30 mai 2022

Quentin Chassaing

Titre de développeur web et web mobile

Projet Verse

(uncensored edition)

Sommaire :

1. Présentation (abstract)
2. Présentation du projet
3. Le REAC
4. Cahier des charges
5. Spécifications fonctionnelles
6. Conception
7. Réalisation du projet
8. Outils techniques utilisés
9. Présentation des fonctionnalités
10. Conclusion
11. **Présentation (abstract)**

Hi, I’m Quentin, I’m 30 years old, I live in a small town so called Figeac. Kind of small, but also pretty.

After my high school diploma, I didn’t know what to do. I was kind of lost in front of the possibilities, after high school it's a huge door that opens, and I never took the time to think about what to do next.

I turned to a degree in administration and management, followed by a master’s degree in territorial development that I haven’t finished. That was a turbulent period, stuck between failures and family issues.

But one day, I asked myself what I wanted to do, to be happy and fulfilled.

And a good friend of mine told me that I could be a developer. Maybe he knows me better than I do. In fact, I’m a video game enjoyer, I spend a lot of time on my computer. Actually, being behind my screen is kind of my comfort zone.

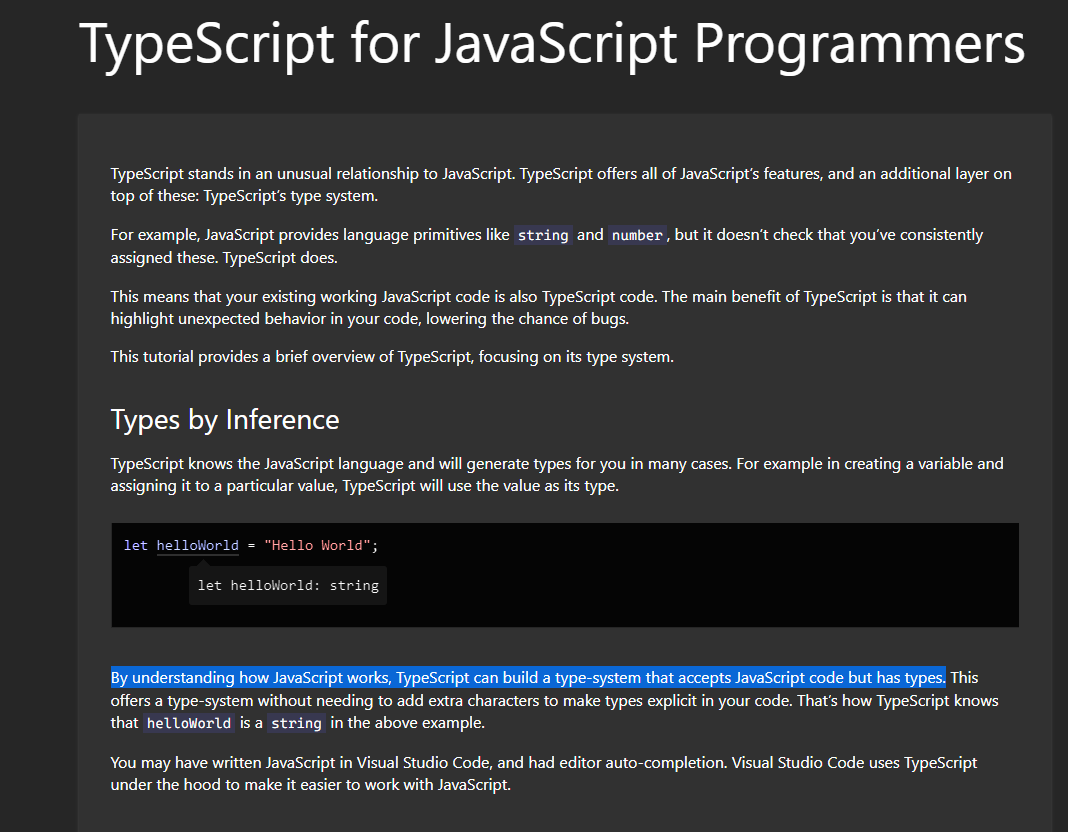
So, I decided to take a look at some online courses, tutorials or any resources you can find on internet. And I remember back then, when I wanted to be a video game developer without knowing what it really means. And I found some courses on Openclassrooms to create a website with **HTML** and **CSS**. And that was a blast, I was hooked by how we can build a website. The languages, the possibilities, a revelation. I started looking for a training course that could give me enough knowledge and enough time to learn more languages. I’m craving for knowledge since I was a kid, but I also need to reassure myself.

And I found in Adrar formation the opportunity to make my journey into the world of developers.

* **Traduction d’une documentation**

Pour la suite, nous continuons avec une brève traduction. **TypeScript** ne fait pas encore partie de mes compétences, mais j’aspire fortement à approfondir l’apprentissage de ce langage qui se fait force de possibilités.

Durant la formation, on nous a enseigné **JavaScript**, qui est un langage que j’affectionne particulièrement. Après quelques recherches, j’ai découvert que **TypeScript** était tout bonnement **JavaScript** avec un typage et donc, offrant la possibilité d’avoir un code bien plus polyvalent.



**Traduction / interprétation :**

**TypeScript** et **JavaScript** entretiennent une relation peu commune. **TypeScript** contient toutes les fonctionnalités que l'on peut retrouver dans javascript avec en plus le système de typage.

Par exemple, Javascript comprend les chaines de caractères et les nombres, mais il ne cherchera pas à en vérifier le type. Alors que **TypeScript** lui, le fait.

De ce fait tout code JavaScript est aussi du **TypeScript**.

L'avantage de **TypeScript** c'est qu'il peut mettre en avant des comportements inattendus dans notre code, et réduire les risques de bugs.

Le tutoriel qui suit va nous permettre d’entrevoir le typage avec **TypeScript**

* Les types par admission :

Sachant que **TypeScript** connait déjà le langage de Javascript, il va, le plus souvent générer un typage à notre place.

Par exemple, en déclarant une variable à laquelle on va assigner une valeur, **TypeScript** va utiliser cette valeur comme type.

ex: let helloWorld = "Hello World";

Toujours sur le même principe, Javascript est connu de **TypeScript** et selon comment Javascript va traiter la valeur,

**Typescript** va alors induire le type de la valeur.

1. **Présentation du projet**

Passons à la présentation du projet, je suis entré à l’Adrar, avec une envie, un besoin. Il fallait que je puisse faire grandir mes compétences et améliorer mes connaissances qui étaient alors bien maigres en comparaison à ce qu’elles sont maintenant.   
Lorsqu’on nous a parlé du projet fil rouge, je me suis alors demandé ce qui pourrait me porter et m’animer suffisamment pour que je puisse apprendre tout en m’amusant.

Le cerveau humain est plus apte à comprendre les choses si on décide de le divertir, et inconsciemment l’information s’implante bien mieux si on ressent une certaine satisfaction en réalisant quelque chose avec laquelle nous nous épanouissons.

C’est ainsi, en suivant ce raisonnement, que nous en arrivons à notre projet, la plateforme **Verse.**



Voilà un petit peu ce que représente ce projet pour moi de manière un peu plus graphique.

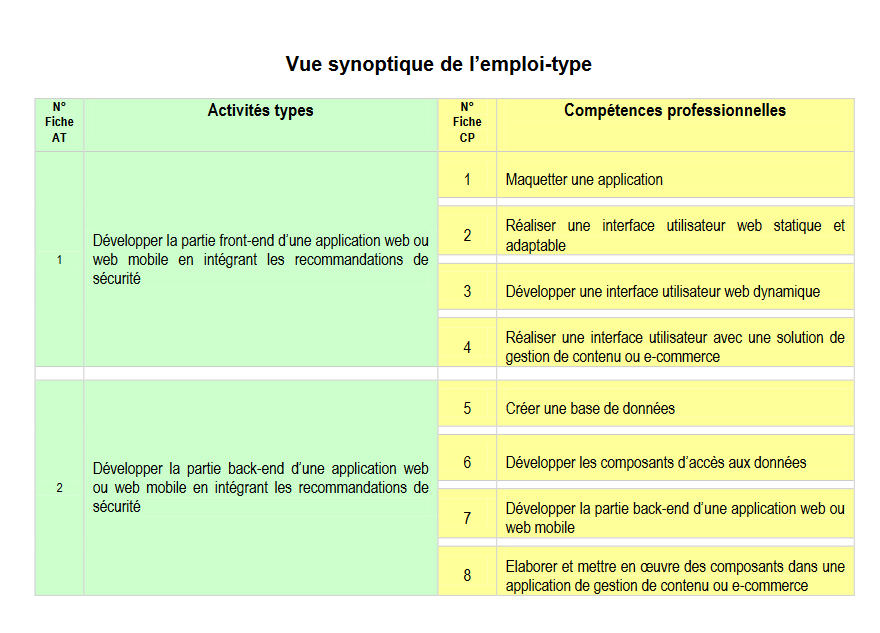
Un bon quart de passion parce que je me suis beaucoup amusé durant la réalisation de ce projet.

Un petit 22-23% de développement, et pourtant c’était surement la partie qui m’intéressait le plus.

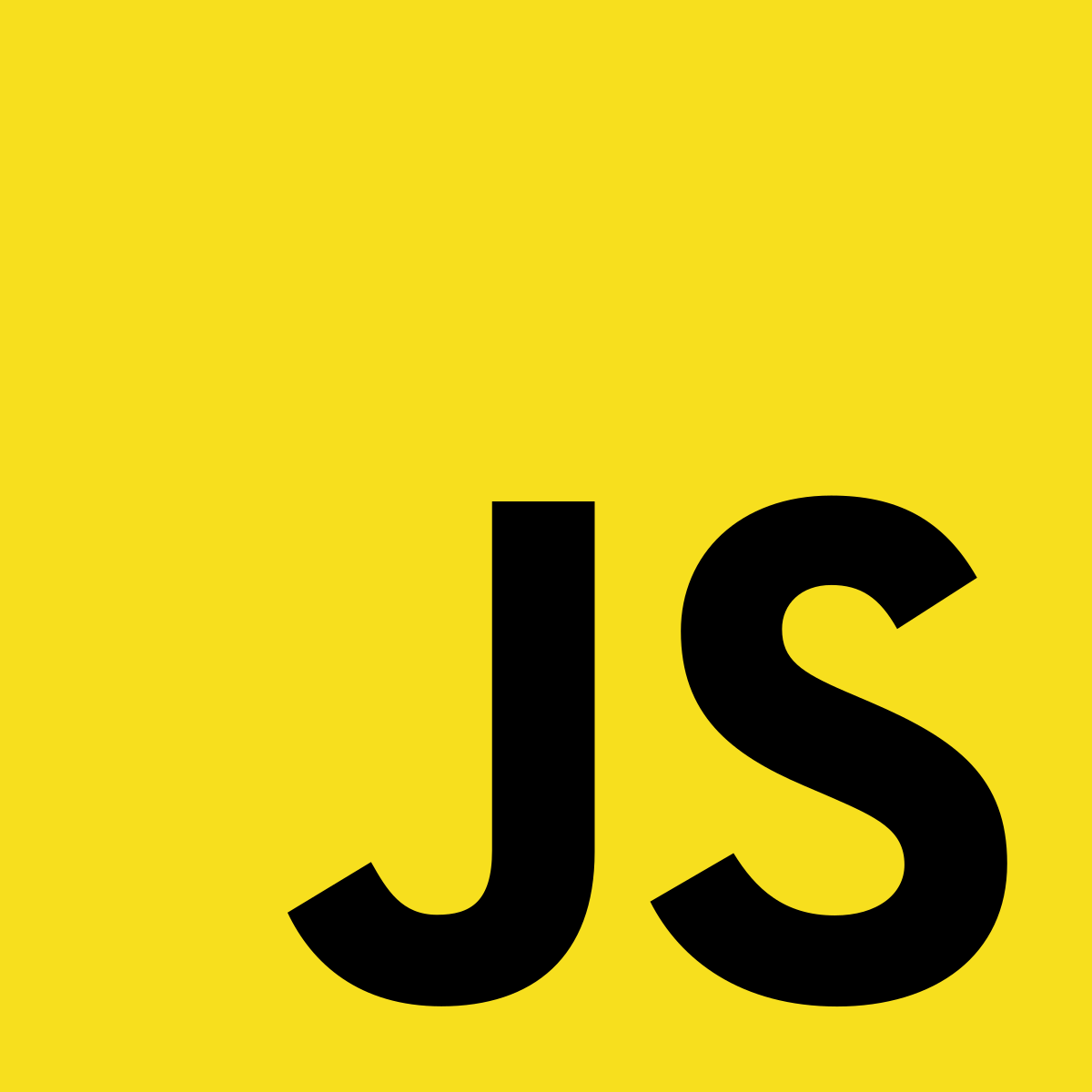
On va dire 6% de préparation à la présentation. Ce qui peut paraitre peu, mais qui au final représente beaucoup en termes d’investissement.

Et le reste, les 47% restants c’est de la réflexion, la recherche logique afin de trouver le bon fonctionnement, l’adaptation aux technologies et surtout l’orientation que doit prendre le projet.

1. **Le REAC**



1. Le maquettage du site s’est fait rapidement, suite à la conception du site, il faut dire que dès le départ j’avais opté pour une interface plutôt épurée et classique pour permettre à l’utilisateur de se repérer facilement dans sa navigation sur le site, sachant que l’espace de ce dernier fonctionne comme une Dashboard à partir de laquelle il peut gérer toutes les fonctionnalités qui lui sont offertes sur le site.



1. Le développement du site est en majeure partie statique et responsive, l’utilisation de **Bootstrap** permet, en majeure partie l’adaptabilité des divers éléments affichés dans le front du site, cependant l’utilisation de **media queries** n’a pas été exclu afin de permettre l’adaptation sur téléphone portable. Pour le moment l’adaptation sur les nouveaux portables fold est en cours de développement.





1. Une partie du contenu du site est quant à lui géré dynamiquement par mon **PHP**, je pense particulièrement aux informations de l’utilisateur qui sont récupéré dans le backend afin que chaque utilisateur ait la sensation d’avoir son propre espace, il voit son contenu et les informations qu’il nous a envoyé lors de son inscription. L’affichage des posts/articles des utilisateurs est lui aussi géré dynamiquement, une fois qu’un utilisateur l’a créé, il est affiché sur la page principale de tous les utilisateurs.



