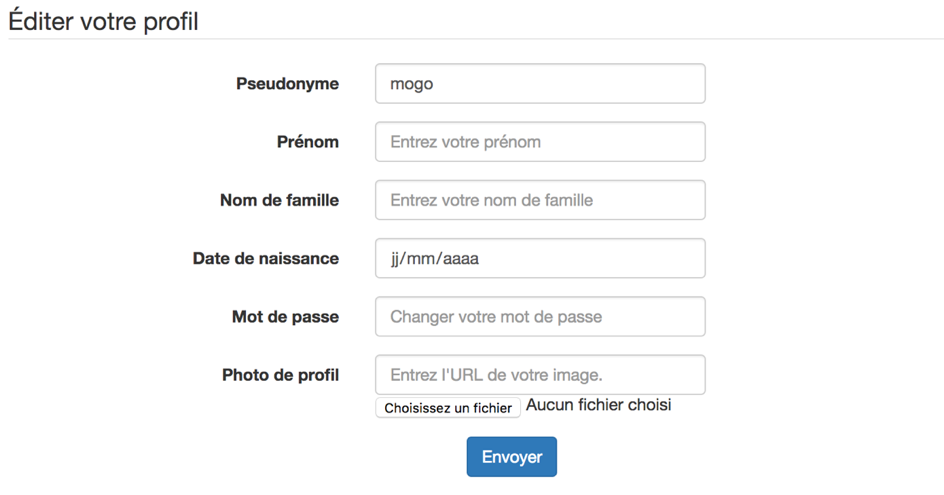
Nous avons décidé de créer un système de traduction afin que notre site soit accessible par le plus large public possible. Il est possible pour l’utilisateur de choisir sa langue préférée dans la barre de navigation grâce à un menu déroulant. Ainsi il est aussi simple pour un utilisateur anglophone de comprendre l’intégralité de notre messagerie instantanée que pour un français. À noter que même si le système n’est disponible que dans ces deux langues aujourd’hui, rajouter une autre langue ou modifier une déjà intégrée est très simple. De plus la dernière langue utilisée est mémorisée grâce à un cookie et la langue apr défaut du site (pour les nouveaux arrivants par ex.) peut être réglée par les administrateurs simplement.



2.4 Comment peut-on modifier ses informations personnelles

Nous avons voulu donner la possibilité à l’utilisateur de modifier ses informations personnelles tels que sa photo de profil, son pseudonyme, son nom et prénom ainsi que son mot de passe. Ces informations sont très utiles pour se retrouver entre amis, il était donc essentiel pour nous qu’elles soient présentes sur notre site.

De même, l’utilisateur doit pouvoir rajouter une photo de profil et/ou de couverture simplement. Toutefois, quelques problèmes indépendants de notre volonté nous ont empêché de faire fonctionner ce système aussi bien que nous le souhaitions.



3.Structure du Projet

3.1 Quels sont les technologies utilisées ?

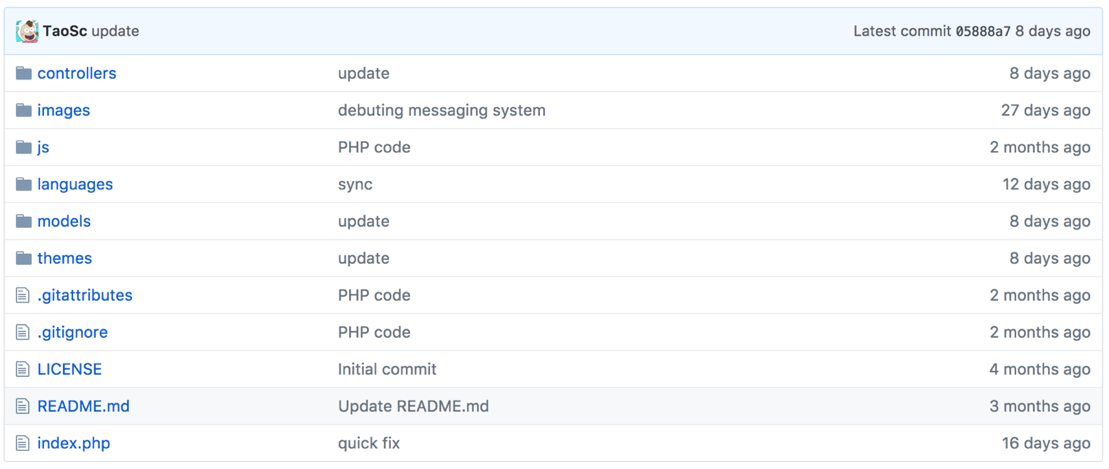
Nous avons utilisé trois types de technologies, il y a d’abord les technologies liées au **front-end** c’est-à-dire la partie visible de la messagerie. Il y a aussi les technologies liées au **back-end**, c’est-à-dire la partie non visible mais gérant la logique du système. Pour finir il y a les **outils liés au travail de groupe**, pour l’organisation. Ces derniers n’ont pas de présence directe dans notre code mais nous ont grandement simplifié la vie.

