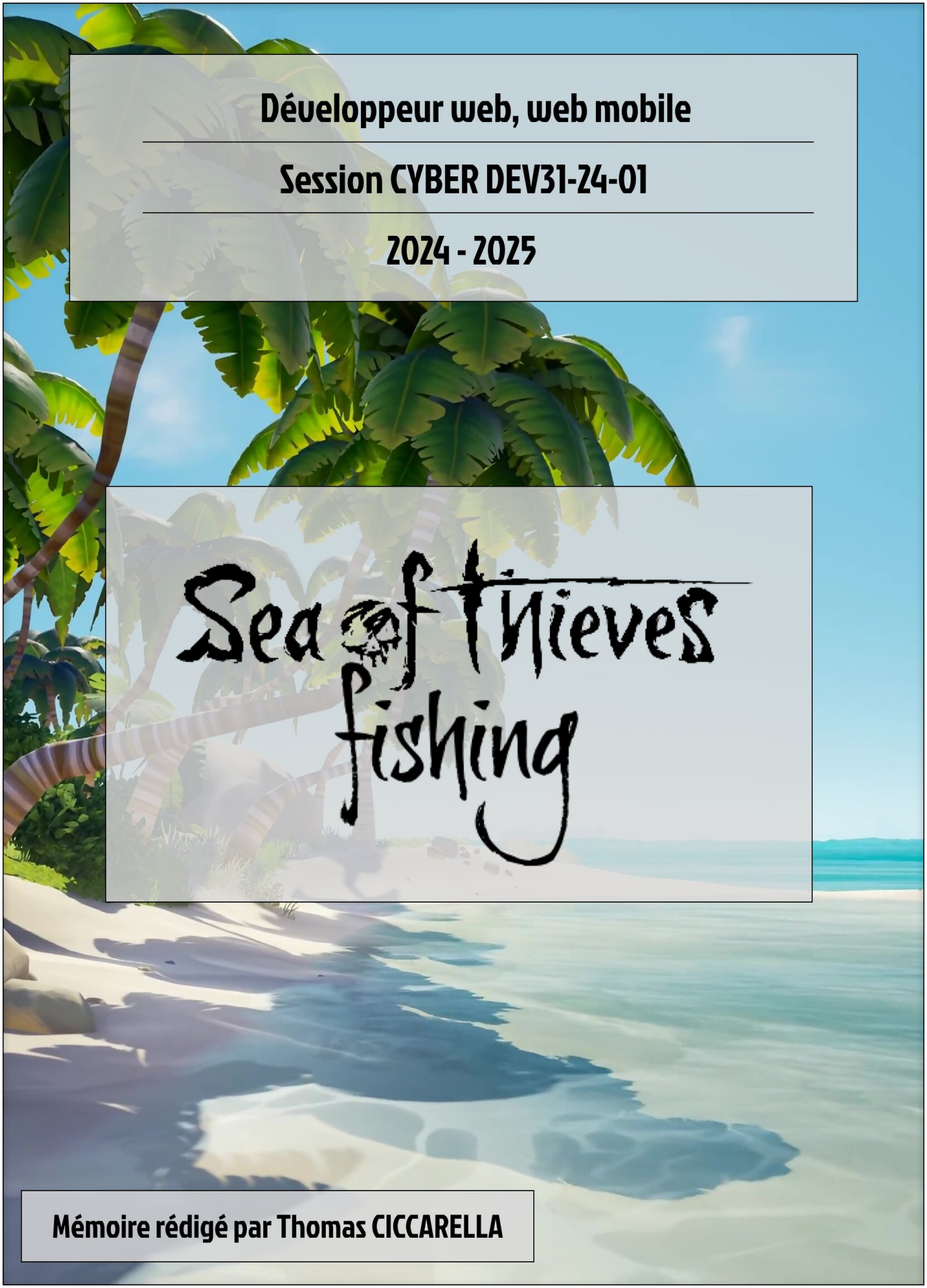


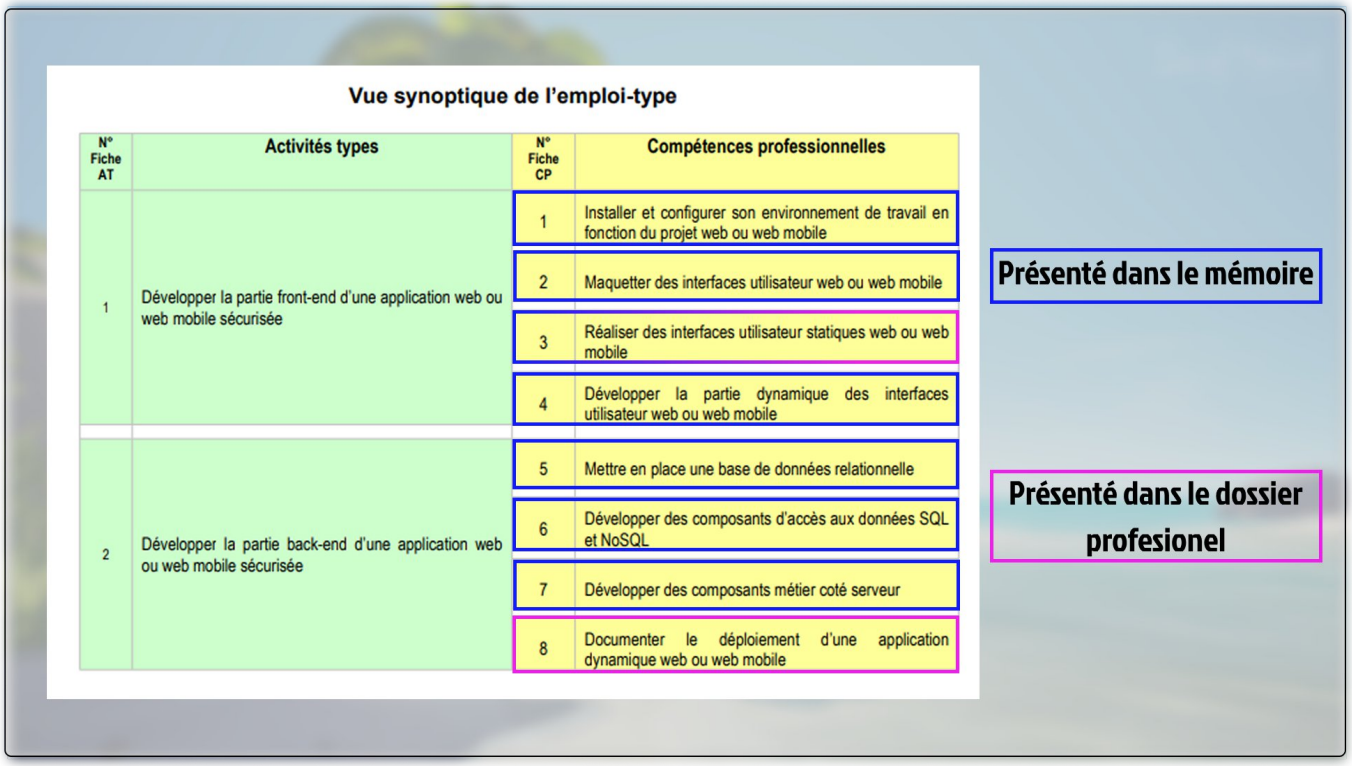
Développeur web, web mobile

Session CYBER DEV31-24-01

2024 - 2025

Sea of Thieves fishing

Mémoire rédigé par Thomas CICCARELLA



Vue synoptique de l'emploi-type

N° Fiche AT	Activités types	N° Fiche CP	Compétences professionnelles
1	Développer la partie front-end d'une application web ou web mobile sécurisée	1	Installer et configurer son environnement de travail en fonction du projet web ou web mobile
		2	Maquetter des interfaces utilisateur web ou web mobile
		3	Réaliser des interfaces utilisateur statiques web ou web mobile
		4	Développer la partie dynamique des interfaces utilisateur web ou web mobile
2	Développer la partie back-end d'une application web ou web mobile sécurisée	5	Mettre en place une base de données relationnelle
		6	Développer des composants d'accès aux données SQL et NoSQL
		7	Développer des composants métier coté serveur
		8	Documenter le déploiement d'une application dynamique web ou web mobile

Sommaire

1. Présentation de mon projet

1.1 Qui suis-je ?

Bonjour, je m'appelle Thomas CICCARELLA, j'ai 24 ans. Je suis né à EVRY Courcouronnes dans la région parisienne. Je suis venu à Toulouse pour des vacances juste après mes 18 ans et j'ai succombé au charme de la ville rose.

Je ne suis jamais reparti.

Étant joueur de rugby depuis mes 7 ans, Toulouse était la ville la plus adaptée pour me lancer dans ma vie de jeune adulte. Elle m'a également permis de débuter ma vie de jeune travailleur.

Sortant d'un CAP métallier j'ai pu passer des licences de soudure, mais n'étant pas satisfait par ce corps de métier j'ai décidé d'y mettre un terme.