1. Afin de réaliser ce site, la base de données était cruciale, alors elle a été développée avant toutes fonctionnalités. En me basant sur le **Modèle conceptuel de données**, le **Modèle logique de donnée** et **MySQL**, la base de données est adaptée à l’accueil du contenu qui fera apparition sur le site, que ce soit un nouvel utilisateur, du contenu qu’il publiera ou même plus tard des salons qu’il ouvrira et pourra personnaliser comme bon lui semble.



1. Afin de relier notre base de données et notre backend, le modèle de la connexion à la **BDD** et les modèles de chaque table ont été réalisé avec **PHP**. Dans chacun des modèles, plusieurs fonctions ont été développé afin de pouvoir préparer chaque requête qui seront envoyées à la base de données et récupérées par notre backend.
2. En plus des modèles, les contrôleurs de nos fonctionnalités ont été développé avec **PHP**, pour terminer la boucle et entretenir cette communication du front à la base de données en passant par le backend.
3. **Cahier des charges**

Je me retrouve alors missionné par le grand du streaming Twitch qui décide de mettre en place un nouveau concept, une plateforme sur laquelle nous pourrons voir l’émergence de nouvelles communautés, l’idée est de donner une notion supplémentaire d’appartenance aux individus qui suivent régulièrement le contenu diffusé sur Twitch.

On aura la possibilité de partager, visionner avec nos amis, rencontrer des gens qui partagent les mêmes centres d’intérêt et de ce fait créer une sorte d’écosystème digital en dehors du site d’origine.

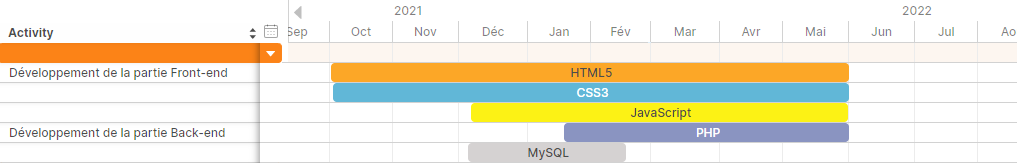
Pour la réalisation de ce projet, le contenu du cahier des charges regroupe l’intégralité de l’étude des opportunités, des risques, forces et faiblesses du projet.

Quand on y jette un œil de plus près on peut voir que Twitch ne possède pas non plus le monopole du marché du streaming.

Sachant que Facebook (Meta), un des autres grands du streaming, a commencé à se lancer dans la course du Meta verse, il faut prendre en compte l’ensemble de la concurrence et jauger la menace qu’elle inclut.

On y retrouve aussi l’étude du marché comprenant notre public cible, surtout centré sur une population en majorité jeune et masculine qui s’oriente vers les jeux vidéo. Cependant j’ai quand même choisi de mettre le cap sur une approche plus généraliste en termes de population. Savoir où l’on va c’est bien, mais il faut aussi voir les opportunités que l’on peut trouver lorsque l’on fait une proposition différente.

Afin d’aller un peu plus loin, le GANTT de ce projet a été établi en même temps que le cahier des charges.



La partie front avec le **HTML**, le **CSS** et le **JavaScript** devait alors débuter dès le commencement du projet pour être approfondie et améliorée au fur et à mesure que de nouvelles fonctionnalités feraient leur apparition.

La **base de données** quant à elle nécessitait plus de réflexion et de ressources que prévue et poursuit son évolution ponctuellement.

Le **PHP** se retrouve être le moteur de mes fonctionnalités actuellement et nécessite, depuis son intégration, une amélioration et un travail continu pour permettre au site d’être la plateforme que l’on souhaite.

1. **Spécifications fonctionnelles**

L’étape qui a suivi notre cahier des charges, le moment où j’ai commencé à mettre en place le fonctionnement du site, les acteurs qui interviendront dessus et surtout la logique des fonctionnalités.

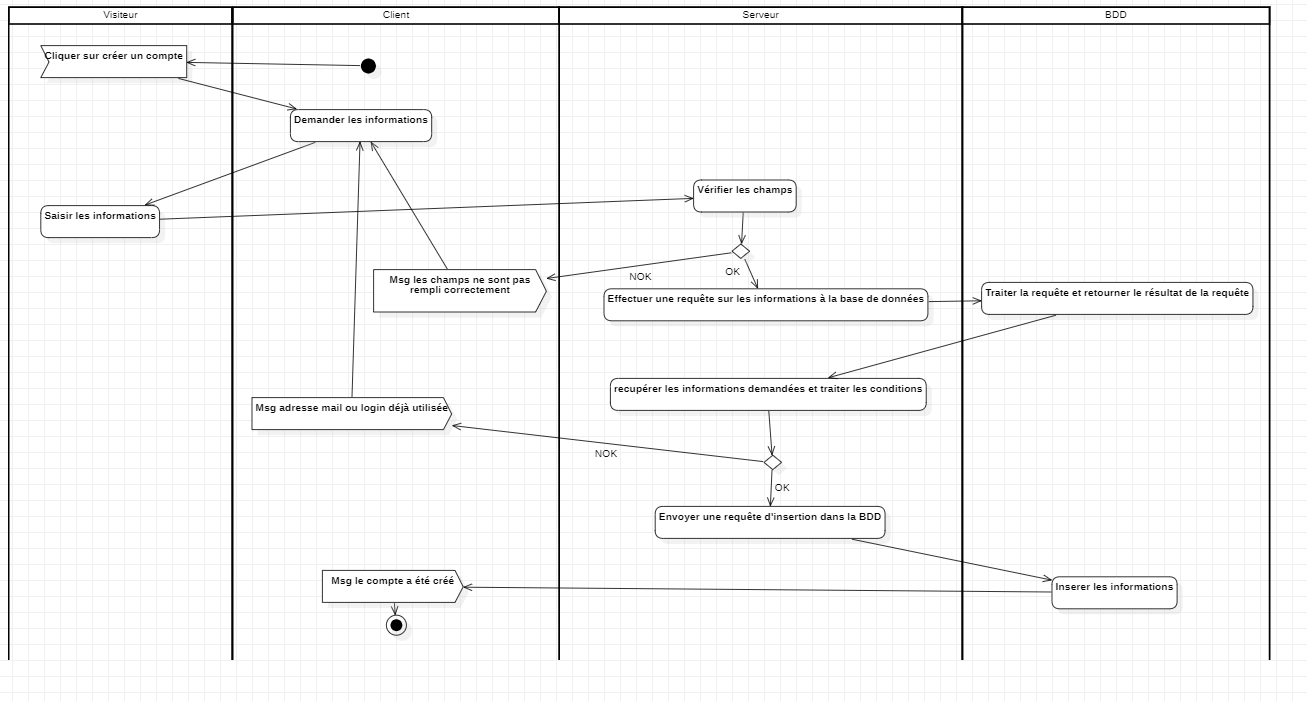
* 1. **Use Case**

Le **Use Case** comme on peut le voir va venir mettre en exergue les fonctionnalités qui seront à la disposition des différents acteurs, en plus des fonctionnalités propres à chacun, on peut voir qu’il y a **un héritage** qui se produit entre tous. Au final seul l’administrateur va posséder des actions qui lui seront propre. A contrario, le visiteur lui sera très limité en termes de droit sur la plateforme. Il devra alors créer un compte afin de pouvoir être élevé au rang d’utilisateur et bénéficier pleinement du site.

* 1. **Diagramme d’activité**

Le Diagramme d’activité va nous permettre de mettre en place le raisonnement derrière chaque fonctionnalité. Il sera le fil directeur dans la réflexion derrière le développement.

* + - **La création d’un compte utilisateur**



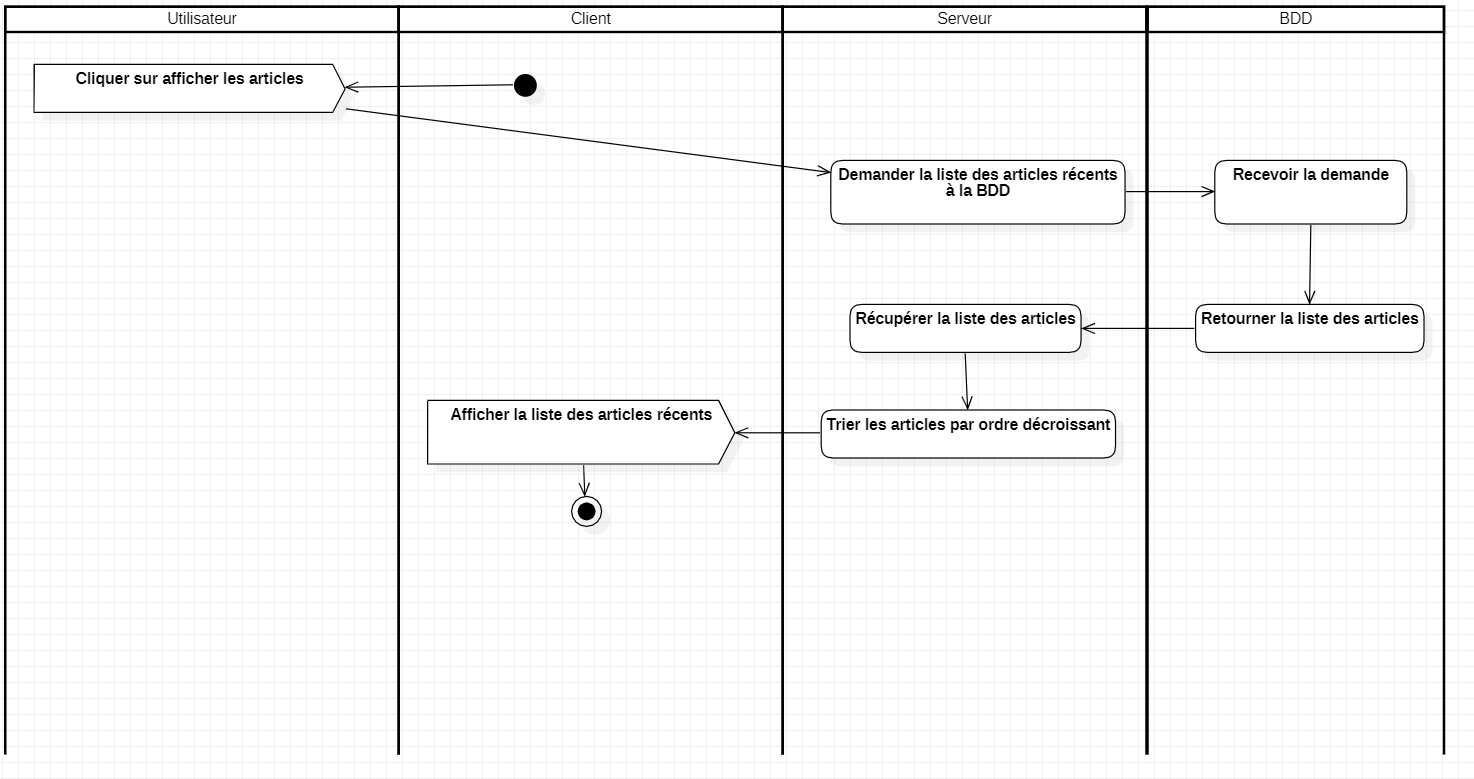
Dans la conception du diagramme d’activité de la création d’un **utilisateur**.

On peut noter qu’il est requis une action afin de lui ouvrir l’accès à la création du compte. Une fois l’action réalisée, on entame le processus de création. Il devra renseigner les différents champs nécessaires et obligatoires à l’insertion de son compte. Durant le processus, des vérifications seront effectuées afin de déterminer si certaines de ses informations sont ou non déjà présentes dans la base de données.

Le but étant de ne pas avoir de doublon sur les informations sensibles pouvant nuire à l’intégrité d’un compte.

Une fois toutes ces actions effectuées. Le visiteur possède un compte et est élevé au rang d’**utilisateur**.

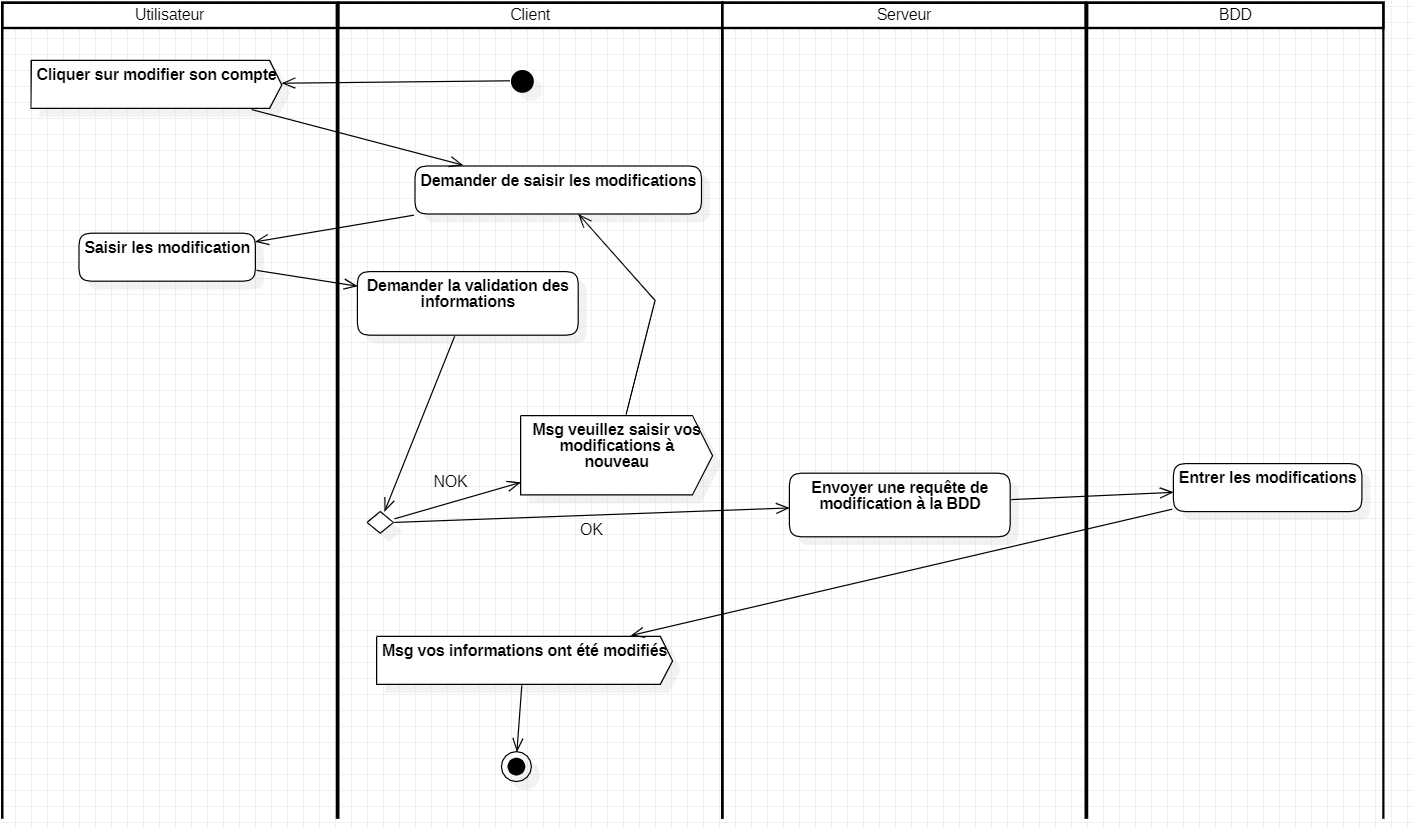
* + - **L’affichage des articles**



L’affichage des articles lui devait suivre un principe simple, il était souhaité que ces articles que l’on a appelé communément des « **posts** » soient accessibles de tous. En reprenant cette idée de réseau social, il était intéressant que les différents membres du site puissent partager ensembles et, par la suite réagir à divers contenus provenant de tous les utilisateurs.