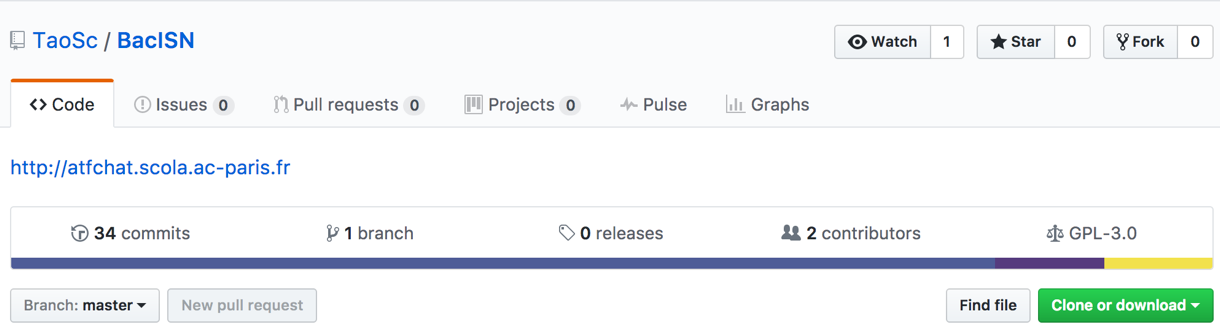
Nous avons utilisé pour le front-end du HTML5/CSS3 qui sont les langages essentiels liés permettant de mettre en place le design d’un site internet. Nous nous sommes servi de Bootstrap ;il s’agit d’un framework web nous permettant de créer le front-end du site en utilisant une série de système pré-conçus avec le principal avantage qu’ils s’adaptent très bien à toute taille d’écran (responsive design). Ainsi nous avons pu, dès les premières semaines, avoir un un mock-up du site fonctionnel aussi bien sur mobile que sur PC. Nous avons aussi utilisé aussi Font Awesome, il s’agit d’une bibliothèque nous proposant toute une série d’icônes vectorielles à notre disposition.

Le JavaScript se place entre le front-end et le back-end car il agit sur le client (le navigateur de l’utilisateur), contrairement au back-end, cependant il est capable de modifier une page selon les commandes souhaitées. Le JavaScript et jQuery (une bibliothèque JavaScript. C’est un pré-requis pour Bootstrap mais nous l’avons aussi utilisé ailleurs) nous a principalement servi dans l’envoi des messages grâce au protocole AJAX qui permet des envois et réception de données sans rechargement de page.

Nous avons utilisé 2 bibliothèques JavaScript : déjà *list.js* qui est une bibliothèque JavaScript permettant de simplifier le système de filtres sur la recherche. Pour finir nous avons utilisé un plugin jQuery se nommant *timeago.js* nous permettant de gérer le temps s’affichant lors de l’envoi de message.

Pour le fonctionnement en arrière-plan nous avons utilisé deux langages de programmation : le PHP et le SQL. Nous avons utilisé le PHP afin de produire des pages web dynamiques via un serveur HTTP. Nous avons choisi le PHP car nous avions tous les deux un minimum de connaissances sur le langage. Nous avons utilisé MySQL pour une base de donnée où nous stockons les informations des membres et messages envoyés par ceux-ci.

 Pour ce qui est des outils de travail collaboratif, nous avons utilisés Git et GitHub afin de travailler de manière coordonnée. Git est logiciel de gestion de versions décentralisés. Il nous a permis d’avoir toutes les versions au fur à mesure de notre avancée, ainsi en cas de problème avec le moindre changement que nous effectuions il nous suffisait de retourner préciser au moment où cela fonctionnait pour savoir quel ligne de code était fautive. Quant à GitHub il nous a permit d’avoir une interface graphique claire et simple pour Git et nous a permis d’héberger gratuitement le code du système.



3.2 Apprentissage des connaissances nécessaires au projet

Pour la réalisation de ce projet nous avons dû acquérir de nouvelles connaissances, que ce soit l’apprentissage de nouveaux langages ou encore un apprentissage d’une manière de coder.

Arthur s’est notamment formé au PHP ainsi qu’à la POO et à l’architecture MVC et a appris rapidement.

Dans ce projet, Tao était familié avec la plupart des outils utilisés. Toutefois il n’avait pas utilisé certains d’entre eux depuis longtemps et il a donc dû « ré-apprendre » par exemple certaines bases du JavaScript qu’ il a pu oublier. De même, ses capacités d’utilisation d’AJAX et de jQuery étant assez faibles, il n’ai eu de cesse de me référer aux documentations adéquates quand il dû coder certaines parties du JavaScript.

Il était aussi intéressant de voir comment certaines de nos méthodes pour résoudre des problèmes différaient.

3.3 Répartition des taches

Dès le début de notre projet, nous avons réalisé un programme de travail afin de fixer les objectifs de chacun pour la réalisation de la messagerie instantanée.

Tao s’est occupé d’une grande partie du back-end, il a notamment créé le système de gestion des membres. Il s’est chargé de la création d’un module de multilinguisme. Il est à l’origine de la création du système de modification de profil, et à celui de gestion des médias dans le but d’un changement de photo de profil. Il est à l’origine de la mise en place du modèle MVC dès le début de notre projet. De même que de la relation entre le PHP et la base données et l’architecture de celle-ci. Il a aussi codé le système de vote (pouce vert/rouge). Pour ce qui est du front-end, Tao s’est occupé de l’Inscription/Connexion. Il s’est aussi occupé du design de la modification de profil.

Quant à Arthur il s’est occupé de la création du système d’envoi et de réception de message pour les utilisateurs, de même il est aussi à l’origine de l’intégration au front-end du bloc de discussion. Il s’est également occupé du système d’amitié entre les utilisateurs afin de leur permettre de discuter ensemble. Pour finir, il s’est aussi occupé de la création du système de notifications et du système de types de membres (bannis, administrateurs, membres classiques). Pour finir il s’est occupé du moteur de recherche servant à l’ajout d’amis permettant la discussion.

Pour ce qui est de la cryptographie nous avons tous les deux travaillés dessus. La cryptographie sera aussi détaillé dans la sous-partie structure globale du projet.