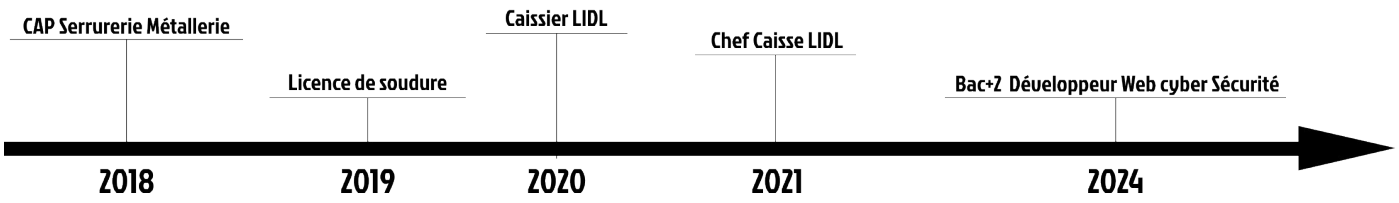
Je me suis reconvertie dans la grande surface chez LIDL où j'ai exercé le poste de caissier durant le COVID, puis le poste de chef caisse pendant 2 ans.

À mes 23 ans, j'ai eu un déclic et j'ai décidé de me lancer dans un tout nouveau projet.

N'ayant jamais eu la fibre scolaire jusqu'ici, j'ai décidé à 23 ans que je mettrais en œuvre mon projet que je me refusais depuis des années. Cela me paraissait inatteignable au vu de mon niveau académique.

Je suis un passionné d'informatique, de hardware, de jeux vidéos et avec une très grande soif de création. Je me suis dit que le métier de développeur serait le métier parfait pour moi. Je me suis donc tourné vers L'ADRAR qui m'a offert la possibilité de faire mes débuts dans les études supérieures, sans brevet ni BAC en main, et pour cela je les remercie infiniment.

Cela me mène enfin à vous introduire et présenter ce projet qui, je l'espère, me permettra d'obtenir mon diplôme et continuer mes études par la suite.



1.2 Abstract

What is my project?

My project is a website on the game SEA OF THIEVES, but for you to understand my project I will explain what is SEA OF THIEVES.

Sea of thieves is an online pirate game in the open world. In this game you can explore the map with totally free movement and you can practice many activities such as for example: explore, fight with other players, Search treasure boxes, chase animals, players, etc…., fish, and so on.

Fishing in this game is so entertaining ! There are a lot of fish in this game, in each region you can catch a different type of fish. All fish have a different price and rarity.

Personally, in this game I love fishing and so I want in this project to detail all fish and their price.

You may wonder why I have decided to create this website. Well, when I play this game I don't have information and I don't know where to start my fishing session and where the specific fish can be found or the price.

This website lists all information about fishing in this game for all gamers, I wanted a tool that is intuitive and user-friendly for more comprehension. Everything is available on my website and it is not necessary to create an account to visit the website.

1.3 Qu'est-ce que Sea Of Thieves ?

Comme je vous l'ai expliqué précédemment, je suis un passionné de jeux vidéos et aujourd'hui je vais vous parler d'un jeu en particulier : **Sea Of Thieves.**

Pour que vous compreniez mieux le sens de mon projet, je vais essayer de vous expliquer le principe et le fonctionnement de ce jeu.

Sea Of Thieves est un jeu de pirates en multijoueur, c'est un jeu que l'on appelle **OPEN–WORLD.** Le principe d'un jeu en open-world est que vous n'êtes pas guidés dans vos déplacements, vous êtes libre d'aller où vous le souhaitez, vous n'êtes pas dans un couloir avec un début de mission et une fin, vous pouvez donc vous déplacer librement dans une grande carte à votre guise en solo ou à plusieurs.

Dans ce jeu, vous pouvez vivre de grandes aventures pour faire fortune. Vous pouvez vous lancer seul ou avec votre équipage dans ce monde, mais par où commencer ? Il faut savoir que beaucoup d'activités s'offrent à vous dans ce jeu. Vous pouvez faire des quêtes proposées par le jeu, il y en a plusieurs types :

- De la recherche de coffres aux trésors
- Des attaques de forteresses marines
- Du transport de marchandises

Et bien d'autres...

Il y a aussi toute une partie multijoueur où vous pouvez attaquer d'autres équipages de joueurs sur la carte, il faut donc rester vigilant.

On trouve également des activités annexes, secondaires et moins mises en avant. Mon projet a pour objectif de mettre en lumière l'une d'entre elles.

Dans ce jeu, vous pouvez pêcher différents types de poissons en fonction de la zone dans la qu'elle vous vous trouvés. Il y a tout un système de rareté pour chaque espèce de poissons. Je vais vous expliquer cela plus en détail .

Ci dessus, vous avez une espèce de poissons nommée **Déferlantes**. Vous pouvez la pêcher dans la région **« The Wild** » (en violet sur la carte). On trouve les Déferlantes normales et les Déferlantes Trophées, ces dernières étant plus grosses plus chère et plus rare.

Chaque espèce de poisson a ses variants, normaux et trophées. De plus, certaines espèces sont plus chères que d'autres.

1.4 Sea Of Thieves Fishing

Mon site s'appelle **Sea Of Thieves Fishing**, il a pour but de référencer toutes les espèces de poissons et leurs variants afin d'indiquer leur valeur et la zone dans laquelle on peut les pêcher. Il y a aussi une petite partie du site dédiée à la chasse dans le jeu.

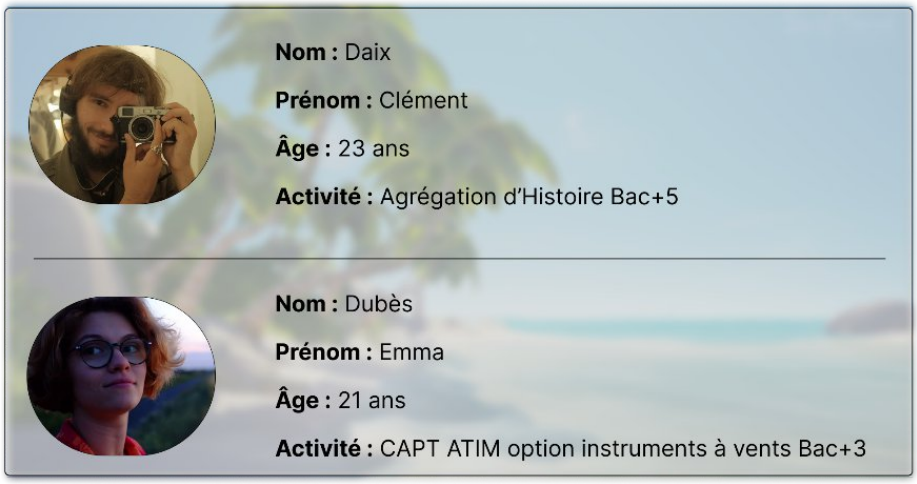
Mais qui utiliserait ce site et pourquoi ?

1.5 Persona pour ce site

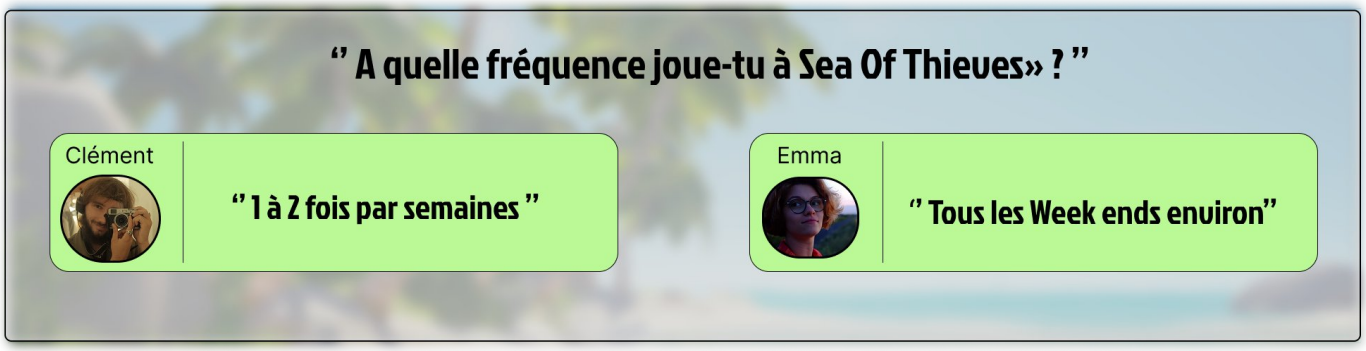
Le public que je vise sont des joueurs connaissant le jeu et voulant se renseigner sur la pêche pour avoir un maximum d'informations en un minimum de temps sur tout type de support, que ce soit sur ordinateur, tablette ou même portable.

J'ai donc décidé de chercher un joueur et une joueuse du jeu pour leur demander le point de vue qu'ils avaient sur la pêche dans Sea of Thieves.

Je vous présente donc les deux personnes qui se sont prêtées au jeu de mes questions.