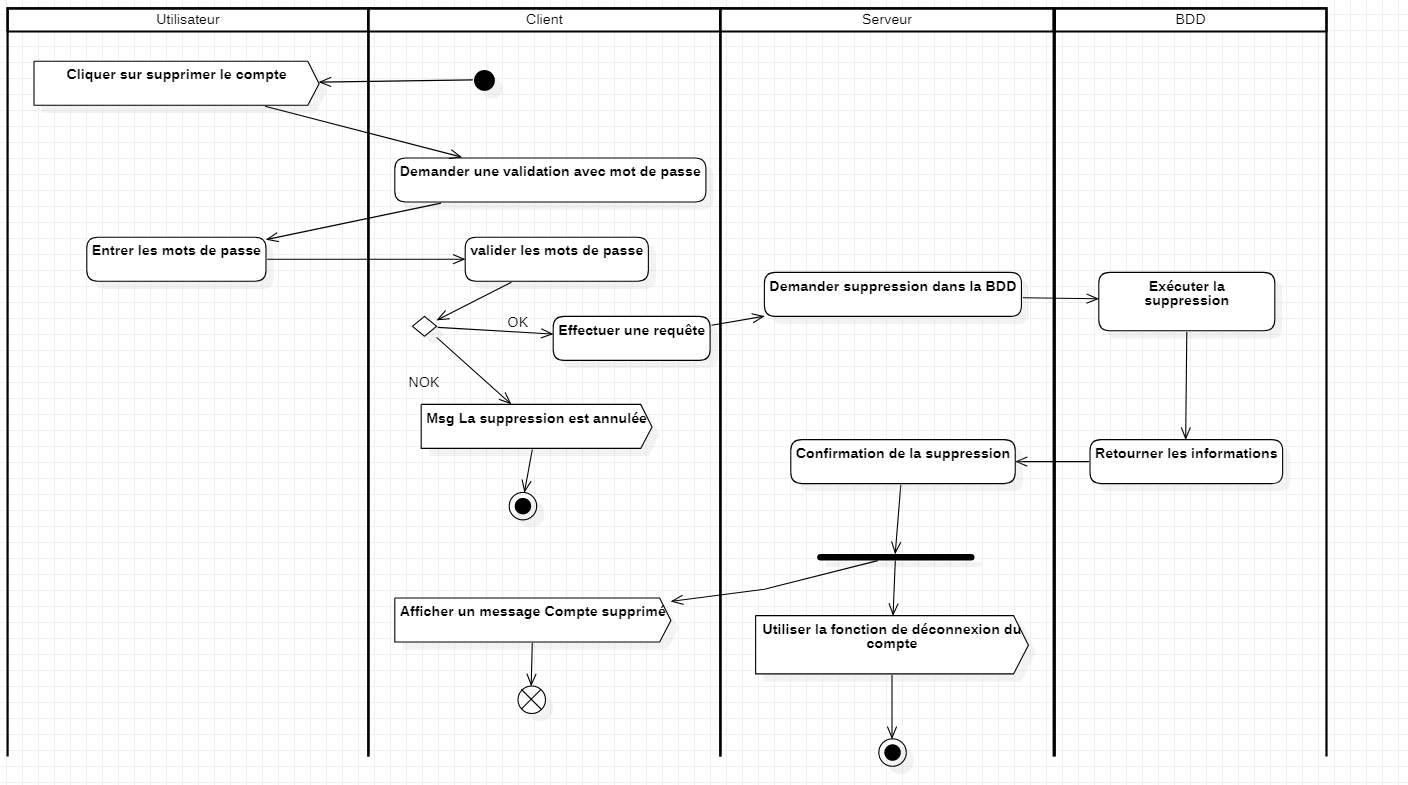
Une fois l’utilisateur sur sa page principale, il peut accéder à tous les articles qui sont classés par ordre chronologique du plus récent au plus ancien. On demande donc à notre backend d’effectuer une requête pour récupérer les articles que l’on trie et retourne à notre front.

* + - **La modification du compte**



Afin de pouvoir modifier son compte, l’utilisateur va devoir accéder à une page où il peut trouver toute ses informations. Et, à partir de là lui est offert la possibilité de modifier ses informations. Dans un rendu général, la modification de compte se déroule de la sorte. Des informations concernant la modification lui seront demandées, par la suite l’utilisateur devra entrer ses nouvelles informations et les anciennes dans un processus de sécurisation de l’opération. Le système procèdera alors la mise à jour des nouvelles informations de l’utilisateur. Et l’utilisateur sera notifié du succès de l’opération.

* + - **La suppression d’un compte**



L’utilisateur a toujours la possibilité de supprimer son compte s’il le souhaite. Nous nous sommes tournés vers le principe de suppression complète du compte.

Si l’utilisateur fait la démarche de cliquer sur la suppression de son compte. Il sera alors envoyé vers une fenêtre où il devra entrer son mot de passe deux fois afin de vérifier qu’il est sûr de sa décision.

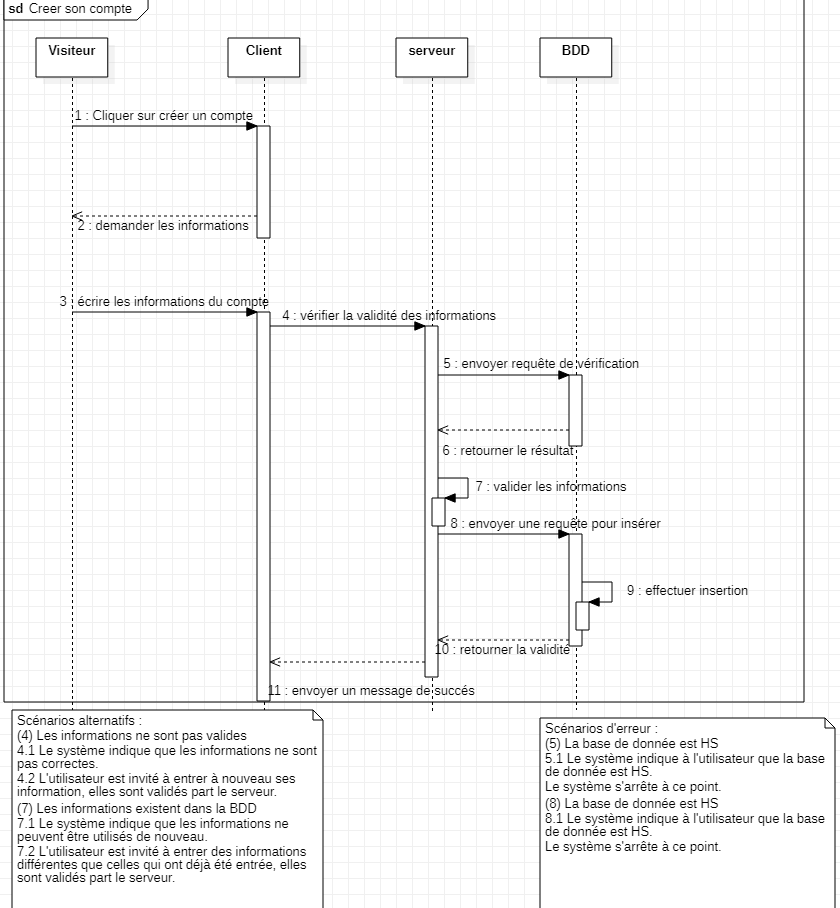
Une fois ceci fait notre back va vérifier si le mot de passe correspond bien à cet utilisateur et va alors valider ou non l’opération.

Si l’opération est bien valide, la requête de suppression est envoyée à la base de données qui va supprimer la ligne correspondante à cet utilisateur. L’utilisateur est alors supprimé et devient à nouveau un visiteur avec les droits qui en découle.

* 1. **Diagramme de séquence**

Alors qu’il est très proche du diagramme d’activité, le diagramme de séquence quant à lui possède une particularité et pas des moindres, il va nous servir de feuille de route pour évaluer au mieux les scénarios alternatifs et les scénarios d’erreurs. Encore une fois, un diagramme nous permettant d’améliorer le développement de chaque fonctionnalité en prenant compte des difficultés pouvant être rencontrés lors de leur utilisation.

* + - **La création d’un compte**



Je vais accorder une importance particulière aux nouveautés qu’apporte ce diagramme de séquence. On y voit donc l’apparition des scénarios alternatifs et d’erreurs. Du côté des scénarios alternatifs, on va relever l’invalidation des informations. Ce scénario-là peut survenir de diverses façons.

Cela peut correspondre à des caractères employés qui sont invalides, un nombre de caractères ne correspondant à celui que nous allons imposer, des caractères qui ne sont pas employables, des champs manquants.

Dans un deuxième temps, il s’agit d’un scénario qui survient lors de la vérification côté serveur, la base de données nous indique qu’une information du champ existe d’ores et déjà dans la base de données ce qui entraine un retour à l’étape d’inscription des champs côté client.

En termes de scénario d’erreur on en a relevé un seul. Il s’agit de la base de données qui est injoignable, cet évènement peut survenir deux fois lors de la création d’un compte utilisateur.

* + - **L’affichage des articles**

Lors de la réalisation de ce diagramme, un seul scénario alternatif m’est venu à l’esprit.



L’absence d’article et l’impossibilité d’afficher un article sur l’écran de l’utilisateur. Afin de pallier à cet éventualité, l’idée d’émettre un message qui prendra la place des articles en attendant qu’un premier post soit publié.

Ici on retrouve encore une fois le scénario d’erreur qui survient lors de la requête qui doit normalement se charger de récupérer et trier tous les articles en base de données. Sauf que cette dernière n’est pas accessible.

* + - **La modification d’un compte**



Lors de la modification du compte on rencontre un scénario alternatif qui est que les informations entrées par l’utilisateur ne correspondent pas.

Il s’agit ici de vérifier que les informations du compte ne sont pas erronées. C’est une sécurité supplémentaire pour s’assurer au maximum qu’une tierce personne ne change pas les informations du compte à l’insu de son utilisateur.

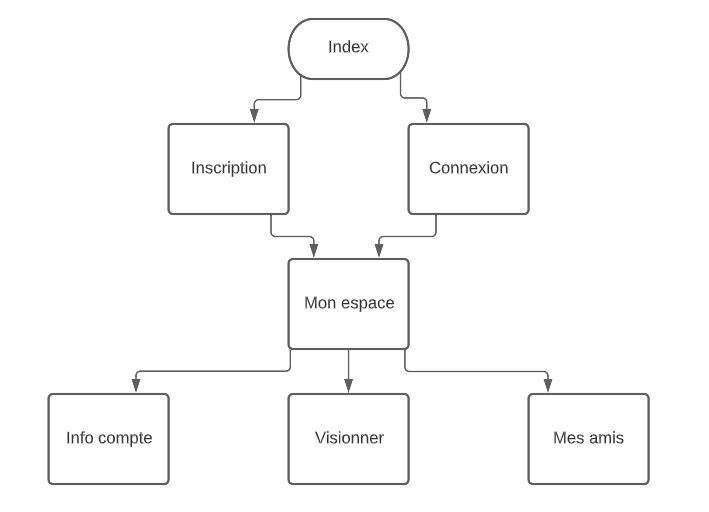
Lors de la mise à jour des informations de l’utilisateur, on a relevé un possible scénario d’erreur nous renvoyant à une base de données qui est inaccessible et met fin au système.

* 1. **Maquettage**

Tout d’abord la présentation de l’organigramme du site. Dans sa conception nous partons d’un visiteur qui arrive sur le site.

Il se retrouve sur la première page qui nous sert à filtrer qui est un utilisateur et qui est un visiteur.

Nous offrons la possibilité de se connecter à l’utilisateur, ou, le cas échéant possibilité de créer un compte si l’on est visiteur.



Dans les deux cas, les acteurs seront une fois la connexion/inscription terminée redirigés vers leur espace à partir duquel ils auront la possibilité d’accéder à toutes les fonctionnalités du site.

S’en suivent les maquettes m’ayant permis la réalisation de l’affichage en Front.