3.4 Structure Globale de la messagerie instantanée

Nous avons cherché à expliquer simplement en quelques lignes comment à partir de l’arrivée sur notre page d’accueil l’utilisateur pouvait discuter avec l’un de ses amis :

1.Tout d’abord lorsque l’utilisateur arrive sur le site web, il y a deux possibilités soit l’utilisateur s’inscrit, soit il se connecte dans le cas où il possède déjà un compte ;

2.Une fois l’utilisateur connecté, il arrive sur la dernière page à laquelle il a accédé. Le plus souvent il s’agit de la page d’accueil ;

3.L’utilisateur doit ensuite rechercher un de ses amis grâce à barre de recherche afin de l’ajouter dans ses contacts ;

4.Une fois la requête envoyée, son ami doit l’accepter ;

5.Lorsque les deux utilisateurs sont amis, ils ont la possibilité de communiquer ensemble via le bloc de discussion.

3.5 Présentation de l’Architecture MVC

Nous avons utilisé une architecture Modèle-Vue-Contrôleur. Il s’agit d’un patron de conception (design pattern en anglais) très utilisé dans la réalisation des sites web. C’est une solution éprouvée et reconnue permettant de séparer l'affichage des informations, le traitement des actions de l'utilisateur et l'accès et la gestion aux données.

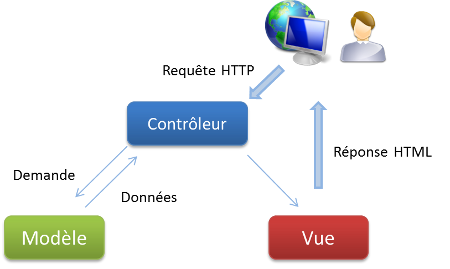
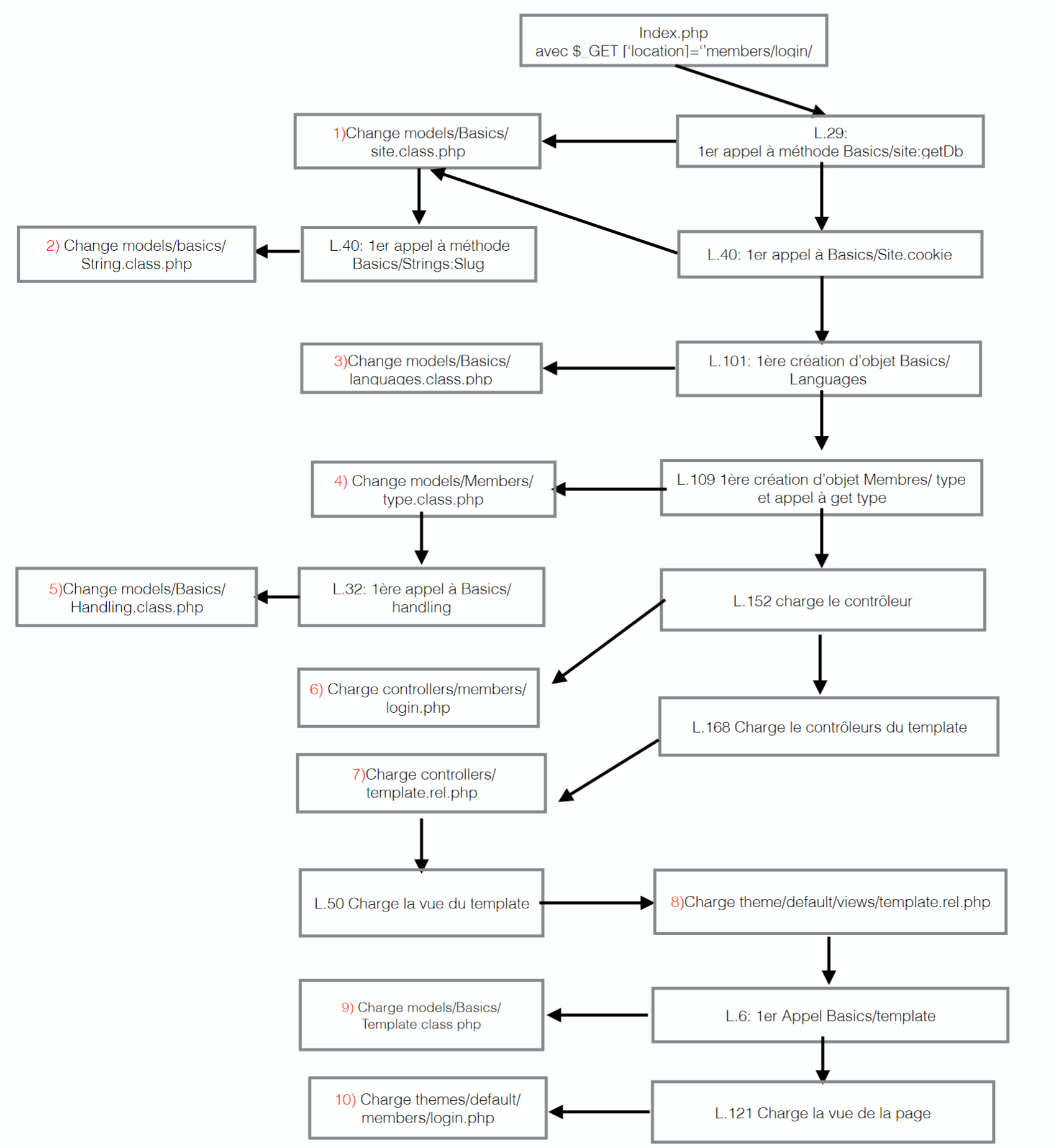
Concrètement c’est à dire que lorsque l’utilisateur demande d’accéder à une page, il n’accède pas vraiment à la page en question. Il accède en réalité à un fichier maître qui s’occupe de rapatrier une multitude de modèles,vues et contrôleurs différents nécessaires à la page demandé. Nous avons réalisé une carte mentale permettant de mieux comprendre le principe d’architecture MVC dans le cas de la page de connexion :