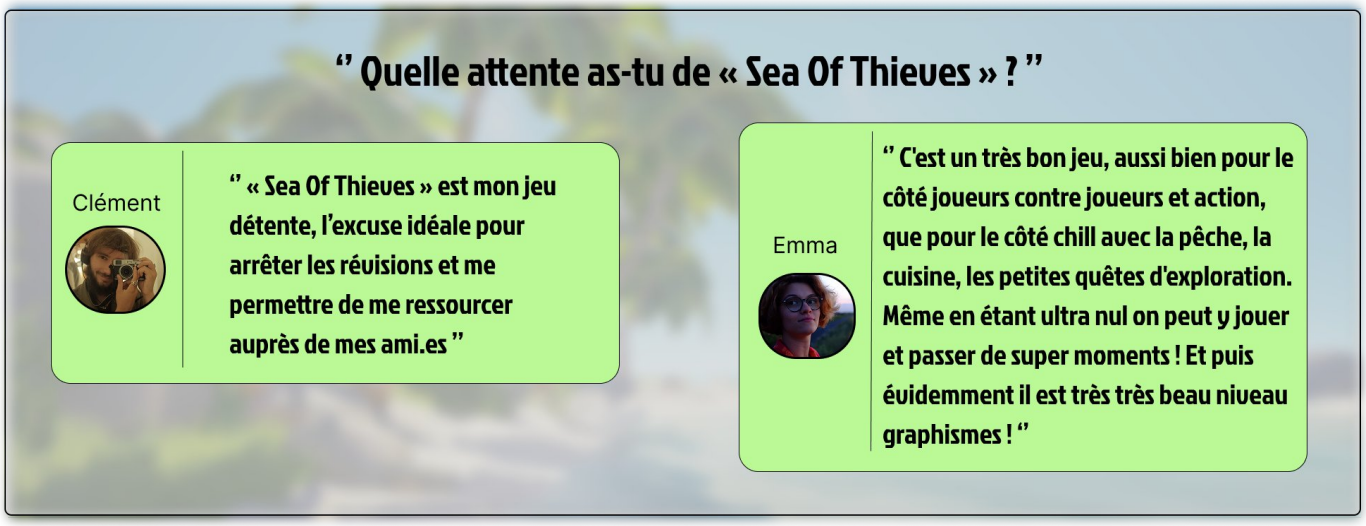


Ma première question consiste à savoir si ce sont des joueurs réguliers de Sea Of Thieves pour connaître leur attachement à ce jeu et leur investissement.



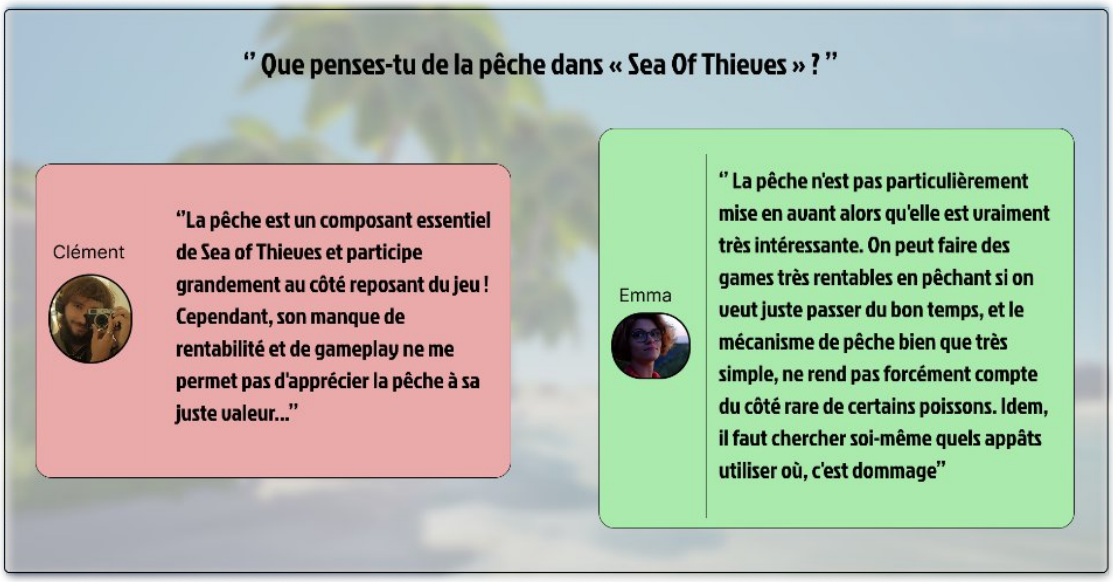
Comme on Peut le voir, ce sont des joueurs réguliers. Ils sont pour l'instant 2 cibles totalement probables pour mon site.

Ma deuxième question était plus orientée pour voir quelle vision ils avaient du jeu.



Les deux réponses me confortent sur le fait que mon site pourrait être utile.

Voici la question la plus cruciale. Maintenant que je sais quel type de joueurs j'ai en face de moi, j'aimerais savoir plus spécifiquement comment il se positionne par rapport à la pêche dans ce jeu.



Les deux réponses sont très intéressantes, elles me permettent de déceler deux visions de la pêche très différentes. Clément trouve que la pêche n'est pas rentable alors que cela reste une activité essentielle dans le jeu. Emma, pour sa part, trouve la pêche très intéressante et rentable.

1.6 Analyse et conclusion persona

Ces 2 visions reflètent des problèmes assez clairs, le manque d'information et d'accessibilité via le jeu.

- Clément trouve que cela n'est pas rentable, mais a-t-il eu connaissance de toutes les espèces et du niveau de rareté de chaque poisson? Les missions principales sont rentables au niveau pécuniaire, mais certains poissons sont plus chers et plus rapides à obtenir que les grandes missions proposées par le jeu. Pourrait-il changer d'avis ? Grâce aux informations de mon site ? En tout cas, Clément ne serait pas concerné par mon site aux premiers degrés.

- En revanche, Emma serait totalement la cible visée par mon site. comme elle le dit si bien, le manque d'information donné par le jeu peut desservir cette activité. Grâce à mon site, elle pourrait avoir toutes les informations dont elle a besoin rapidement pour optimiser ses sessions de pêche.

Pour conclure, ces deux avis m'ont apporté des informations et renseignements cruciaux quant-à l'orientation de mon site. Dans un premier temps il faut qu'il puisse apporter un maximum d'informations aux utilisateurs pour leur permettre d'optimiser leur session de pêche, et surtout de faire prendre conscience à ces joueurs qui n'ont pas les informations que la pêche peut être un attrait tout aussi intéressant que les quêtes principales du jeu.

Et pour cela, il va falloir que mon site soit simple et intuitif avec des visuels aussi clairs et agréables que possible pour que tout type de joueur puisse comprendre et naviguer sur mon site dans les meilleures conditions possibles.

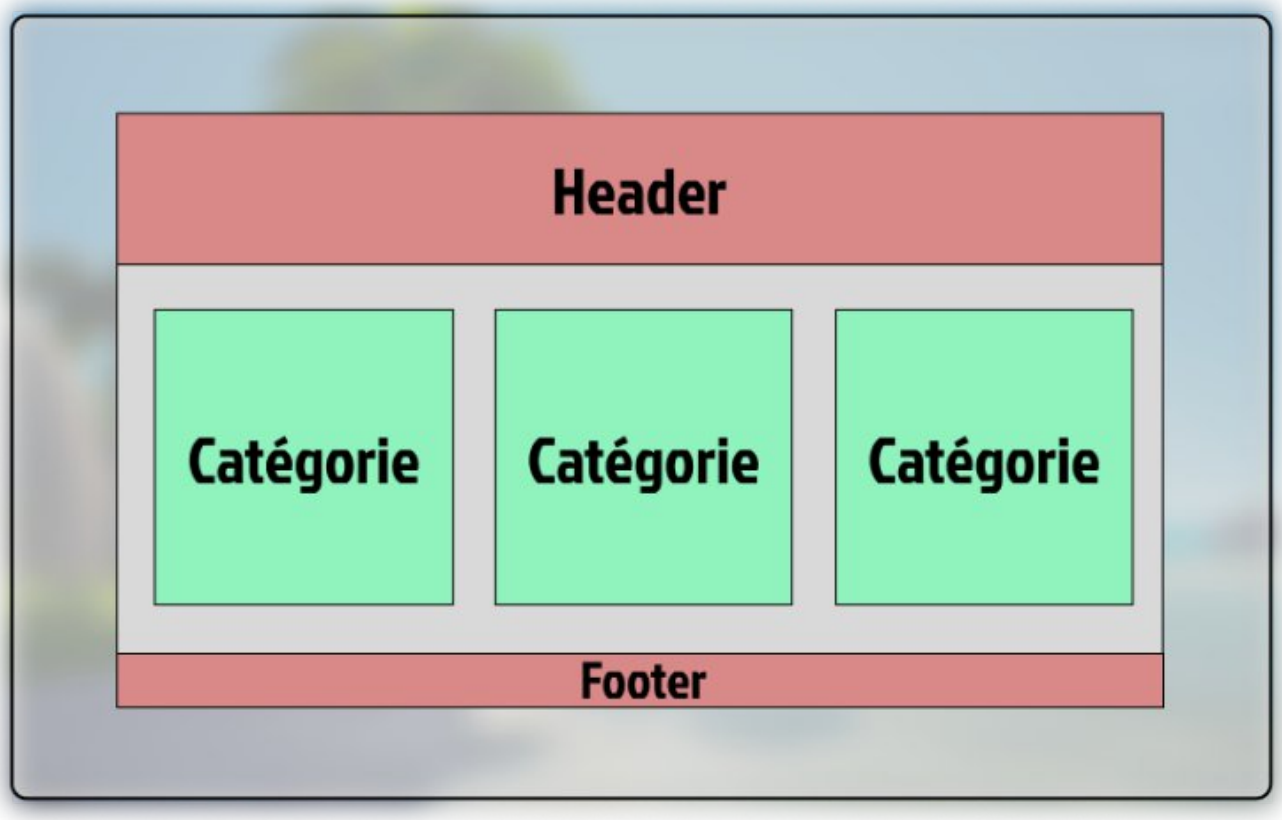
2. Maquettage

Pour que le site soit le plus intuitif possible, je voulais que la première page sur laquelle un visiteur arrive soit épurée et agréable visuellement. Pour cela il faut effectuer ce que l'on appelle un maquettage. Le maquettage sert à créer le visuel du site pour les équipes chargées du développement et également à donner une idée au client du visuel. Cette étape favorise une meilleure compréhension entre l'attente du client et la mise en pratique des équipes de création pour que chacun sache dans quelle direction le projet se dirige.