* + - **Les zonings et wireframes**



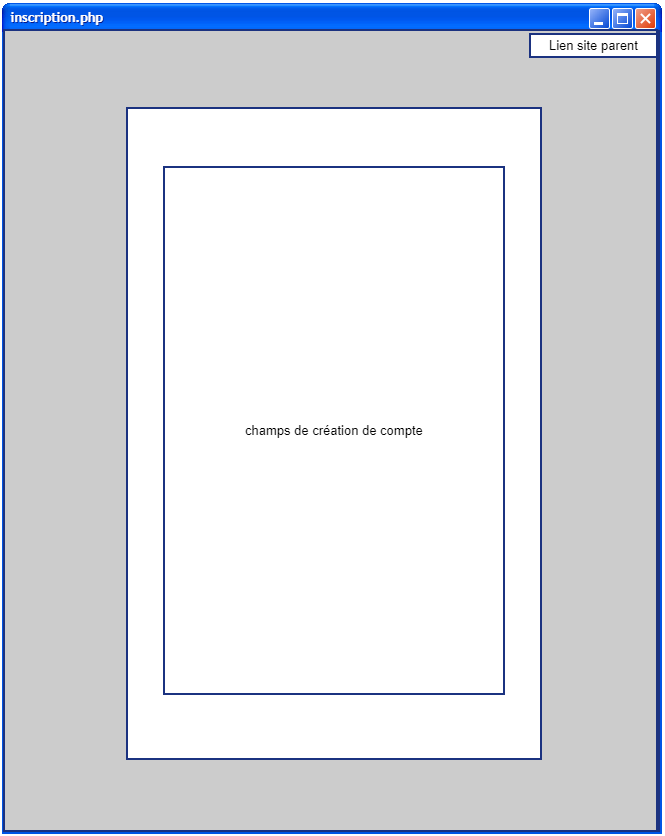


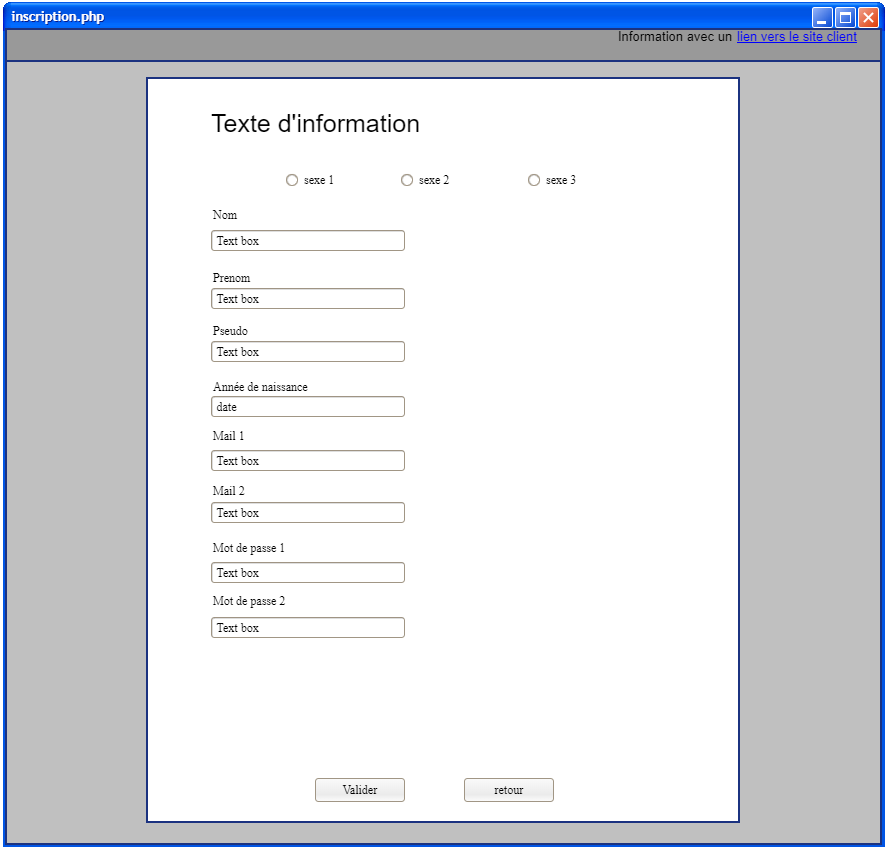
La page d’accueil arbore une maquette très simple. Peu d’informations doivent y figurer, pour seule directive, qui devra par la suite se retrouver sur les autres pages du site, un lien vers **Twitch**.

Ensuite, pour ce qui est de la mise en forme de la page, il va falloir conserver de l’espace pour obtenir un design minimaliste.

Seuls deux boutons figurent sur cette page afin de permettre à l’utilisateur de savoir où il doit se tourner pour progresser sur le site.

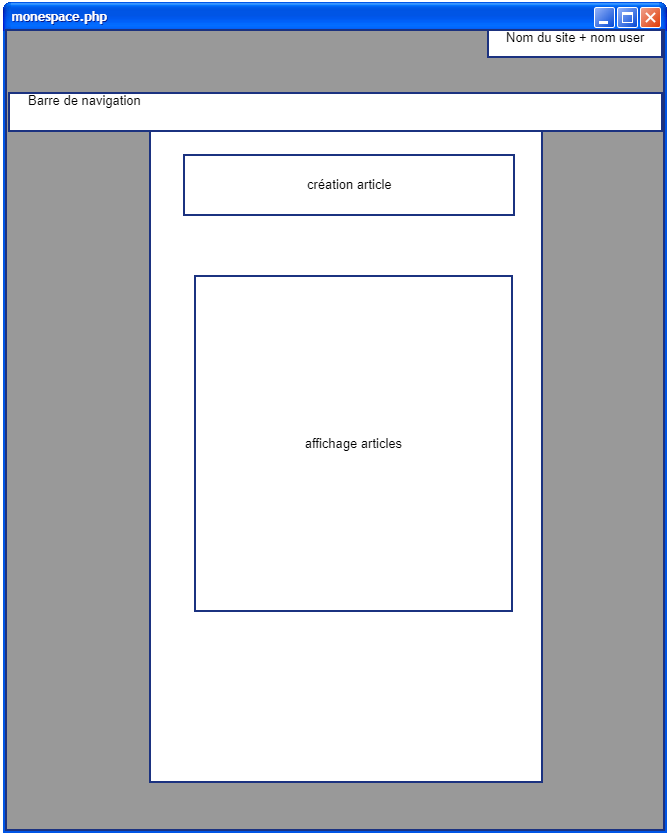
Maintenant, on va suivre le cheminement d’un visiteur qui décide de créer un compte :



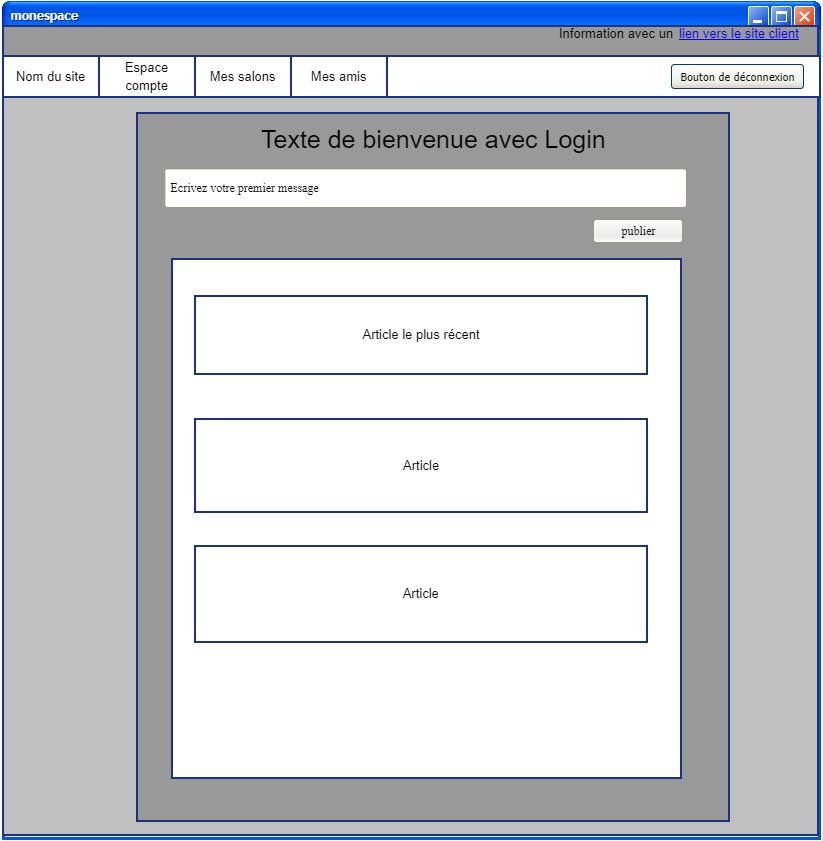


Notre visiteur cherchant à s’inscrire arrive donc sur cette page où il est invité à entrer ses informations personnelles. Pour réaliser la création de son compte.

Une fois le compte finalisé, notre anciennement **visiteur** est connecté à son compte **utilisateur** qu’il vient tout juste de créer.

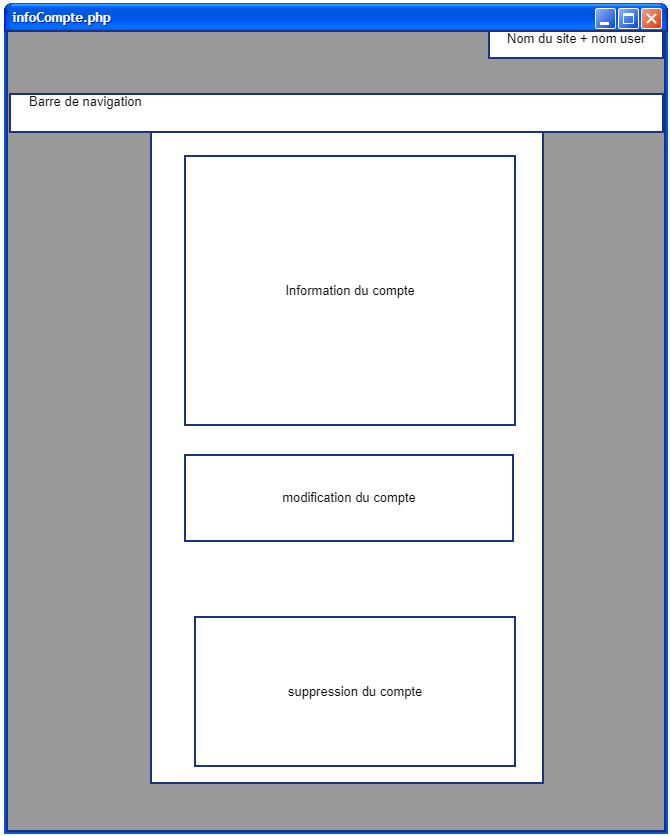


Une fois connecté notre nouvel utilisateur se retrouve sur son espace d’où il a accès à toutes les fonctionnalités qui lui sont offertes par le site **Verse** au travers du menu de navigation. En plus de cela, sur cette page il a la possibilité d’écrire un article afin de partager avec les autres utilisateurs.



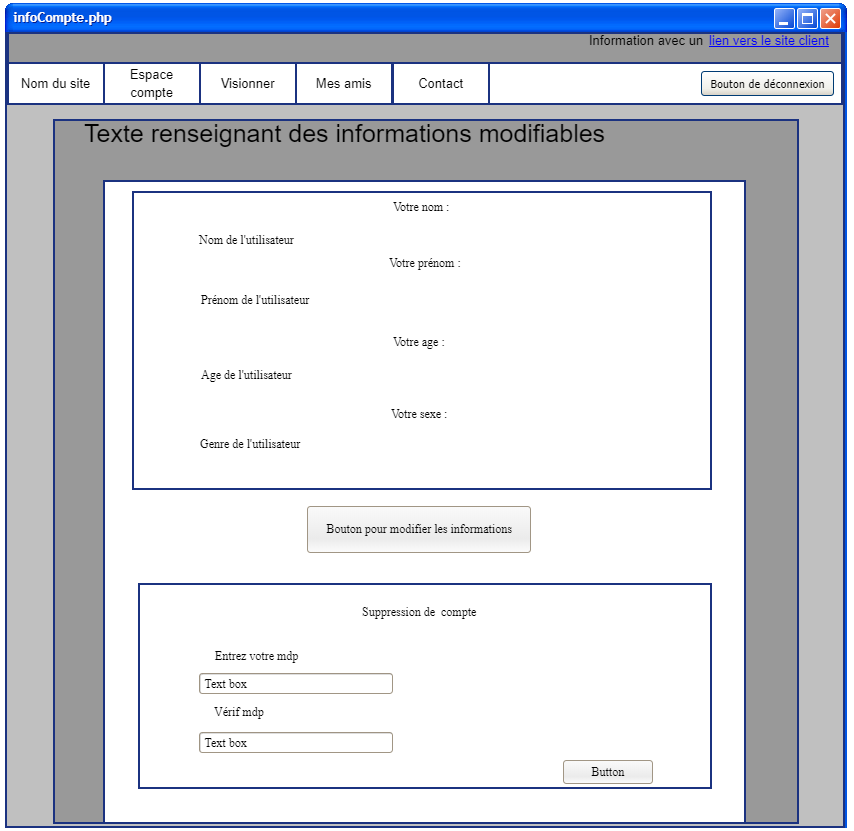
Mais c’est aussi l’endroit où il peut consulter les derniers articles posté par les autres utilisateurs.

Vient ensuite le moment où l’utilisateur désire voir ou modifier ses informations, ou même, décider de supprimer son compte :



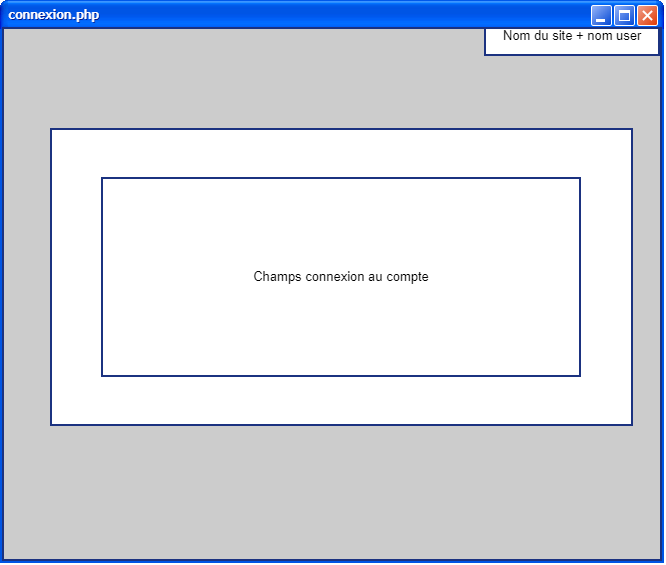
L’utilisateur se retrouve dans l’espace où il peut accéder aux informations qu’il a entré.