Schéma expliquant les fichiers appelés lors de la connexion

3.6 Présentation de la cryptographie utilisée

Nous nous sommes basés sur un système de cryptage utilisant une fonction symétrique. Elle est aussi appelée la cryptographie à clé secrète. Elle permet à la fois de crypter un message mais aussi de le déchiffrer avec une même clé. Nous avons utilisé ce type de cryptographie afin d’avoir des messages cryptés dans notre base de donnée tout en ayant la possibilité de décrypter les messages pour les afficher de manière transparente pour l’utilisateur. Pour cela nous avons utilisé du PHP qui va permettre de crypter le message, il sera ensuite envoyé de manière cryptée sur notre base de données MySQL. Il sera ensuite décrypté par le même modèle PHP avant d’être envoyé à la vue.

* 1. Stratégie adoptée et description de la partie personnelle

Dans ce projet, comme précisé dans le planning prévisionnel je me suis occupé de différentes fonctionnalités. Nous allons ainsi détailler point par point le fonctionnement.

Je me suis personnellement occupé du système d’envoi et de réception de message pour les utilisateurs. Pour cela, j’ai utilisé du PHP ainsi que du MySQL pour sauvegarder les messages dans la base de donnée mais aussi pour les afficher. J’ai aussi utiliser du JQuery avec le protocole Ajax afin d’envoyer les messages sans rechargement de page.

Je suis aussi à l’origine, de l’intégration du front end du bloc de discussion, pour cela nous avons utilisé un bout de code en Html5 et Css3 déjà existant. Nous l’avons adapté pour qu’il puisse s’intégrer au mieux dans notre design. J’ai aussi dû rendre ce bloc de discussion dynamique. Nous l’avons rendu dynamique afin qu’ils puissent afficher les dernières discussions de l’utilisateur et non un design fait par défaut.

Il m’a tenu à cœur de m’occuper de m’occuper du système d’amitié entre les utilisateurs, car j’étais complétement dans le floue sur la manière de coder cela. Je me suis beaucoup renseigner sur des forums afin de coder ce système. Nous allons maintenant détailler un système que j’ai codé. Nous allons observer le système de notifications.

*Algorithme (Controller*)

*controller (template)*

*Si on n’est pas dans l’admin*

*Si le membre est connecté*

*récupère le nombre de notifications dudit membre dans $notificationsCount en appelant Members\Single ::notificationsCount*

*récupère les notifications dudit membre dans $notifications en appelant Members\Single ::getNotifications*

Code Controller

if ($admin) {

...

} else {

$navigation = [ ..

];

$currentMemberObj = new Members\Single($currentMemberId);

$notificationsCount = $currentMemberObj->notificationsCount();

$notifications = $currentMemberObj->getNotifications();

}

Commençons par étudier les contrôleurs, on vérifie d’abord deux conditions, que le membre n’est pas dans la partie administrateur, on vérifie aussi que le membre est connecté. On demande dans ces cas-là, deux informations au model. Tout d’abord le nombre de notifications, et par la suite on récupère les notifications.

*Algorithme (View)*

*View (template)*

*Si le membre est connecté*

*Classe boostrap pour le placement dans la barre de navigation*

*Classe menu déroulant*

*Classe font-awesome pour l'icon d'une cloche*

*Si le nombre de notifications est supérieur à 0 ça affiche le nombre de notifications à droite de la cloche*

*Quand l'on clique sur la cloche est que le nombre de notification est égal à 0*

*Cela renvoie vous n’avez aucune notification*

*Quand l'on clique sur la cloche et que le nombre de notification est supérieur à 0.*

*Cela renvoie à une notification cliquable qui renvoie au profil*

*Ca renvoie que la requête d'amie a été envoyé par " "*

Code View

<?php

if ($currentMemberId) {

?> <ul class="nav navbar-nav notification pull-right">

<li class="dropdown">

<a data-toggle="dropdown" href="#null"><i class="fa fa-bell"></i><?php if (isset($notificationsCount) AND $notificationsCount > 0) echo ' ' . $notificationsCount;?></a>

<ul class="dropdown-menu"><?php

if (empty($notifications))

echo $clauses->get('you\_do\_not\_have\_any\_notification');

foreach ($notifications as $notificationLoop)

echo '<li>

<a href="' . $linksDir . 'members/' . $notificationLoop['sender']['slug'] . '/">

friend request from ' . $notificationLoop['sender']['nickname'] . '

</a> </li>' ;?> </ul> </li> </ul>

<form class="navbar-form navbar-right">

<div class="form-group">

<div class="input-group">

<input type="text" class="form-control" id="search-box" placeholder="<?php echo $clauses->get('search\_friends\_placeholder'); ?>">

<div class="input-group-btn">

<button type="submit" class="btn btn-default" title="<?php echo $clauses->get('search'); ?>"><span class="glyphicon glyphicon-search"></span></button>

<div id="display-results"></div>

</div>

</div>

</div>

</form><?php }?>

Nous nous intéressons maintenant aux vues, tout d’abord il faut observer que le code s’exécute dans l’unique cas où le membre est connecté, autrement dit s’il possède un id. La liste à puce se place à droite sur la barre de navigation. La balise <li> représente un élément de la liste à puce déclaré dans la balise <ul> précédemment. A l’intérieur d’une balise < a> pour les liens, nous utilisons Font-awesome afin d’avoir l’icône d’une cloche symbole des notifications sur internet.