2.1 Zoning

Pour commencer j'ai effectué un zoning. Le zoning est l'une des cinq étapes cruciales pour l'identité visuelle d'un site car il permet d'assigner les zones des éléments qui vont constituer la page.

Vous avez donc ici le zoning de la page d'accueil :

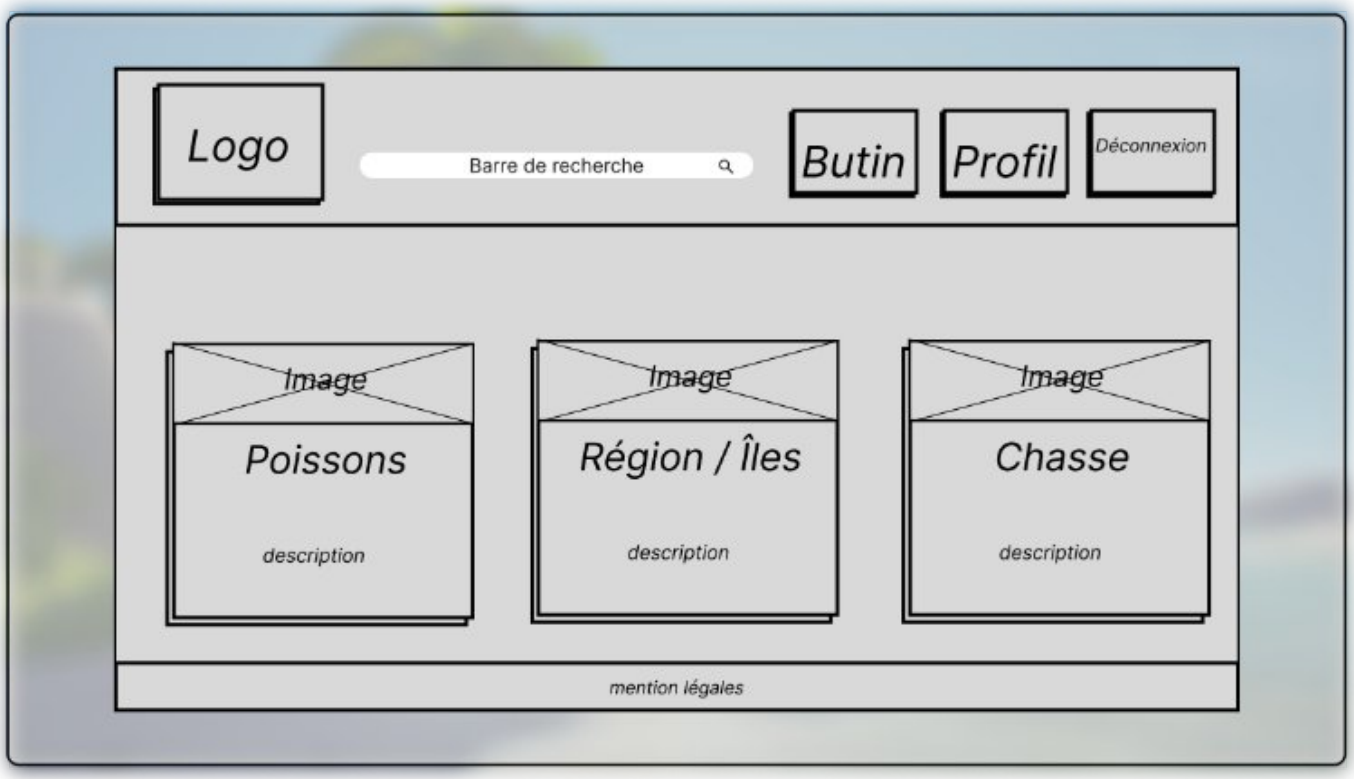


- Le header (l'entête de page où plusieurs éléments pourront être implantés)
- Catégorie (c'est dans cette zone que l'on va mettre les éléments principaux de la page d'accueil)
- Le footer (pour afficher diverses informations secondaires)

2.2 Wireframe

La deuxième étape du maquettage est le wireframe, il part des zones définies dans le zoning. Il va donner une notion de fonctionnalités du site comme par exemple les boutons, les zones cliquables, etc.

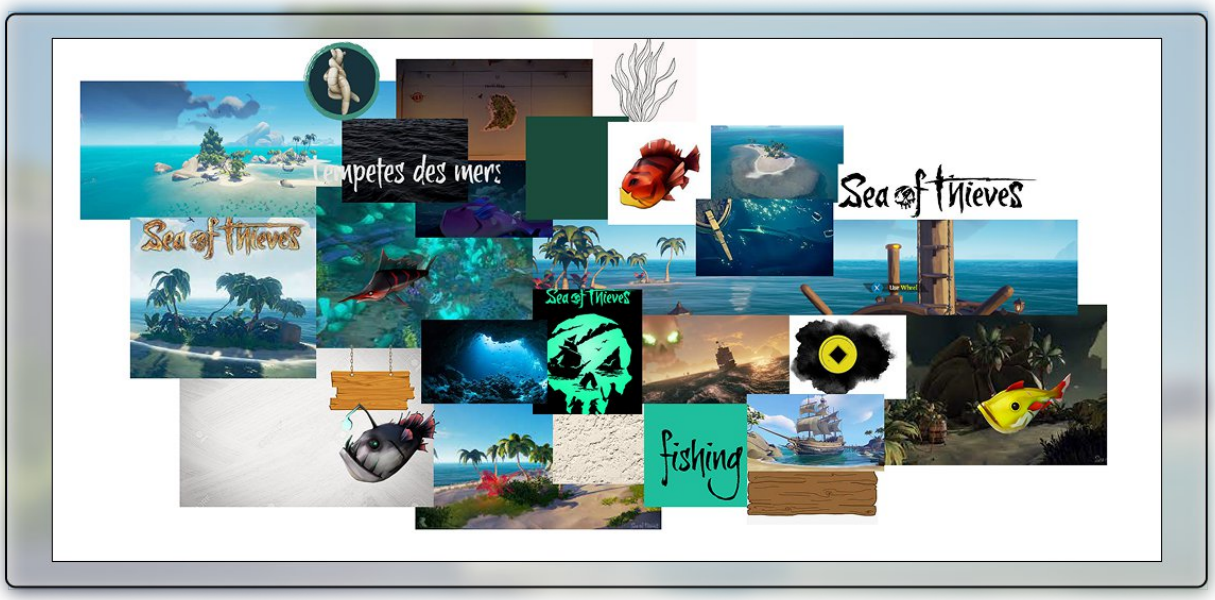
Vous avez donc ici le wireframe de la page d'accueil pour faire suite au zoning :



- Pour commencer, le header (l'entête) auquel nous avons ajouté les éléments suivants : le logo / une barre de recherche / le butin / le profil / la déconnexion. Ils sont tous cliquables et interactifs avec des effets quand on les survole.
- Vous avez ensuite la zone catégorie où trois onglets cliquables vous présente les trois principaux sujets de ce site : les poissons / les régions et îles / la chasse, ils ont tous une image, un titre et une description pour une meilleure compréhension.