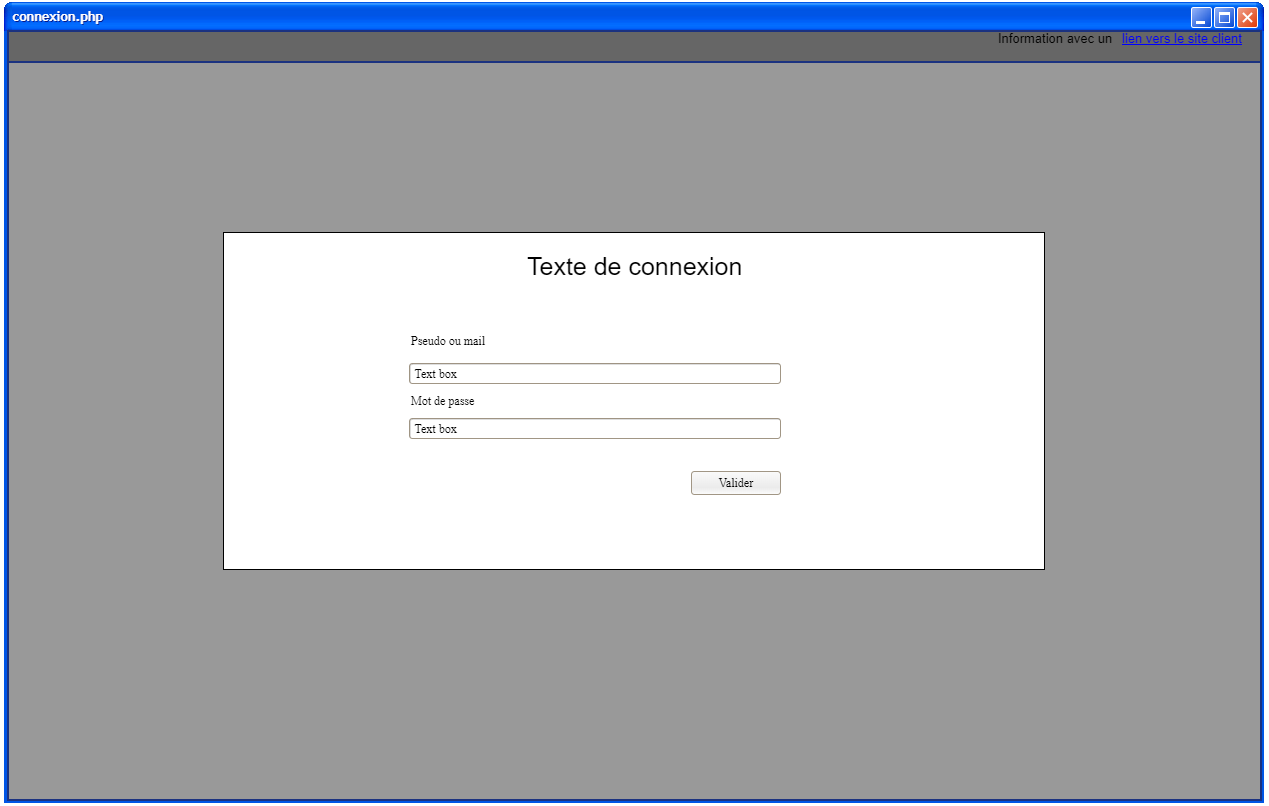
En plus de ça, il a la possibilité d’y modifier son login ainsi que son mot de passe en cliquant sur le bouton de modification qui lui ouvrira une fenêtre modale qui lui demandera d’entrer ses informations et de les valider, qu’il souhaite modifier son login ou son mot de passe.



Et dans des cas extrêmes, il peut décider de supprimer son compte. Il lui sera demandé d’y entrer son mot de passe et de le vérifier. Ceci fait, son compte sera alors définitivement supprimé et il sera renvoyé vers la page d’accueil.

Imaginons un utilisateur, donc une personne qui possède déjà un compte, décide de se connecter au site. Sur la page d’accueil il va donc cliquer sur se connecter. Ce qui l’enverra sur la page de connexion.





L’utilisateur accède donc à la page connexion où on lui demande d’entrer ses informations. Une fois l’opération effectuée, il peut cliquer sur valider.

Ceci fait, il est connecté à son compte et se retrouve sur la page principale de son espace.

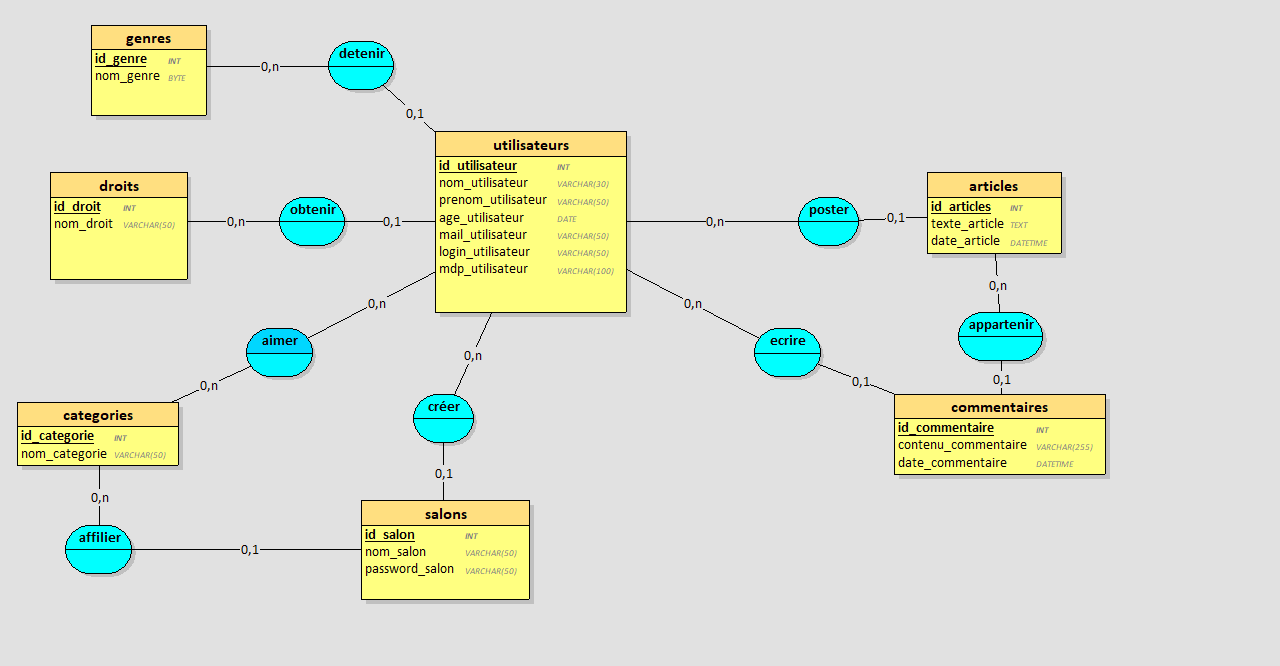
1. **Conception**

* **MCD, Modèle conceptuel de données**

Pour la conception du site internet, j’ai réalisé un MCD qui m’a donné la marche à suivre pour la réalisation de la base de données.

Le MCD que je vais vous présenter n’est qu’une partie du MCD final.

Cela permettra plus de lisibilité sur la conception des fonctionnalités qui sont déjà en place sur le site.



Notre MCD met donc en évidence le rôle central de l’utilisateur autour duquel toutes les autres entités vont graviter.

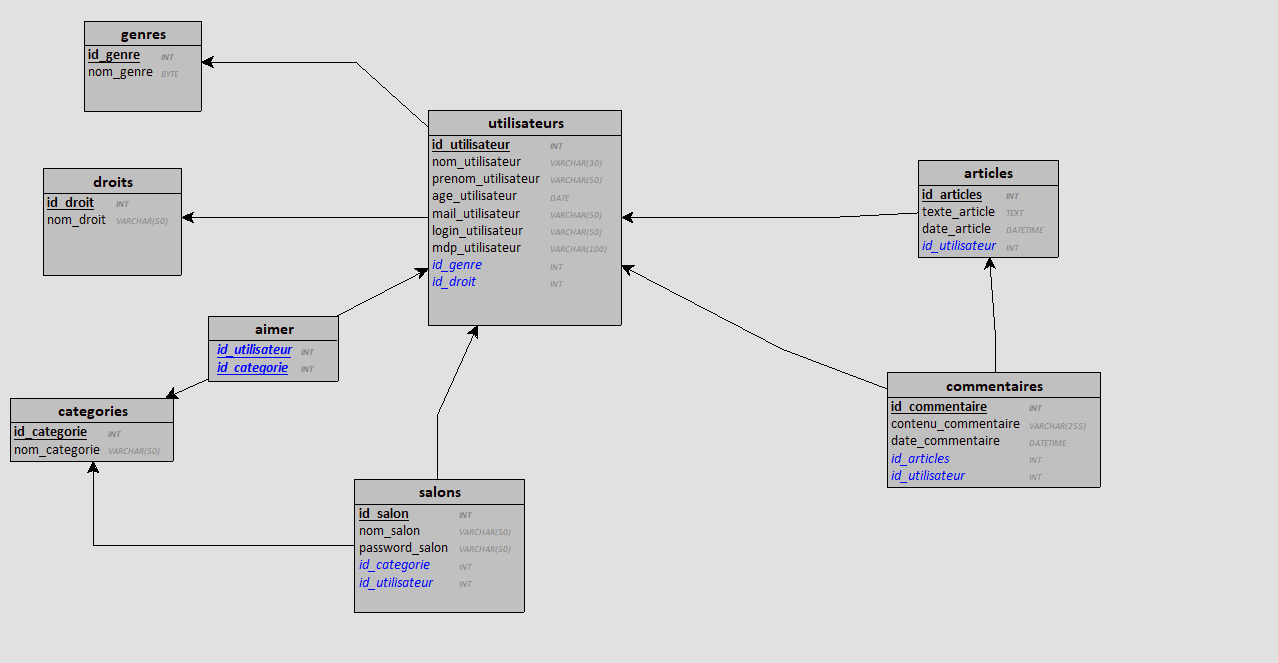
Nous n’avons qu’une seule association sur ce modèle, entre Utilisateurs et Catégories, c’est ici le seul endroit où l’on peut retrouver des relations multiples entre les deux entités.

L’**utilisateur** peut **aimer** plusieurs **catégories**, et les **catégories** peuvent **être aimé** de plusieurs **utilisateurs**.

Pour aller plus loin, nous allons observer le MLD pour pouvoir observer les différentes liaisons entre les entités et voir comment les clés étrangères ont été distribués entre les différents acteurs de ces modèles.

* **MLD, Modèle Logique des données**

Grace au MLD on peut observer les relations entre les entités. On voit immédiatement que la table **utilisateurs** récupère les clés étrangères de la table **genres** et **droits**. On peut observer qu’entre la table **catégories** et **utilisateurs** émerge l’association **aimer** qui récupère en clés primaires l’id\_utilisateur et l’id\_categorie.



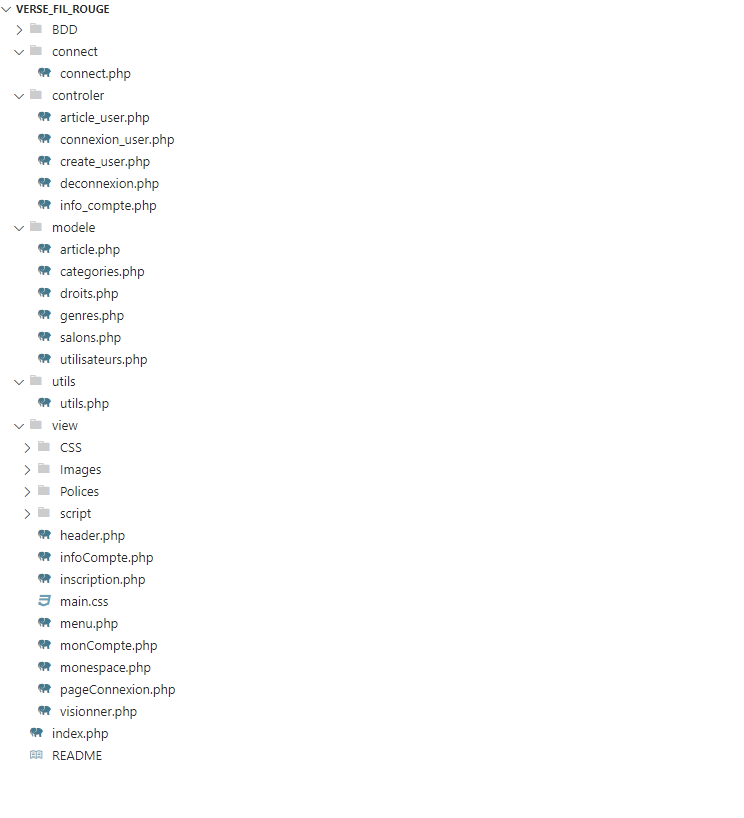
Les **salons** ne pouvant être créé que par un seul **utilisateur** vont alors récupérer son id en clé étrangère et selon la conception actuelle, les salons ne pourront avoir qu’une seule **catégorie** ce pourquoi elle récupère en clé étrangère id\_categorie.

1. **Arborescence des fichiers**

Pour développer le site internet de Verse, nous avons opté pour un modèle **MVC**, le bien nommé, **Modèle, Vue et Contrôleur** afin d’obtenir une forme d’ordre à l’intérieur de nos fichiers. Et permettre une maintenance bien plus facile et adaptée au développement continu de cette plateforme.

A l’intérieur de notre **Modèle** l’ensemble des classes qui nous permettront d’établir une relation avec la base de données. C’est ici que je vais préparer les requêtes afin de pouvoir les utiliser ultérieurement. C’est ici que nous trouvons la logique en lien avec les données.

Ensuite le **Contrôleur** qui va contenir les différentes actions, feature de mon backend **PHP.** C’est ici que nous traitons les informations et que nous utilisons les **Modèles des classes**.

Et enfin, la **Vue,** qui sera là où seront stockés les fichiers qui représentent la partie visible par l’utilisateur.

Un des points fort du **MVC** c’est sa lisibilité. Et pour moi, il était important, dans la conception, de pouvoir aussi miser sur l’avenir du site et permettre faciliter le travail collaboratif.

Afin de rendre les fichiers plus lisibles et bien plus reconnaissables, la refactorisation des fichiers était un enjeu important.

1. **Outils techniques Utilisés**
   1. **Le Frontend**

Pour la partie architecture de la page le squelette et la mise en forme des éléments trois langages entre dans la danse accompagnée d’un Framework.

Tout d’abord on a **HTML5**, qui vient s’occuper du squelette de la page. Chaque élément concentré sous son architecture accompagné de Bootstrap qui vient permettre une première mise en forme de notre page grâce à ses librairies.

Le coup de pinceau de **CSS3** qui vient améliorer le rendu visuel, la mise en page de notre site, lui aussi accompagné de **Bootstrap**.

Et enfin **Javascript** qui fait une petite apparition avec l’affichage d’une fenêtre modale.