Par la suite on regarde si l’utilisateur à des notifications. Dans le cas où l’utilisateur a des notifications, alors on affiche le nombre à côté de la cloche.

Ensuite nous appelons la classe « dropdown » afin de créer un menu déroulant. Nous avons ensuite une condition avec deux possibilités, soit l’utilisateur n’a pas de notifications dans ce cas-là on affiche une chaine de caractère expliquant que l’utilisateur n’a aucune notification. Dans le cas où l’utilisateur a des notifications, on crée un lien cliquable grâce à la balise <a> renvoyant vers le profil de la personne ayant envoyé la notification. Nous créons aussi le visuel en affichant le nom de la personne ayant envoyé la notification.

Algorithme (Model)

*Model (members) Classe Members\Single*

*Méthode notificationsCount*

*Appelle méthode Basics\Handling::countEntries pour compter le nombre de notifications où le sujet est l'id du membre*

*Méthode getNotifications*

*(Même condition que notificationsCount)*

*Récupère les notifications en les classant par date décroissante pour chaque notification ;*

*Récupère toutes les infos du membre ayant reçu et ayant envoyé*

*Méthode sendFriendRequest*

*Si les conditions pour envoyer une demande d'amis sont respectées*

*Rajoute dans table notifications les informations de ladite demande d'ami*

*Méthode acceptFriendRequest*

*S’ils peuvent être copains*

*Enlève la notification de demande*

*Méthode cancelFriendRequest*

*S’il y a une demande d'amis en cours mais que l'un des deux ne veut pas être copains*

*Enlève la notification de demande*

Code models

<?php

namespace Members;

class Single {

public function notificationsCount() {

return \Basics\Handling::countEntries('notifications', 'subject\_id = ' . $this->member['id'] . ' AND seen = \'0\'');

public function getNotifications() {

$request = \Basics\Site::getDB()->prepare('SELECT \* FROM notifications WHERE subject\_id = ? ORDER BY created\_at DESC');

$request->execute([$this->member['id']]); $notifications = $request->fetchAll(\PDO::FETCH\_ASSOC);

foreach ($notifications as $key => $notificationLoop) {

$notifications[$key]['sender'] = (new Single($notificationLoop['user\_id']))->getMember();

$notifications[$key]['receiver'] = $this->member;

} return $notifications;

} public function sendFriendRequest($requestedId) {

$requestedMember = (new Single($requestedId))->getMember(); i

if ($this->member['type\_id'] != 3 AND $requestedMember['type\_id'] != 3AND !$this->befriend($requestedId) AND !$this->requestPending($requestedId, true)) {

$request = \Basics\Site::getDB()->prepare('INSERT INTO friend\_requests (from\_u, to\_u) VALUES (?, ?)'); $request->execute([$this->member['id'], $requestedId]);

$request = \Basics\Site::getDB()->prepare('INSERT INTO notifications (subject\_id, name, user\_id) VALUES (:subject\_id,

:name , :user\_id)');

$request->execute(['subject\_id'=> $requestedId, ’

name' => 'friend\_request\_sent',

'user\_id' => $this->member['id']

]); return true; } else return false; }

public function acceptFriendRequest($senderId) {

$requestedMemberObj = new Single($senderId);

$requestedMember = $requestedMemberObj->getMember();

if ($this->member['type\_id'] != 3 AND $requestedMember['type\_id'] != 3 AND !$this->befriend($senderId) AND $requestedMemberObj->requestPending($this->member['id']))

$request = \Basics\Site::getDB()->prepare('DELETE FROM notifications WHERE user\_id = ? AND subject\_id = ?');

$request->execute([$senderId, $this->member['id']]);

return true; } else return false; }

public function cancelFriendRequest($requestedId) {

$requestedMember = (new Single($requestedId))->getMember();

if ($this->befriend($requestedId)) {

} elseif ($this->requestPending($requestedId, true)) {

$request = \Basics\Site::getDB()->prepare('DELETE FROM notifications WHERE (user\_id = ? AND subject\_id = ?) OR (subject\_id = ? AND user\_id = ?)');

$request->execute([$requestedId, $this->member['id'], $this->member['id'], $requestedId]);

} return true;

Nous allons maintenant étudier les modèles, la fonction notificationCount récupère le nombre de notifications dans la base de donnée que l’utilisateur n’a pas encore vue.

La fonction getNotification récupère toutes les notifications où l’id du membre connecté et celui de l’utilisateur demandé sont présents dans les champs et les organise de manière décroissante. Cette fonction montre aussi celui qui envoi et celui qui reçoit la notification. En retour elle renvoi les notifications.

La table de notifications est aussi utilisée dans les fonctions acceptant une requête d’ami et celle annulant une requête d’ami, dans les deux cas elle supprime la notification associée de la table, celle-ci n’ayant plus lieu d’être.

4. Conclusion

Comme expliqué dans la bibliographie, nous n’avons pas codé nous même une partie du front-end. Il y a différentes raisons à cela. Tout d’abord un manque de temps, le temps de la création de projet étant relativement court, nous avons préféré nous focaliser sur le back-end. La deuxième raison pouvant expliquer cela est un manque d’imagination et de connaissances de notre part.

Cependant il est possible d’observer par notre cahier des charges que nous avons réussi à réaliser l’ensemble des taches que nous nous étions fixés.

Pour autant il existe encore quelques améliorations possibles afin de faire de cette messagerie instantanée cryptée un réseau social. Pour cela il serait possible d’avoir des conversations groupées. Il serait aussi possible d’avoir la possibilité de partager des photos avec ses amis.