2.3 Mood board

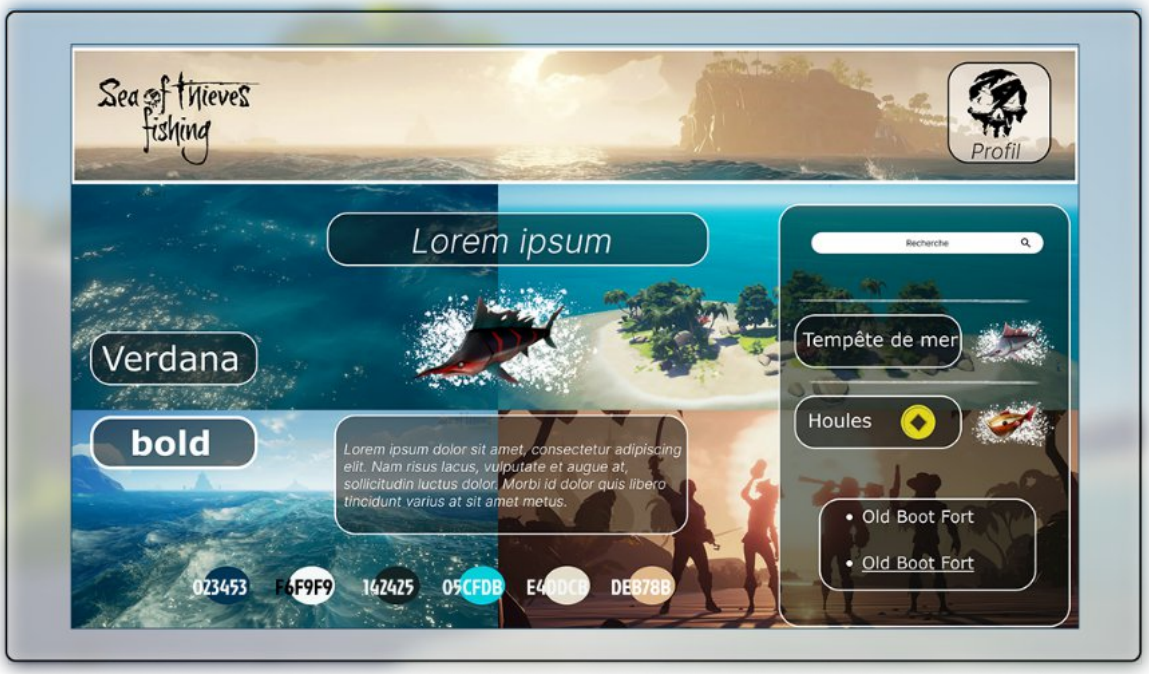
Le mood board est la troisième étape du maquettage. Elle est un peu différente des deux précédentes, le but d'un mood board étant de donner une idée de l'ambiance visuelle du site, les inspirations vers lesquelles il va tendre.Le mood board est surtout destiné aux équipes de création pour les aider à trouver une direction artistique (on l'appelle aussi la DA).



- Sea of thieves est un jeu de pirate qui se passe dans des mers équivalentes aux Caraïbes. Les tons de couleurs son donc très clairs et bleutés.
- Mais il y a également des éléments avec des couleurs assez vives qui peuvent dénoter parmi la clarté principale ce qui peut donner une certaine forme d'importance et priorité dans le regard. Il faut donc jouer avec ces éléments pour créer une harmonie visuelle qui rendra le tout agréable au regard.

2.4 Style tiles

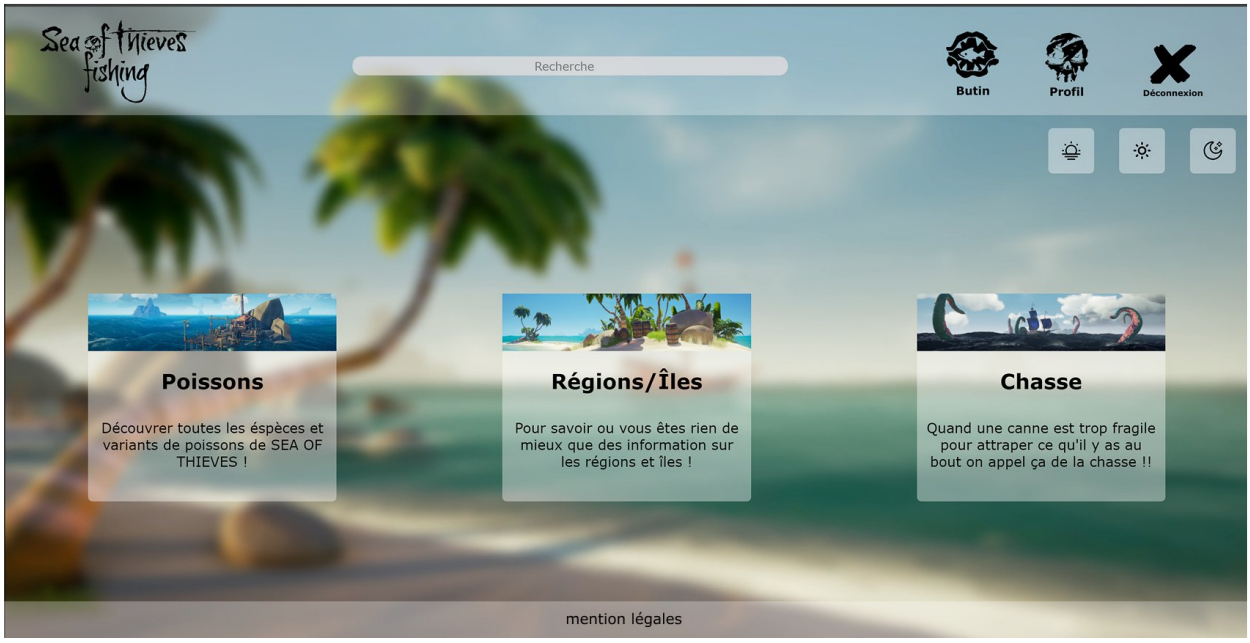
Dans la continuité d'un **mood board,** le **style tiles** est donc la quatrième étape du maquettage. son rôle est de mettre en place la direction artistique choisie avec le choix des couleurs, des polices et tout ce qui touche au style du site.On va proposé différent visuels avec des codes couleur et des noms de police pour que les équipes sachent ce qui est choisi et pourront utiliser lors de la création.



- En ce qui concerne les couleurs choisies, j'en ai sélectionné six qui rappellent les principales couleurs d'une plage des Caraïbes, avec un peu de blanc et de noir pour les textes.
- Quant aux police j'ai fait le choix de n'en conserver qu'une seule avec pour uniques variation l'épaisseur des lettres. J'ai choisi la police **Verdana** que je trouve simple et très lisible. De plus, elle dénote du décor et de la police utilisée par le jeu qui est très singulière. Elle est donc parfaite pour bien attirer l'attention sur les textes contenant les informations importante.
- Vous pouvez également y retrouver certains éléments graphiques comme la monnaie du jeu et les visuels des poissons qui seront référencés sur le site.

2.5 Mockup

La cinquième et dernière étape du maquettage est le **mockup**. Son rôle est de donner le visuel définitif du site, ici celui de la page d'accueil. On peut l'utiliser pour faire un retour au client afin qu'il valide le visuel final pour son site.



Voici la version finale de la page d'accueil de mon site Sea Of Thieves Fishing. C'est le résultat de toutes les autres maquettes :

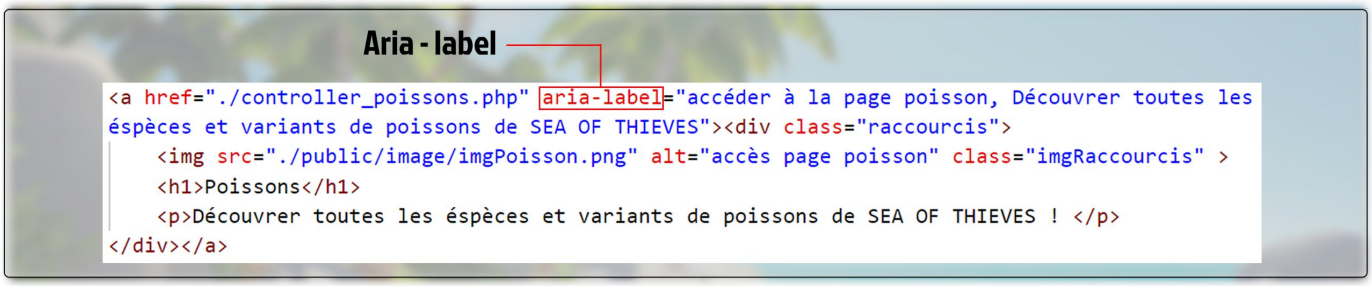
- Nous avons suivi le **zoning** pour sectoriser la page.
- Puis nous avons ajouté les fonctionnalités et les éléments du **wireframe.**
- Et pour terminer nous avons appliqué la direction artistique choisie pour donner le visuel final avec toutes les informations du **style tiles**.

3. Accessibilité - performance

L'accessibilité dans un site web est primordiale. Si vous voulez que tout le monde, y compris les personnes souffrant d'un handicap (visuel, auditif, moteur, etc.), puisse y avoir accès, il faut mettre en place certains éléments et faire certains tests.

3.1 Adaptabilité aux personnes handicapées

Une des mises en place possibles sont les « aria-label » qui ont pour rôle d'aider les personnes utilisant un lecteur d'écran. Un « aria-label » va servir à décrire ce avec quoi l'utilisateur interagit.



Dans l'exemple ci-dessus, vous avez le code HTML de l'accès à la page « Poisson » dans l'accueil de mon site. **L'aria-label** va décrire la possibilité d'action et expliquer ce que l'on trouve dans la page grâce à la description que les utilisateurs non-handicapés peuvent voir.

Il y a aussi des tests pour vérifier les contrastes et la gestion des couleurs pour les personnes daltoniennes. Nous vérifions donc que les contrastes et les couleurs s'adaptent bien à leur handicap pour leur offrir une navigation sur le site web agréable.

Pour cela je me suis servi de l'extension Chrome (colorblindly) qui m'a permis de tester plusieurs types de daltonisme et vérifier si l'affichage reste bien lisible.