* 1. **Le Backend**

Pour la base de données nous avons **PhpMyAdmin**, notre application qui nous a accompagné dans la gestion de cette dernière. Le langage qui nous a servi dans la conception de la **BDD** est **MySQL**. (Voir annexe\_1)

Pour ce qui est des fonctionnalités présentes sur le site, la gestion des utilisateurs, l’affichage dynamique et la communication entre le client et la base de données, **PHP**.

* 1. **Le versionnage**

Sans oublier, bien sûr, **GitHub** qui m’a accompagné tout au long du projet pour les sauvegardes de toutes les étapes de la conception au développement du projet en lui-même.

1. **Fonctionnalités**
   1. **L’inscription de notre visiteur**

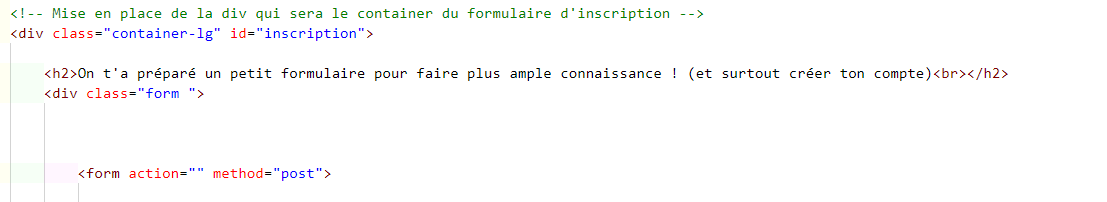
Il est maintenant temps de présenter ce qui représente le cœur de notre projet, le code. Encore un peu de patience, dans un premier temps, je vais présenter ce que le visiteur va voir lorsqu’il va cliquer sur le bouton rejoindre le site :

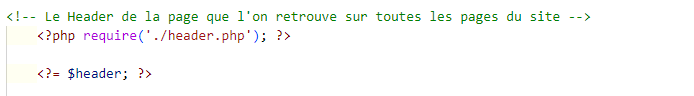
.

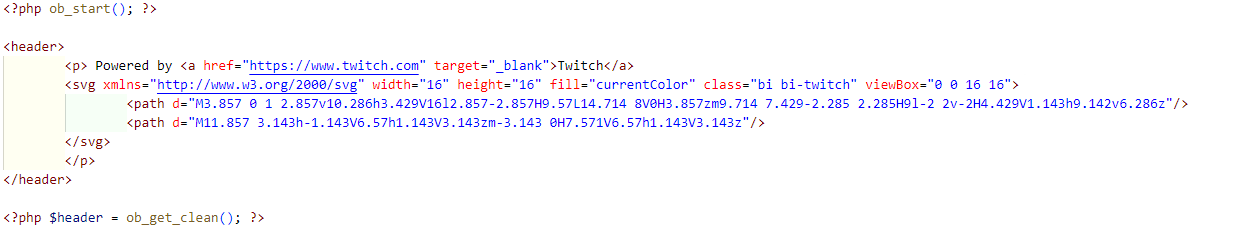


Pour le **HTML**, on a tout d’abord un header que l’on vient importer via du **PHP**

On retrouve la fonction native de Php ob\_start() et ob\_get\_clean() qui permettent l’enregistrement du html que l’on entre à l’intérieur de ses deux dernières.





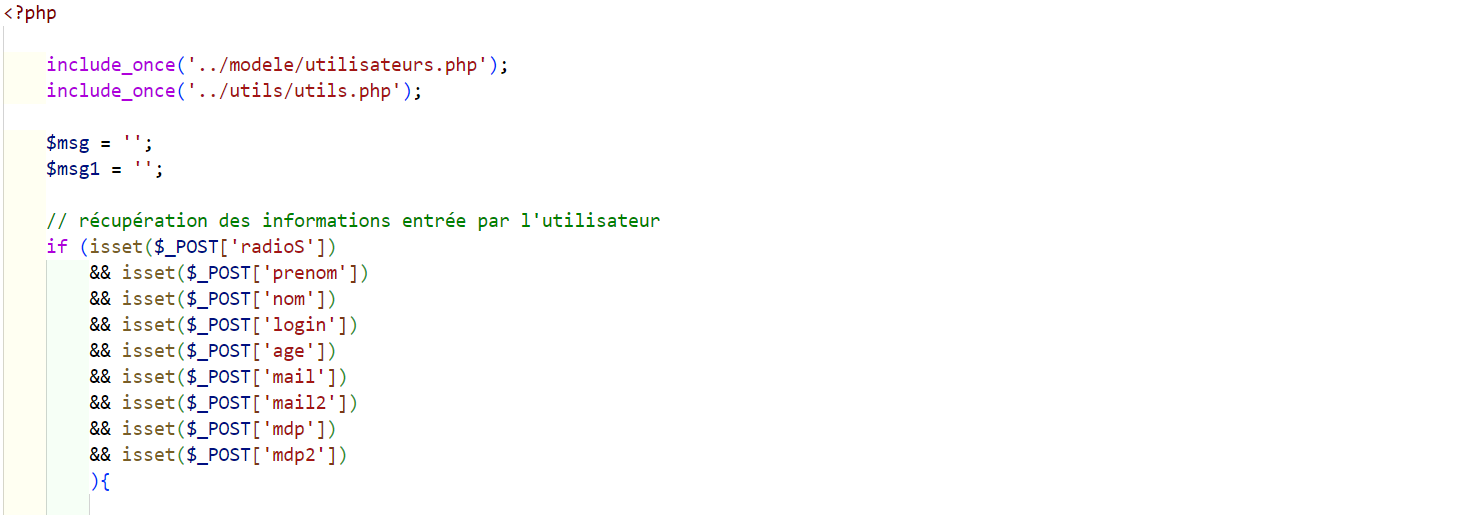


Ici on voit la balise <div> qui va englober l’intégralité du formulaire, c’est en utilisant une classe Bootstrap que je commence la mise en forme de l’ensemble du document, la classe container-lg étant une classe propre au **Framework**.

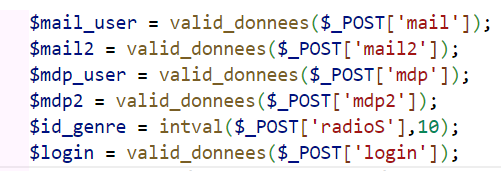
Le formulaire quant à lui, à des fins de protection des informations qui seront entrées, va utiliser la super globale **$\_POST** pour récupérer les informations qui seront entrées

Une fois que les champs ont été remplis et que le visiteur appuie sur le bouton valider. Que se passe-t-il ? Eh bien c’est ici que le **PHP** va entrer en jeu.

On va directement dans notre fichier create\_user.php, le contrôleur en charge de l’inscription alias de la création de notre nouvel utilisateur.

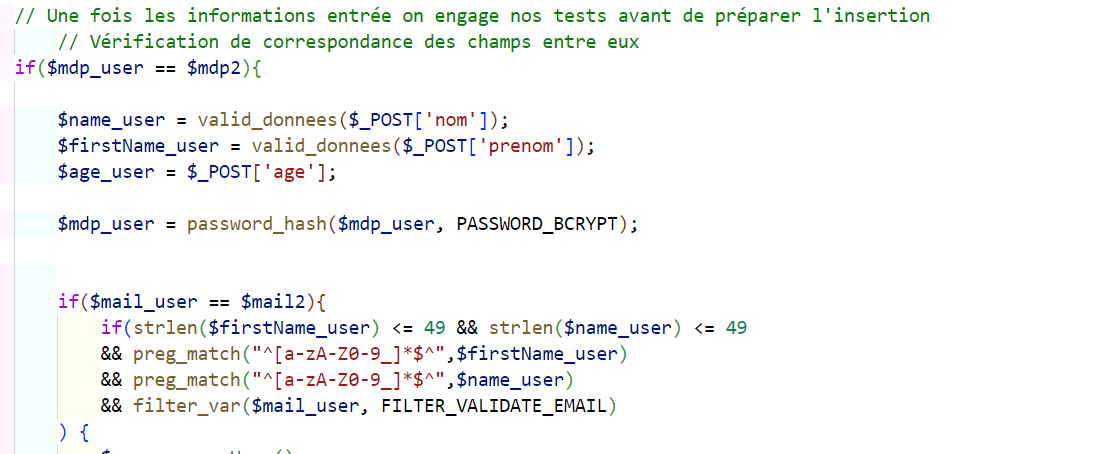


Je vais alors vérifier que ma super globale **$\_POST** a bien récupéré les champs lors du submit de notre bouton de validation.



On va alors appliquer sur les champs récupérés une fonction valid\_donnees () afin de se protéger des injections XML.

Ceci fait on va enchainer avec nos premières vérifications afin de s’assurer que les champs de mail/mot de passe et vérification du mail/mot de passe soient identiques. Une fois les mots de passe validés, il est directement haché.



Pour mieux comprendre la suite on va d’abord jeter un œil à notre modèle utilisateurs.php.

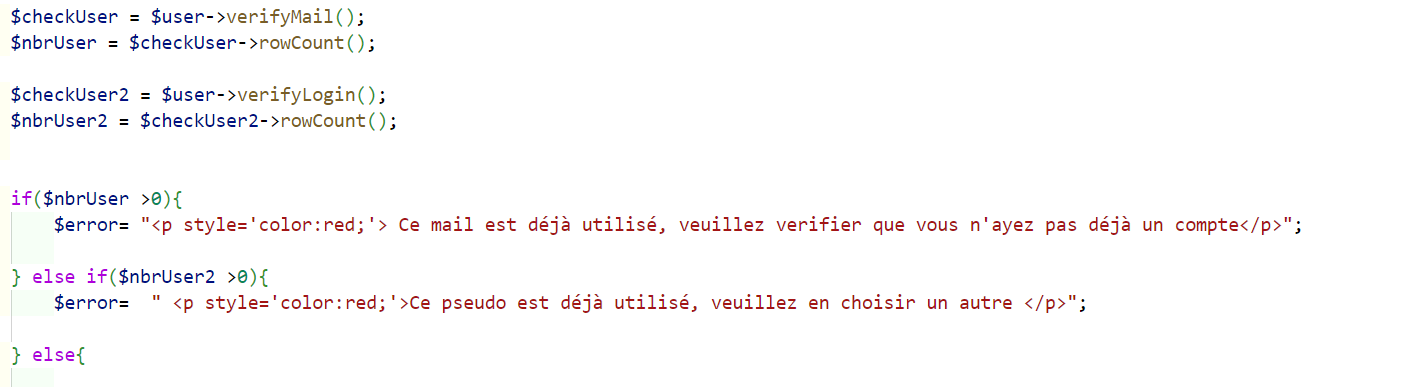
Voici la classe utilisateur qui va nous permettre la création de l’utilisateur, elle est directement connectée à la base de données par notre constructeur qui va instancier une nouvelle classe BDD à l’intérieur du constructeur de notre classe User.

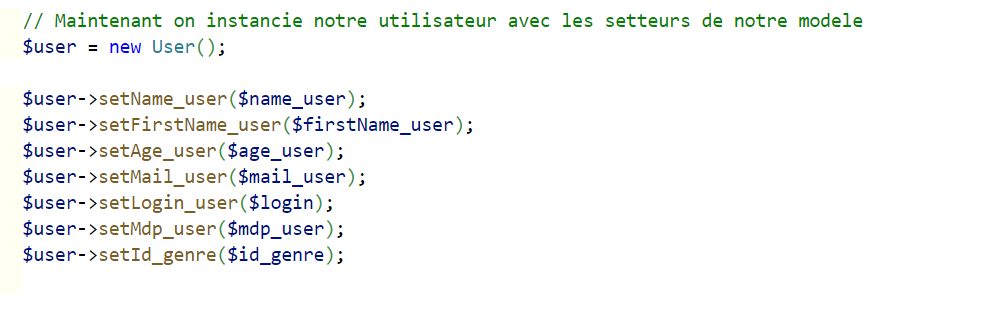


On va alors retrouver dans cette classe l’intégralité des éléments de la table utilisateurs.

Pour en revenir à notre contrôleur, on a récupéré l’intégralité des informations inscrites dans nos champs, on a validé les champs doubles, on a haché le mot de passe. Il nous reste plus qu’à entamer la création de notre utilisateur.

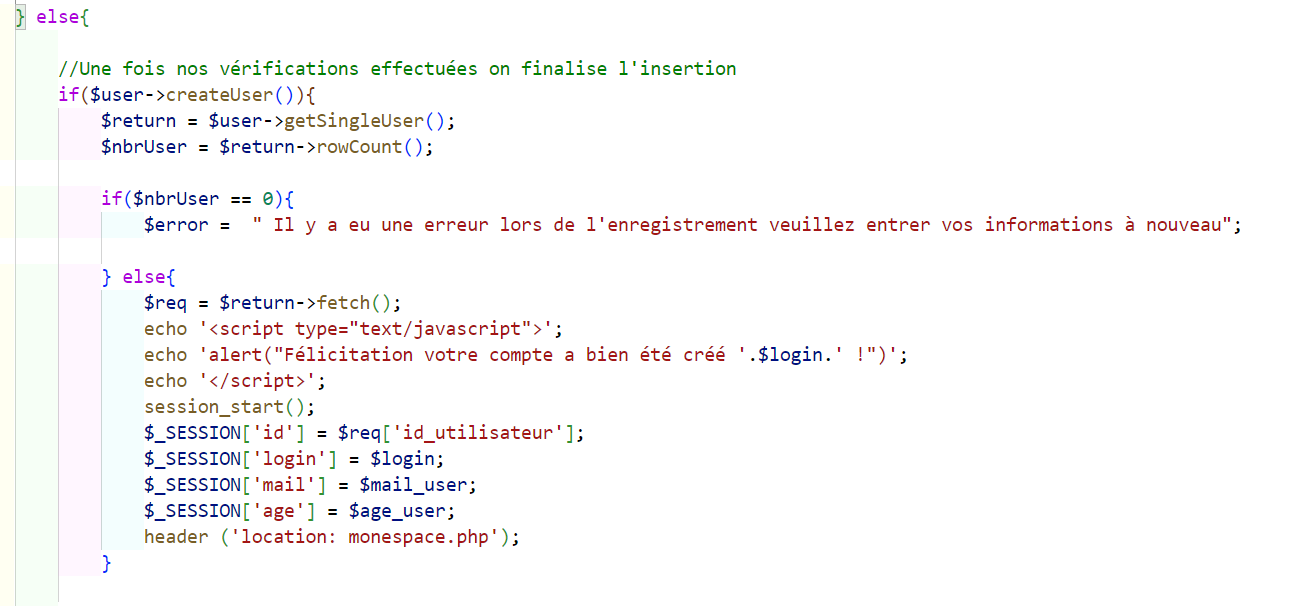
Une fois ceci fait nous allons pouvoir vérifier si le mail ou le login de l’utilisateur sont déjà utilisés sur un autre compte utilisateur. Ce sont les seules informations uniques et propre à chaque utilisateur.