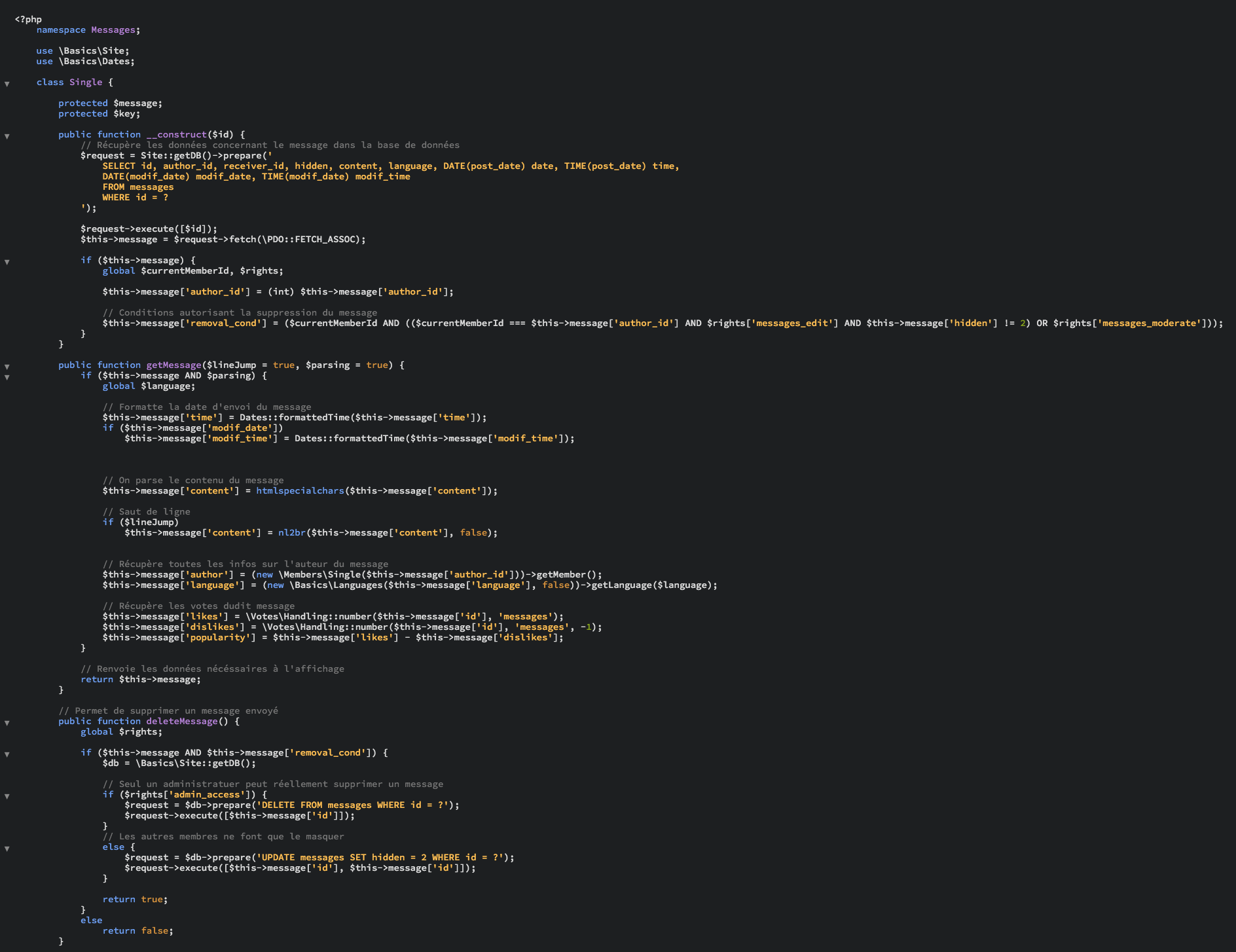
D’un point de vue juridique et éthique ; tous les codes, bibliothèques ou autres dont nous nous sommes servis sont soit sous la licence MIT ou dans le domaine public. Nous avons donc pu les utiliser sans problème dans notre projet que nous avons mis sous licence GPLv3. Bien que plus restrictive qu’une licence MIT, elle nous semble toutefois adaptée et suffisamment ouverte pour permettre des contributions libres sans problèmes.

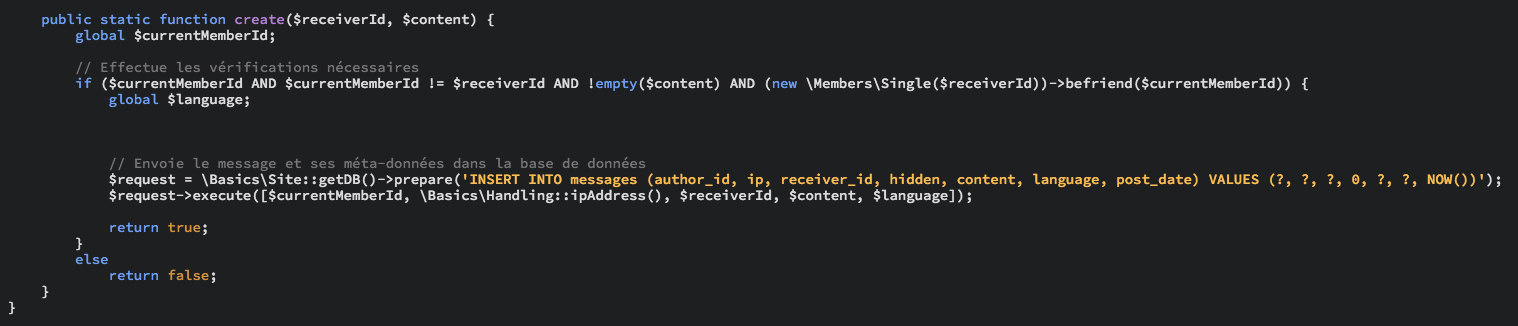
Cette année d’ISN fut un réel plaisir pour moi, cela me permit d’acquérir de nombreuses connaissances en codage mais aussi sur les points de vue juridique, ou encore sur le architectures des réseaux. J’ai beaucoup aimé cette spécialité car elle est extrêmement différente des cours « classiques ». Il y a assez peu de théorie et beaucoup de pratique. Ce projet, ne fut pas ma première expérience de développement informatique mais ce fut pour moi une première dans un groupe, d’autant plus avec une personne que je ne connaissais même pas au début de l’année. Je suis extrêmement content du résultat qui dépasse même nos objectifs sur certains points.