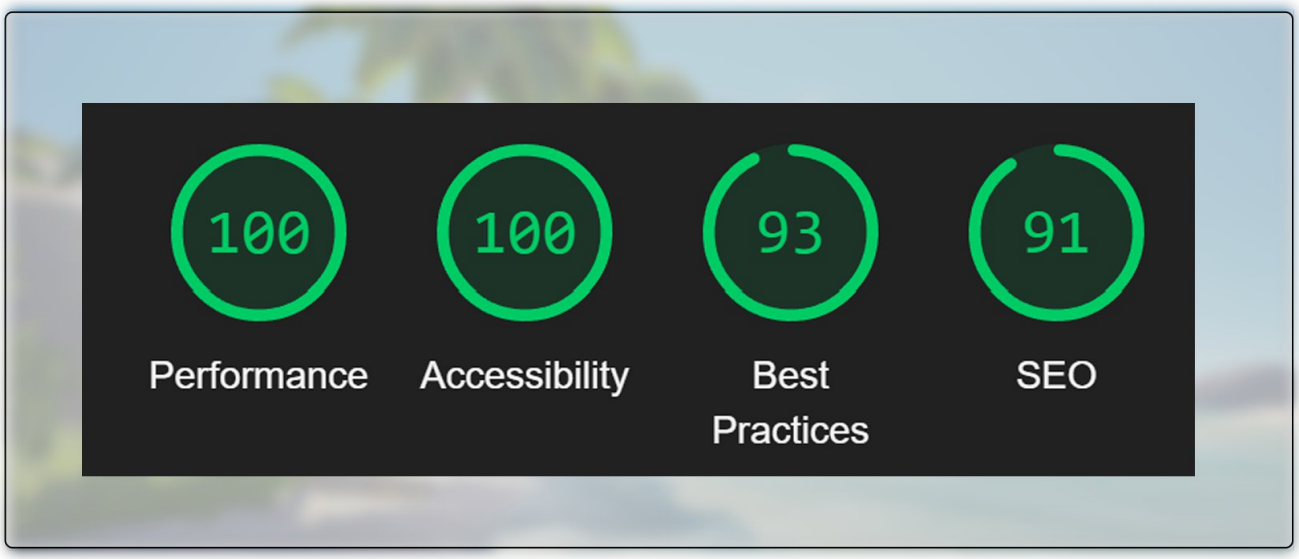
Vous pouvez retrouver ici un exemple de la page d'accueil de mon site vue par une personne atteinte de **tritanopia** .

3.2 Analyse et optimisation

Pour qu'un site soit performant et bien optimisé, on peut s'aider d'outils qui vont noter et relever les secteurs cruciaux pour qu'un site soit de bonne qualité.

LIGHTHOUSE est une extension Chrome qui nous offre ce genre de service. Elle permet de noter notre site dans plusieurs registres et nous montre les points à travailler.

Ci-dessous l'analyse que **Lighthouse** a faite de ma page d'accueil.



- **La performance** représente le temps de chargement, la fluidité de navigation et l'optimisation de la page. Dans mon cas, la page d'accueil est performante, elle charge vite et est adaptée à tout type de connexion sur ordinateur comme sur mobile.

- **L'accessibilité** vérifie si la page est compatible avec les lecteurs d'écran, si elle respect des contrastes et la bonne utilisation du balisage HTML. Dans mon cas, j'ai adapté ma page d'accueil aux personnes en situation de handicap et j'ai également vérifié les contrastes.
- **Les bonnes pratiques** vérifient si le site est bien sécurisé, s'il y a des éléments inutiles et si les conventions de développement son respectées au maximum. Dans mon cas, j'utilise certaines balises de manière un peu différente de ce qui est attendu. Cela sera un point à améliorer par la suite.
- **Le SEO** est l'optimisation pour les moteurs de recherche, il vérifie les formats et les normes attendus par les moteurs de recherche. Dans mon cas, certains textes sont un peu petits par endroit.

4. UML

Pour notre site, il est primordial de réaliser un **UML (Unified Modeling Language)**. Un UML est un langage de modélisation qui est représenté par des diagrammes. Ces derniers sont là pour schématiser visuellement un système d'information. Ils permettent de décrire la structure ainsi que les échanges entre les acteurs **(utilisateur, visiteur du site)** et le système d'information. Il va montrer le comportement et les composants de ce dernier. Pour faire plus simple, on peut dire que son rôle est de :

- **Visualiser :** représenter graphiquement les concepts et les relations entre utilisateur et système.
- **Spécifier :** définir les exigences et les fonctionnalités du système.
- **Construire :** guider le développement du système.
- **Documenter :** offrir une documentation claire et structurée pour les équipes de développement.

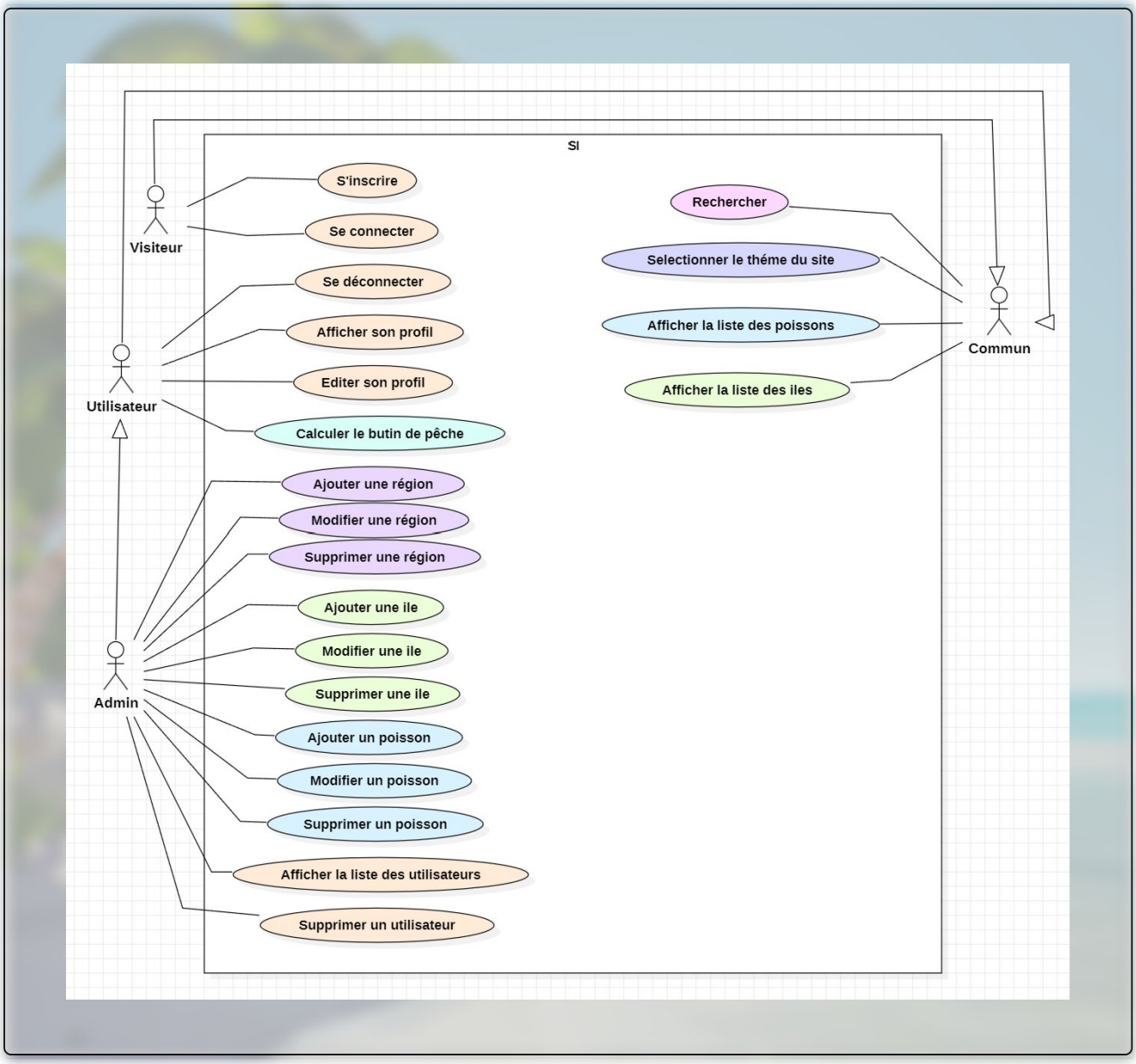
Il existe plusieurs types de diagrammes UML mais nous allons nous attarder sur un en particulier : Le **Use Case**.

4.1 Use Case

Le USE CASE ou cas d'utilisation en français consiste à décrire l'échange entre un acteur et le système d'information. Son rôle est de montrer ce que les différents acteurs peuvent et ne peuvent pas faire avec le système d'information.

- Ce que j'appelle les **ACTEURS** sont les utilisateurs du site. Chaque acteur peut avoir un rôle, un nom et des droits différents sur le site. Exemple d'acteur : **un visiteur / un utilisateur / un admin**. Dans un Use Case, ils sont en général représentés sur les côtés par des bonhommes bâtons avec leur rôle noté en dessous.
- Le rôle du **système d'information (SI)** , est d'encapsuler les fonctionnalités du site avec lesquelles les acteurs vont interagir. Le SI est représenté au centre du use case, c'est un grand rectangle blanc contenant toutes les fonctionnalités du site.