Ici nous allons vérifier si à partir du mail et du login utilisateur nous trouvons des résultats identiques dans notre base de données avec un message informant le visiteur quel élément, le cas échéant est déjà inscrit dans nos données.

Si les informations entrées par notre visiteur sont uniques, alors nous pouvons l’inscrire dans notre base de données et l’accueillir chez **Verse.**



Si tout est bon et que l’insertion se fait correctement, l’utilisateur est créé. En cas d’erreur un message est affiché pour informer l’utilisateur que son inscription est interrompue suite à une erreur.

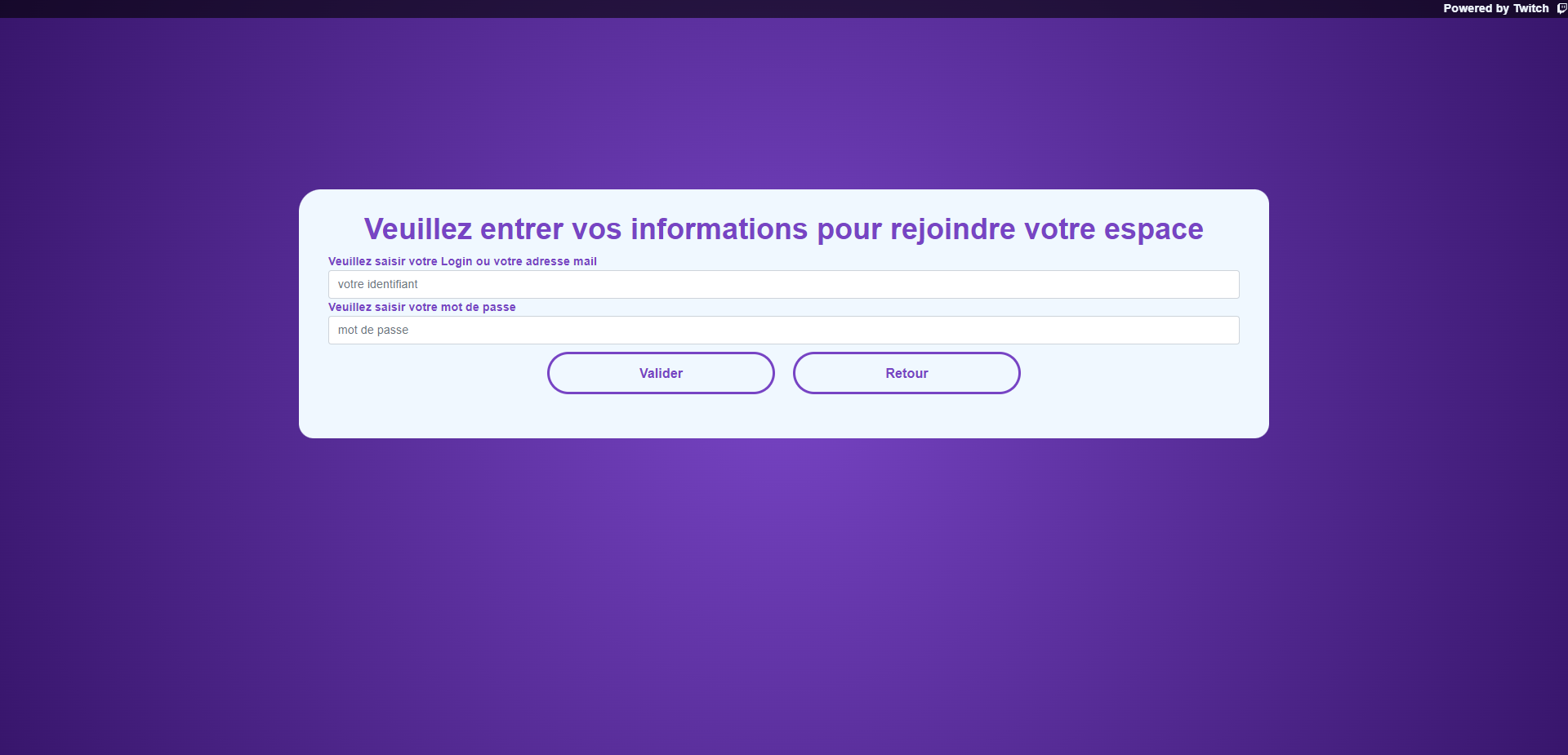
L’insertion de l’utilisateur se fait avec l’aide d’une requête préparée afin de se protéger des injections SQL. Chaque information qui a été set auparavant sont passée en paramètre de la fonction et une fois appelée dans le contrôleur, la requête est envoyée directement à la base de données qui l’exécute.



Et, si tout se passe correctement, un message informe notre utilisateur que son compte a bien été créé, nous ouvrons ainsi une session avec ses informations sur le site et l’envoyons directement sur son espace.

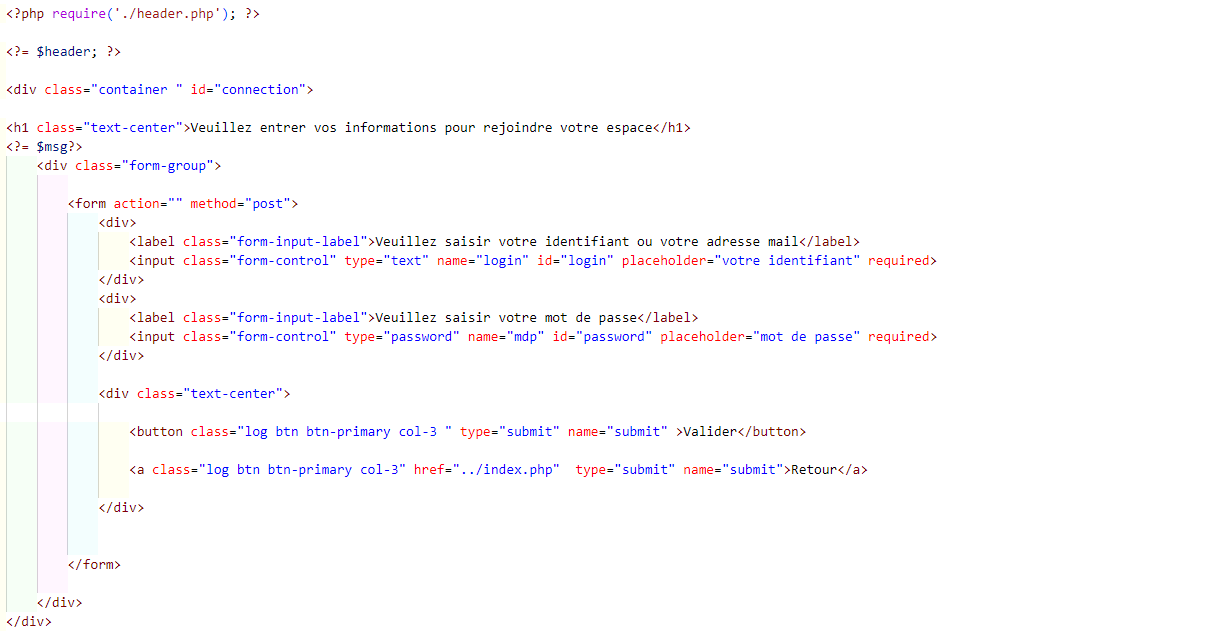
Imaginons que notre nouvel utilisateur se déconnecte, et décide de se connecter à nouveau. Nous allons donc observer le déroulement de cette action.

1. **La connexion de l’utilisateur**



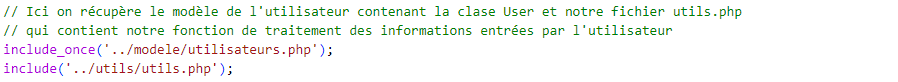
L’utilisateur se retrouve alors sur cette page qui va lui demander de renseigner deux champs, tout d’abord un champ comprenant soit son identifiant, soit son adresse mail et un autre champ pour son mot de passe.

Il s’agit ici d’un html très simple comprenant le header que nous avons vu auparavant sur la page d’inscription et d’un formulaire qui va récupérer encore une fois par une méthode $\_POST les champs qui seront remplis par l’utilisateur.

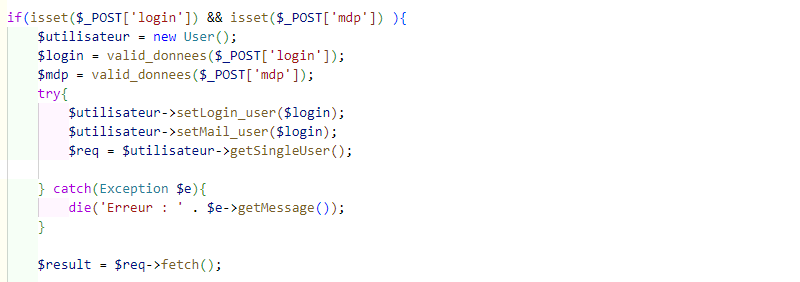


Une fois que notre utilisateur valide l’envoi des champs qui lui sont demandés, notre contrôleur va alors les récupérer. C’est là qu’entre en jeu notre fichier connexion\_user.php.

On va alors commencer par déclarer les fichiers dont on a besoin pour notre contrôleur.



On commence par vérifier que nos champs sont bien présents, et le cas échéant, on va instancier une nouvelle classe User.



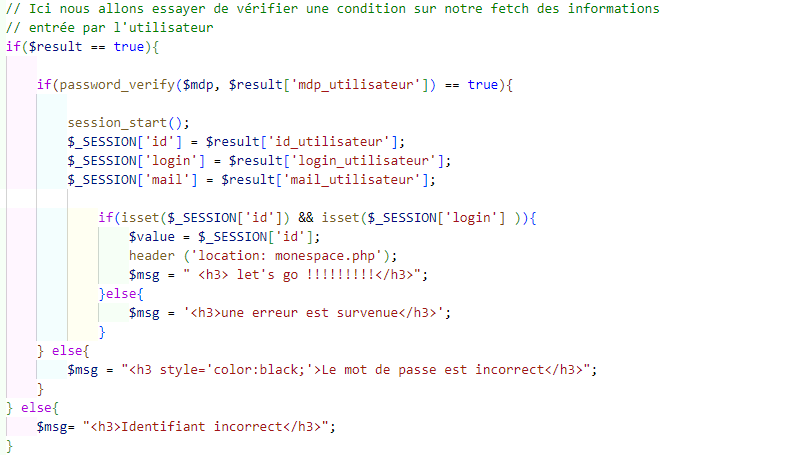
Notre fonction valid\_donnees () va alors traiter les chaînes de caractères qui ont été entrées par l’utilisateur.

On va par la suite essayer de trouver une correspondance stricte de l’utilisateur dans la base de données avec notre requête préparée getSingleUser (). Sur la base soit du pseudonyme de l’utilisateur soit de son mail.



Ceci fait, on va effectuer le **fetch** de notre requête qui sera stocké à l’intérieur de notre variable $result.

Nous établissons donc une condition qui va nous permettre d’affirmer ou non la présence de l’utilisateur dans notre base de données.



Si ce n’est pas le cas, alors l’utilisateur est informé que son mail ou son identifiant ne sont pas correct. Et cela met un terme à notre fonction.

Mais si c’est bien le cas, alors nous passons à l’étape suivante de vérification du mot de passe.

Avec la fonction native **PHP**, password\_verify (), nous cherchons donc une compatibilité avec le mot de passe haché qui correspond au compte du login utilisateur récupéré dans notre variable $result. Dans le cas ou il n’y a aucune correspondance, l’utilisateur est informé que le mot de passe qu’il a entré est erroné, et la fonction s’arrête.

Et encore une fois, dans le cas où les mots de passe correspondent, nous démarrons la session de l’utilisateur en y incluant son id, son pseudo, ainsi que son adresse mail.

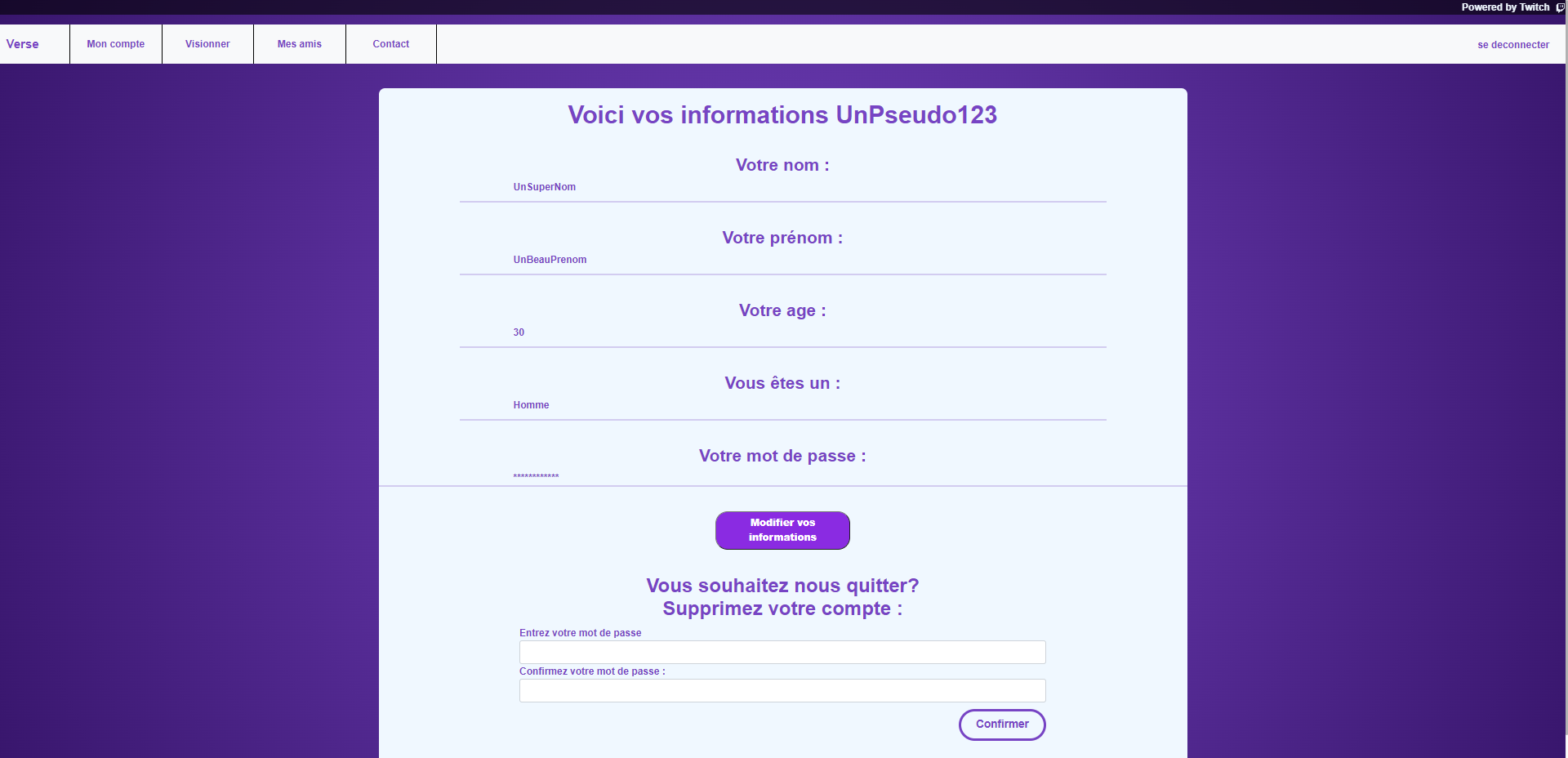
Une dernière vérification pour s’assurer que le login et l’id de l’utilisateur sont bien présent dans la session qui vient d’être ouverte. Le cas échéant notre utilisateur est envoyé directement sur son espace.