5. Bibliographie

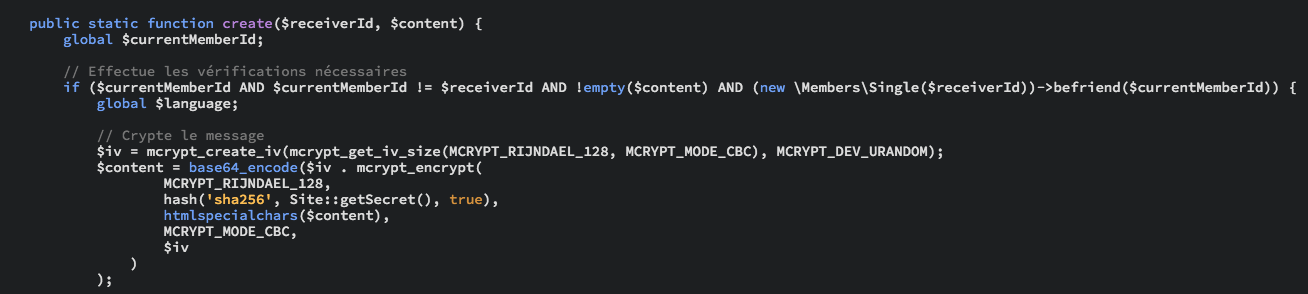
|  |  |
| --- | --- |
| Code source | Codepen :  Design du profil (https://codepen.io/jofpin/pen/olgcn)  Design du bloc de discussion (https://codepen.io/drehimself/pen/KdXwxR) |
| Bibliothèque JavaScript | list.js |
| Plugin jQuery | timeago.js |
| Protocole | AJAX |
| Framework | Bootstrap v3.3.7 |
| Bibliothèque HTML/CSS | Font Awesome |