Vous pouvez donc voir ci-dessous mon Use Case qui est composé de quatre acteurs :



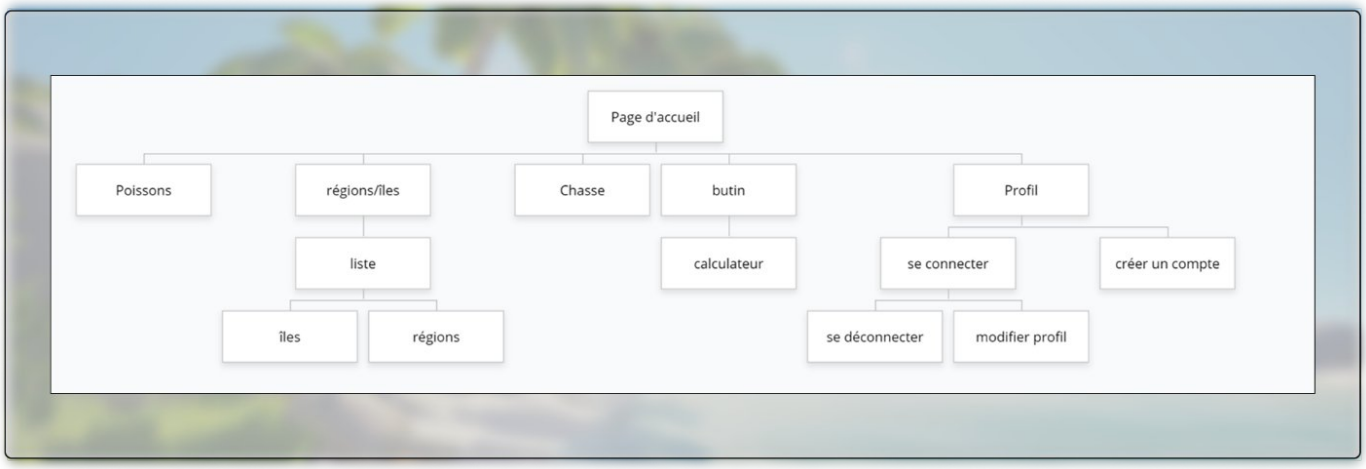
- **Le visiteur :** il représente une personne venant sur le site web sans être connectée. Il a donc accès à la fonctionnalité « s'inscrire » ou « se connecter ».
- **L'utilisateur :** il représente une personne qui a créé un compte et s'est connectée à mon site web. Elle a donc plus de possibilités que le visiteur, et elle peut utiliser les fonctionnalités suivantes :
    - **Se déconnecter :** cette fonctionnalité citée plus haut permet de quitter le rôle d'utilisateur sur le site et de retourner au rôle visiteur. Par conséquent vous perdez aussi toutes les fonctionnalités offertes au rôle utilisateur.
    - **Afficher son profil :** cela consiste à visionner les informations de son compte (nom /prénom / adresse mail).
    - **Éditer son profil :** si l'utilisateur veut modifier son profil après sa création.
    - **Calculer le butin de pêche :** cette fonctionnalité est destinée uniquement aux utilisateurs et elle permet de calculer le total de la session pêche de l'utilisateur.
- **Commun :** c'est un utilisateur virtuel qui il n'existe pas à proprement parler. Il sert à regrouper ce que l'utilisateur et le visiteur ont en commun.

- **Admin :** il représente celui qui aura le plus de droit sur le site son rôle sera de tenir à jour le site pour les visiteurs et utilisateurs . Dans ce cas de figure nous avons une **notion d'héritage**, c'est-à-dire que l'admin va hériter de tous les accès et droits du visiteur et de l'utilisateur . Il pourra :
    - **Ajouter et supprimer** des poissons / des régions / des îles : le but est que l'admin tienne à jour le site et donc il a toutes ces possibilités.
    - Il peu également **supprimer** un utilisateur si jamais le besoin s'en fait ressentir.

Pour conclure sur le Use Case et justifier mes choix, je dirais que j'ai cherché à faire un site le plus accessible possible, sans aucune contrainte de connexion pour rendre l'expérience plus agréable au visiteur et qu'il ne se sente pas restreint au niveau des informations dont il pourrait avoir besoin.

4.2 Arborescence

Pour avoir un site facile d'accès et intuitif au maximum je vous propose de vous montrer l'arborescence de mon site pour mieux comprendre son fonctionnement et ses différentes pages.



Tout d'abord, quand vous arrivez sur mon site, vous êtes sur la page d'accueil. Elle référence les 3 principales sources d'information :

- **Poissons**
- **Régions / îles**
- **Chasse**

La partie « Poissons » redirige sur une page contenant toute les information sur les poissons dans le jeu.

La partie région est représentée par une liste. Elle renvoie tout d'abord sur une page comportant toutes les îles, rangées par région dans des blocs bien distincts. Chaque île est cliquable et envoie vers la page de renseignement de l'île en question.

La partie chasse elle envoie sur une page similaire aux poissons avec toutes les informations sur les animaux chassables du jeu.

Mais quand vous êtes sur la page d'accueil, vous aurez aussi d'autres possibilités d'accès qui seront disponibles dans l'entête.

En tant que **visiteur**, vous aurez accès au bouton profil qui, vous redirige vers la page de connexion pour créer un compte sur le site.

Mais si vous êtes un **utilisateur**, vous avez plus d'éléments affichés comme la fonctionnalité butin qui vous permet de calculer le total de vos sessions de pêche

Vous avez aussi accès à votre profil en cliquant sur l'onglet profil,qui vous offre un visuel de vos informations personnelles : prénom / nom / email.

Pour finir, vous avez la possibilité de vous déconnecter via le bouton déconnexion qui se trouve tout à droite de l' entête.

5. Conception de base de données

Maintenant que les différents rôles et l'organisation du site ont été présentés, je vais vous parler de la conception de base de données.

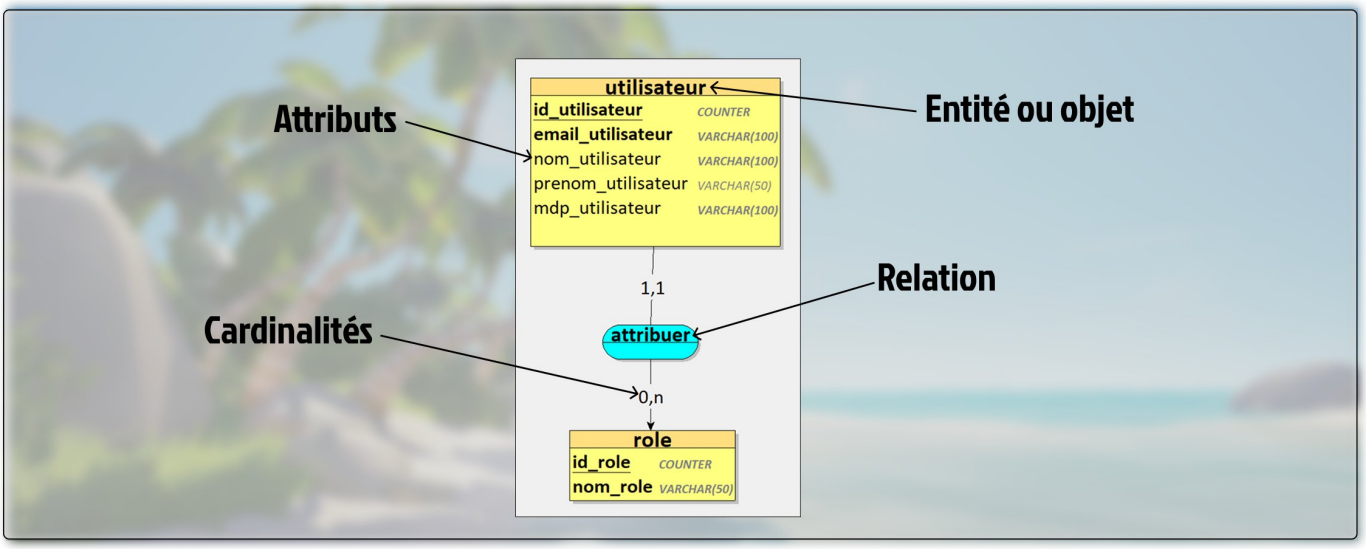
La conception de base de données a pour but de structurer, définir, organiser et modéliser les données du système d'information. Cela permet d'avoir une vue d'ensemble sur les relations entre les éléments du site et l'utilisateur. Pour faire un peu plus simple, ce sont des graphiques qui montrent, sous forme de rectangles, des sections et leur contenu, et expliquent comment toutes ces informations fonctionnent et interagissent entre elles. Il existe plusieurs étapes dans la conception de base de données mais nous allons nous attarder sur deux d'entre elles.