Maintenant que notre utilisateur est créé, qu’il peut se connecter, admettons qu’il décide de modifier ses informations sensibles. Tel que son pseudonyme ou son mot de passe.

Rendez-vous dans l’onglet Mon compte !

1. **Les modifications de l’utilisateur**

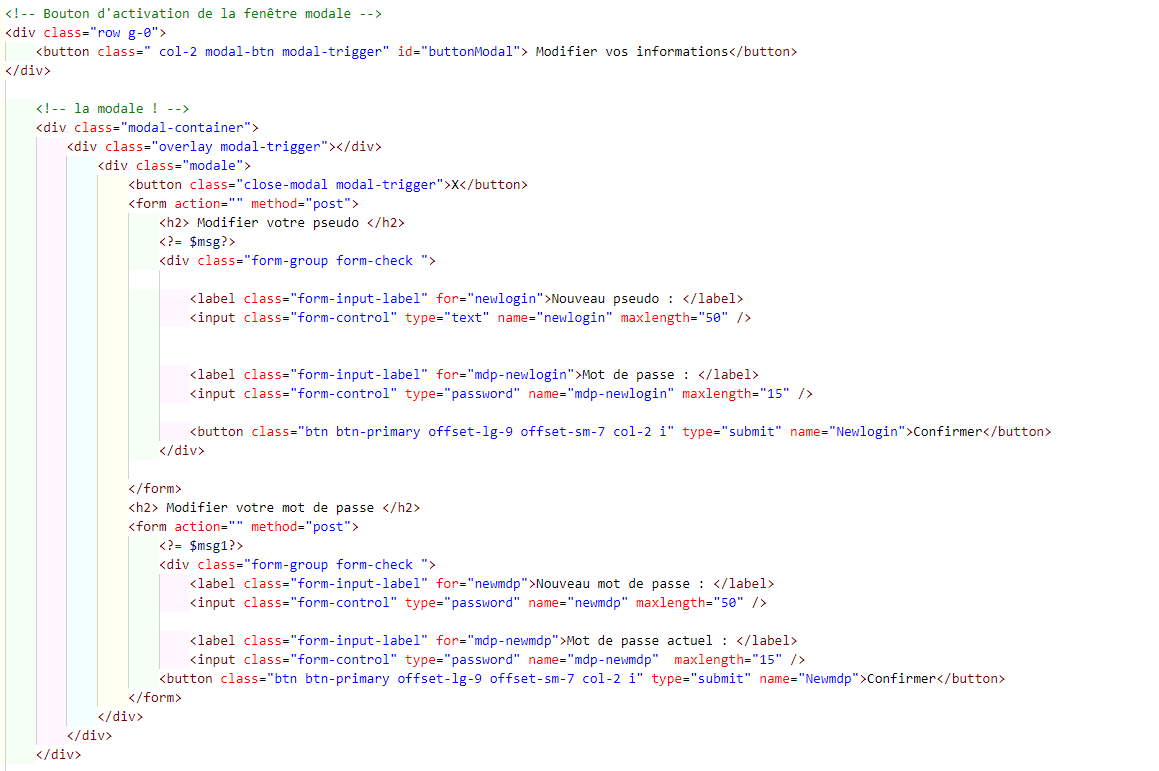
L’utilisateur a donc la possibilité de modifier ses informations sur notre site en accédant à la rubrique mon compte. En plus de l’affichage de toutes ses informations, il va retrouver un bouton l’informant qu’il peut effectuer des modifications.



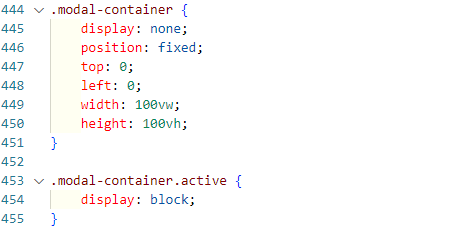
En cliquant sur ce bouton, va alors apparaître une fenêtre modale proposant de modifier son pseudonyme ou son mot de passe.



Pour présenter brièvement la modale dans le **HTML** :

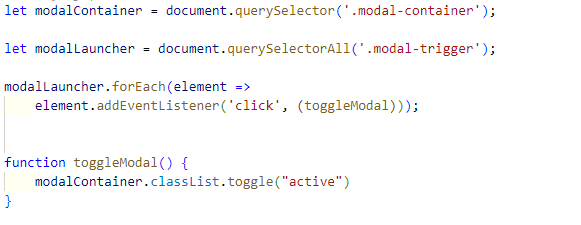


On a tout d’abord le bouton qui est au vu et au su de tous, et la modale qui est invisible.



On va faire en sorte de donner à notre conteneur une propriété display avec comme valeur none. Qui sera modifié par effet de cascade lorsque la classe ‘active’ sera ajoutée à ce même conteneur.

Et c’est là que notre script JavaScript entre en jeu :



On va donc récupérer notre conteneur ainsi que notre modale dans les deux variables qu’on déclare en haut du screenshot.

On va donc, pour chaque élément avec l’attribut de classe : ‘modal-trigger’, écouter un évènement de click. Qui fera appel à notre callback de toggleModal, qui fonctionnera comme un switch sur notre conteneur grâce à la méthode native toggle().

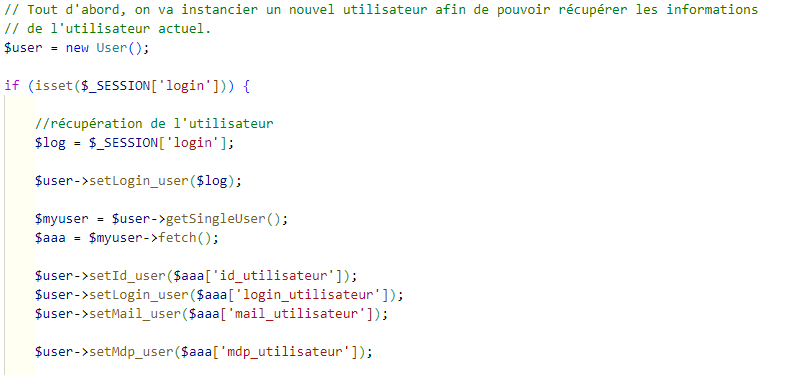
Ce qui nous permet de donner ou de retirer la classe active à notre modale.

Maintenant passons à notre modification du login et du mot de passe.

Nous plongeons à nouveau dans notre **PHP**.

On va alors retrouver encore une fois notre modele utilisateurs.php, qui est requis dès lors que l’on touche à la table utilisateurs.

Si on instancie un nouvel utilisateur et que récupère ses informations, c’est parce qu’il a beau être dans sa session, il ne transporte pas avec lui toutes ses informations. Le mot de passe par exemple lui est présent uniquement dans la base de données pour des raisons de sécurité.



Sur le même principe que vu précédemment, on reprend notre fonction getSingleUser() pour effectuer le fetch de notre utilisateur.

Une sur cette page, toutes ses informations sont récupérées et stockées dans notre $aaa.

On va donc pouvoir jeter un œil sur comment nous modifions le login.



Dans le cas où notre newlogin et notre mdp-newlogin sont bien dans notre superglobale $\_POST alors, on entame la fonction de modification du login.

On ne change pas les bonnes habitudes et on passe les champs à notre fonction valid\_donnees ().

Afin de modifier son identifiant, on demande à l’utilisateur son mot de passe, que l’on va alors vérifier. S’il ne correspond pas au mot de passe de son compte alors, on va lui notifier de cette erreur.

Sinon on vérifie que le fetch de notre utilisateur est toujours vrai et dans ce cas on va set l’id de l’utilisateur avec celui que nous avons récupéré et stocké dans notre $aaa, on va alors set le nouvel identifiant et on va faire appel à notre fonction de mise à jour de l’utilisateur, updateUser().



Notre requête préparée qui nous permettra alors de modifier les informations de notre utilisateur et qu’on retrouvera ensuite dans la mise à jour du mot de passe.

Une fois la modification du login effectuée, on modifie aussi l’identifiant dans la session. Et on redirige l’utilisateur vers la page d’accueil de son espace.

Pour la modification du mot de passe le déroulement de l’opération est très similaire.



1. **Conclusion**

Le projet **Verse** en est encore qu’à ses premiers balbutiements, je souhaite poursuivre ce projet le plus loin possible. J’apprécie toujours autant travailler dessus et j’aimerais énormément pouvoir l’utiliser avec mes amis. C’est

Au fur et à mesure que la formation avançait, je ressentais le besoin d’avoir un projet sandbox dans lequel j’aurais la possibilité de créer toutes sortes de fonctionnalités, continuer d’apprendre et expérimenter.

Mais pour les besoins de la formation, j’ai fait le choix de poursuivre sur des fonctionnalités qui seraient en rapport avec ce que nous étions en train d’apprendre. Mais surtout, un bon moyen de m’assurer que le projet puisse être un minimum présentable et que je puisse, un peu, me rassurer sur ce que je pouvais actuellement réaliser.

Ce projet n’est pas terminé, et j’aspire fortement à en poursuivre le développement, tout d’abord parce que ce serait amusant de pouvoir l’utiliser avec mes amis, et puis aussi pour continuer de monter en compétence. Il pourrait donc bien devenir ce bac à sable dans lequel je pourrais m’amuser.

Pour ce qui est des objectifs à court terme, je prévois l’intégration de TypeScript pour pouvoir en apprendre plus sur ce langage. Ce sera surement mon premier projet après la formation.

Il serait bien trop présomptueux de dire que le projet est tel que je le souhaitais, j’ai rencontré quelques difficultés, surtout dans la réalisation du launcher de vidéos et l’utilisation de l’API de Twitch afin d’intégrer le visionnage du contenu du site de streaming ne serait-ce que dans une page du compte utilisateur. Et le développement des fonctionnalités d’ajout en ami, de chat par message et les fonctions de réaction aux articles sont encore absentes. Avant d’utiliser Boostrap, j’ai consommé beaucoup trop de ressources de temps dans la réalisation du Front. Il a changé au moins 3 fois depuis le début du projet alors qu’à chaque fois il était déjà bien avancé.

J’avais beaucoup d’appréhension quant à l’utilisation de PHP et des bases de données, bien avant la formation, j’avais essayé d’apprendre PHP en autodidacte et j’avais eu énormément de mal à en comprendre le fonctionnement. Aujourd’hui je me rends compte que je n’avais surtout pas la bonne méthode pour apprendre le développement en général. Je ne prenais pas les choses dans le bon sens et ne respectait aucune logique. Et les bases de données relevaient presque du traumatisme de mes études en Qualité et logistiques industrielles où nous avions entrevu les Système de Gestion de Base de Données et ça me paraissait presque barbare. Je dois reconnaitre qu’aujourd’hui, les Bases De Données m’apparaissent bien plus utiles qu’auparavant.

L’important pour moi était de prendre du plaisir, et je me suis vraiment beaucoup épanoui dans la réalisation de ce projet. Si je dois retenir quelque chose de cette réalisation, c’est avant tout que je suis capable, que tout ne sera pas toujours facile, même les choses les plus évidentes, mais que j’aime vraiment beaucoup travailler avec les langages de programmation, particulièrement la réflexion derrière chaque chose. Je suis satisfait de la quantité de connaissances que le développement implique et mon besoin d’apprendre est plus que comblé quand je vois tout ce qu’il me reste à découvrir.

Pour la suite, l’après formation, j’ai quelques pistes pour débuter mon aventure en tant que développeur. Lorsque j’ai effectué mon stage, ma tutrice qui est une webdesigner indépendante, était en train de lancer une entreprise de communication sur Figeac. J’ai pu discuter avec l’équipe qui compose aujourd’hui cette compagnie et le courant était vraiment bien passé. Je me dis que c’est peut-être un bon moyen pour moi de mettre le pied à l’étrier.

Et pour finir, je tenais à remercier nos formateurs et l’équipe de l’Adrar pour la patience, et la disponibilité dont ils ont fait preuve.

Sans compter les savoirs qu’ils ont partagés avec nous.

Mentions particulières :

A **Jonathan GRUSON,** qui était notre formateur référant au cours de ces mois de formation, et qui personnellement a su m’accompagner et m’aider bien au-delà de ce qu’il doit le penser.

A **Rodolphe BRUMENT,** qui lui aussi nous a donné énormément de son temps, en restant très positif même dans les moments les plus stressants de la fin de formation.

Puis ma famille aussi, un classique, mais qui a su me soutenir, même s’ils se demandent encore pourquoi je suis tous les jours derrière un ordinateur.

Et merci à vous, qui avez lu ce document que j’ai rédigé. Vraiment, merci du temps et de l’attention que vous avez porté à ce mémoire.