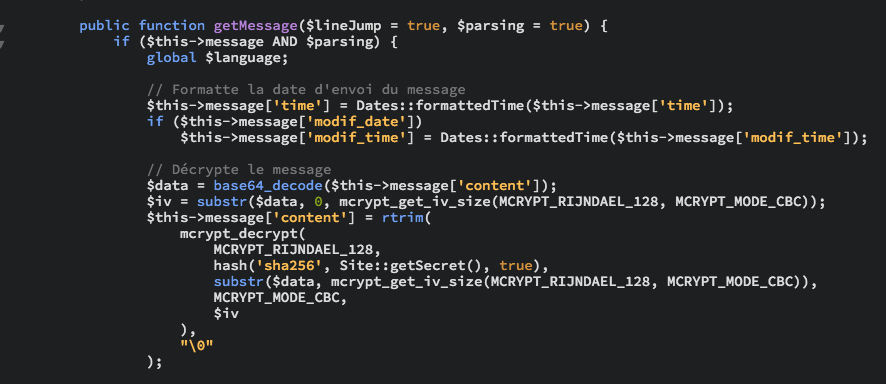
6. Annexe

6.1 Système d’envoi et de réception de messages



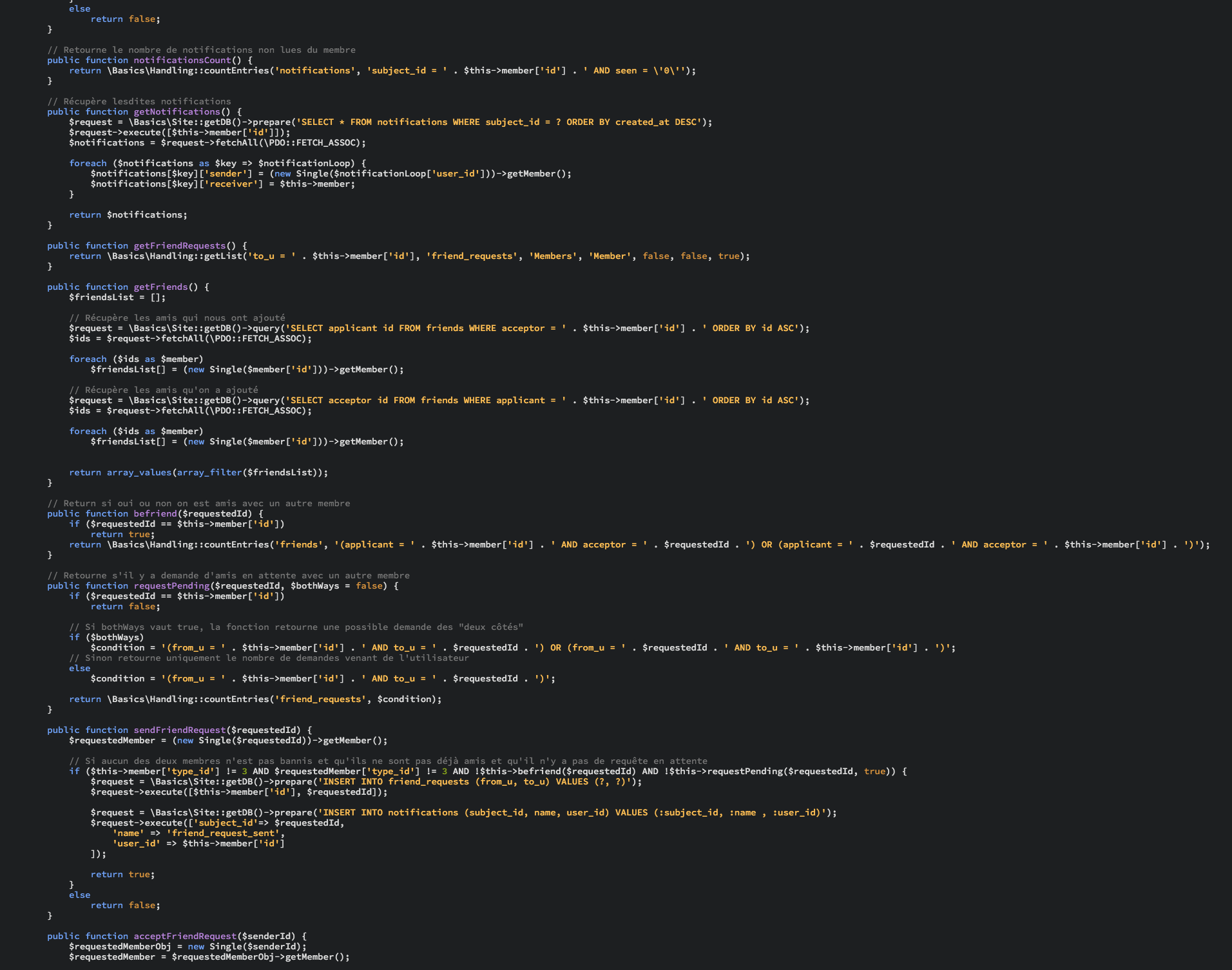
6.2 Système de cryptographie

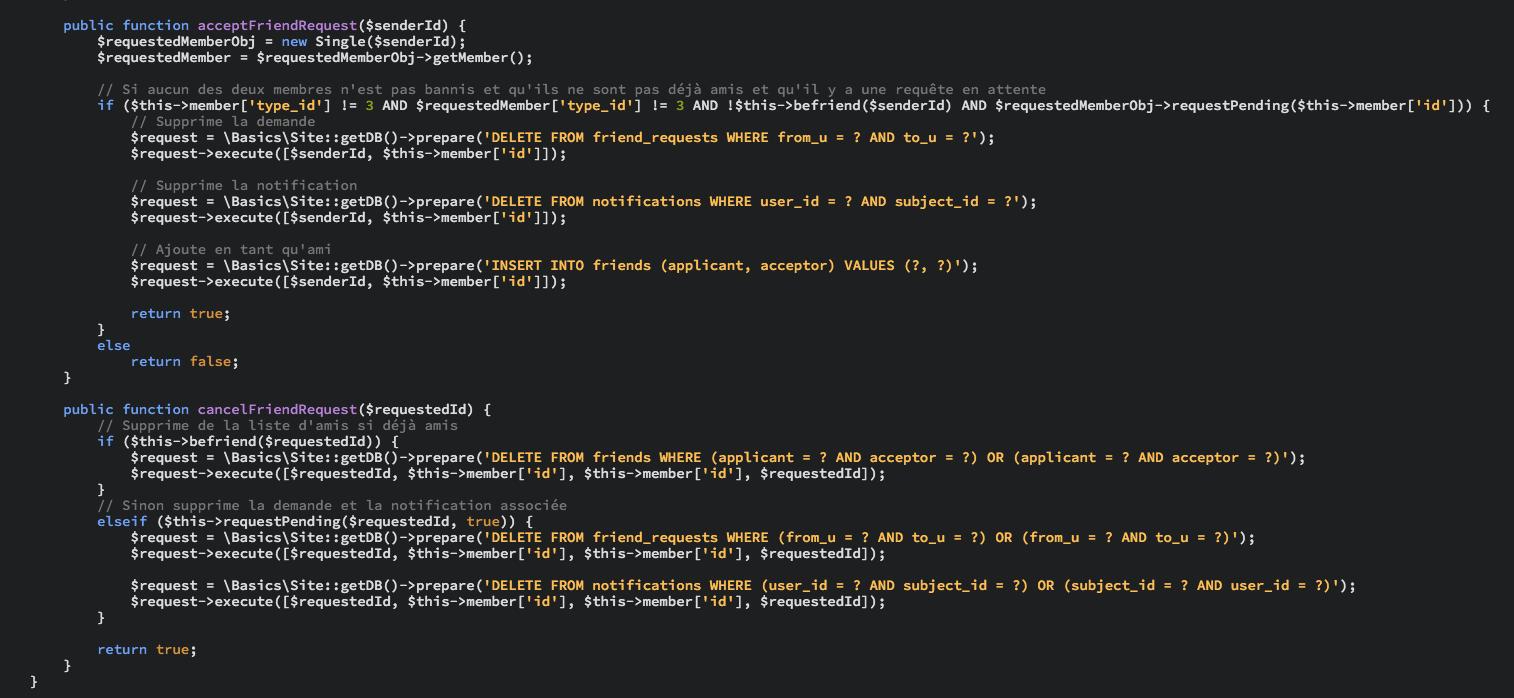




6.3 Système de gestion du membre







6.4 Cahier des charges

../../Downloads/Cahier%20des%20charges%20Arthur%252FTao%252FFrançois%20-%20Feuille%201.pdf

6.5 Planning prévisionnel

../../Downloads/Planning%20prévisionnel%20Arthur%252FTao%252FFrançois%20-%20Feuille%201%20(3).pdf