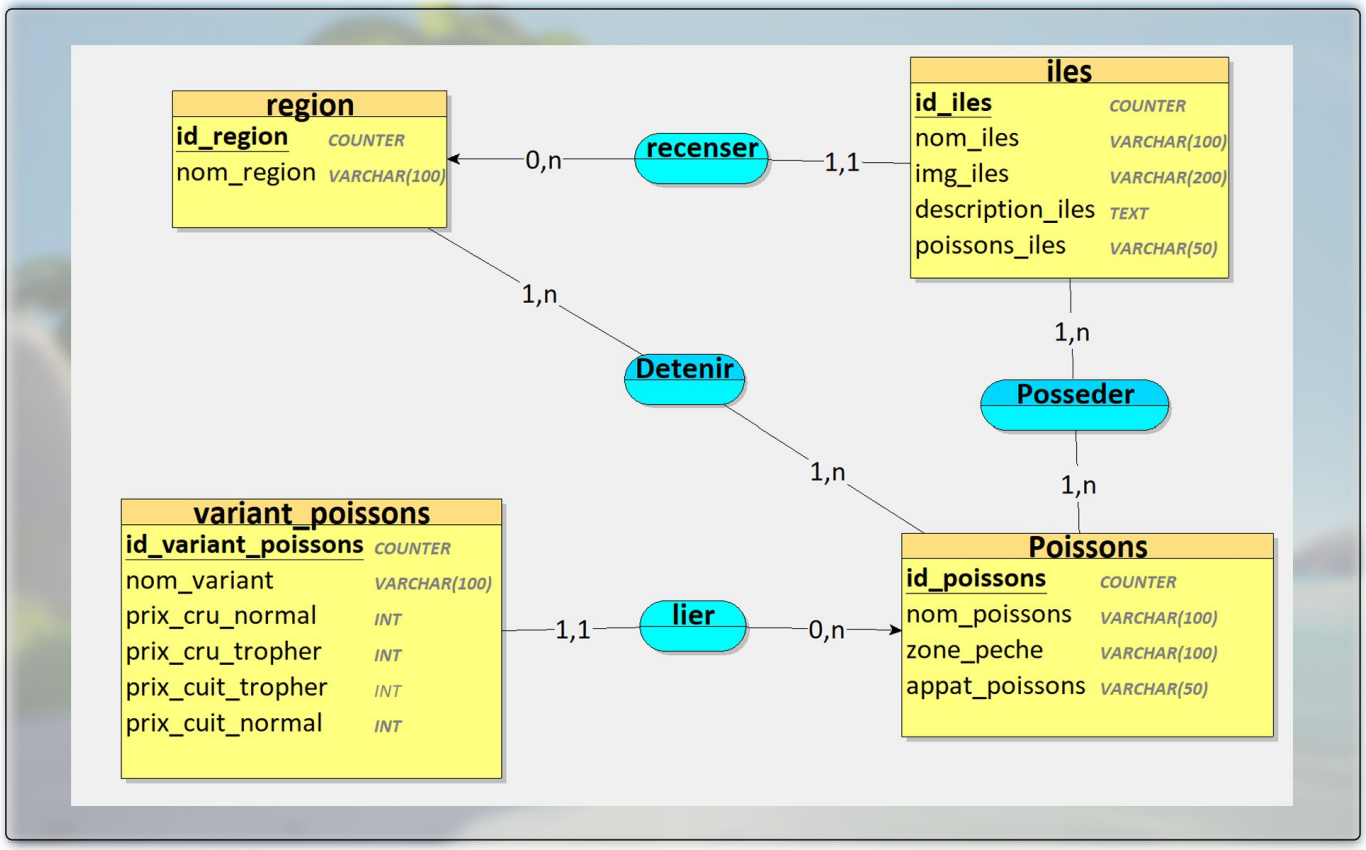
5.1 Modèle conceptuel de données

Le **modèle conceptuel de données (MCD**) est l'un des graphiques qui permet la conception de bases de données.

Le MCD permet d'illustrer les données du système d'information, il met en avant plusieurs éléments :

- Les **entités ou objets** : une entité représente un objet, comme ici l'utilisateur de mon site. Son but va être de créer des utilisateurs uniques à chaque création de compte.
- **Les attributs** : les attributs sont les propriétés ou caractéristiques de notre objet. Par exemple, pour notre utilisateur, les attributs vont être les email_utilisateur / nom_utilisateur / prénom_utilisateur / mdp_utilisateur. Tous ces attributs vont être assignés à notre entité ou objet.
- **Les relations** : les relations ont pour but de montrer les associations entre les entités car elles sont parfois amenées à communiquer entre elles. Dans le cas de notre utilisateur, il est en relation avec l'entité « rôle », les relations sont représentées avec un verbe à l'infinitif en général.
- **Les cardinalités** : elles ont pour but de démontrer le nombre d'occurrences entre les entités. Dans notre cas, l'utilisateur a obligatoirement un « rôle » mais ne peut en avoir qu'un seul, ce qui nous donne une cardinalité (1,1), alors que l'entité rôle n'a pas obligatoirement un utilisateur mais par contre un rôle peut avoir plusieurs utilisateurs, ce qui nous donne une cardinalité (0,n)





Vient ensuite la partie principale du site.

L'entité « région » recense les îles et elles ont une relation avec une cardinalité (0,n) côté région car une région peut posséder une ou plusieurs îles. Du côté des îles c'est une relation avec une cardinalité (1,1) car une île ne peut posséder qu'une région obligatoirement. Nous avons donc ici une relation type 1.

Les « îles » possèdent des poissons et ont une relation avec cardinalité (1,n) car une île possède obligatoirement des poissons et peut en avoir plusieurs différents. Et du côté poissons, il y a aussi obligatoirement un ou plusieurs poissons pour plusieurs îles, soit une cardinalité (1,n). Nous avons cette fois une relation de type n.

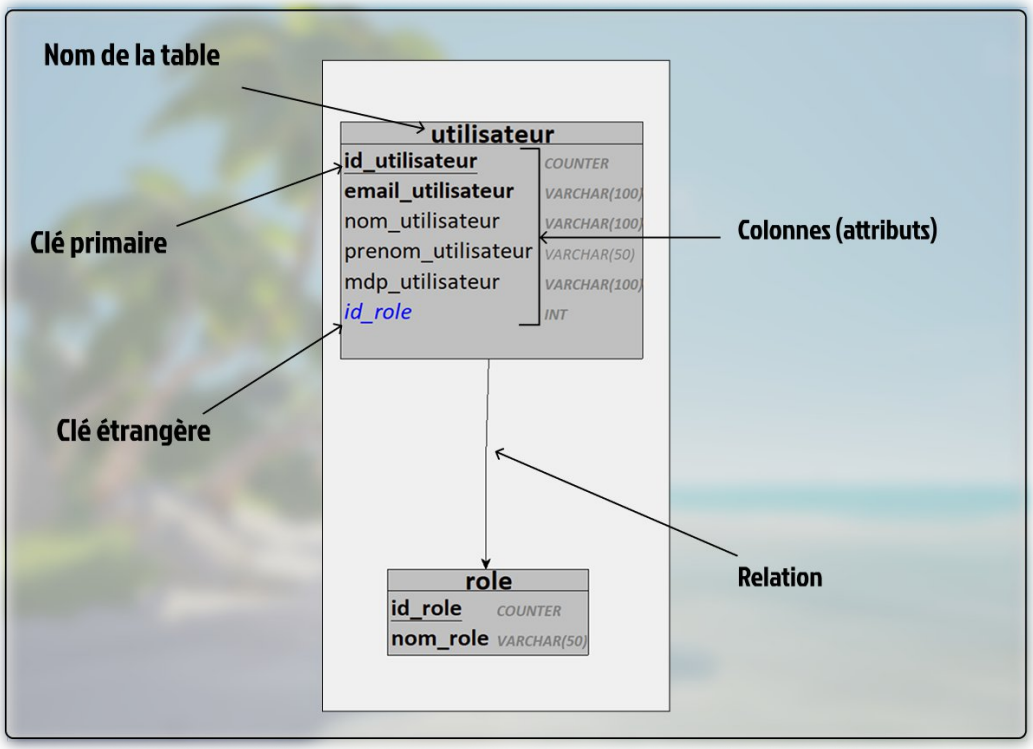
L'entité « Poissons » est également en relation avec l'entité régions. Nous avons donc une cardinalité (1,n) côté poisson, car un poisson est obligatoirement lié à une ou plusieurs régions. Quant à la région nous avons aussi une cardinalité (1,n) car une région a obligatoirement un ou plusieurs poissons. Ici, nous avons également une relation de type n.

Pour finir, les poissons sont liés au variant car un poisson peut avoir un ou plusieurs variants (0, n). Mais du côté des variants, il est obligatoire qu'un variant soit lié à un poisson (1,1). Pour ce cas, nous avons une relation type 1.

5.2 Modèle logique de données

Le modèle logique de données ou **MLD** fait suite au **MCD**.

Le **MLD** est une vue destinée pour l'implantation en base de données. Il est représenté par des tables, des colonnes, des clés primaires et des clés étrangères. Je vais vous expliquer son fonctionnement pour une meilleure compréhension.



Le premier élément est **la table**. Elle se nomme ici « utilisateur ». Ici, c'est la table utilisateur. Elle possède **une colonne**, ce sont les attributs du MCD.

Dans la colonne, on peut lire «id_utilisateur ». Il a le rôle de **clef primaire**, c'est a dire qu'elle a pour objectif d'être un identifiant unique en base de données. En effet la table utilisateur va créer des utilisateurs avec tous les attributs dans la colonne, qui ont donc tous un identifiant unique (1,2,3..). Ces identifiants sont rattachés à toutes les informations (exemple : id_utilisateur 1 / nom_utilisateur Ciccarella / prénom_utilisateur Thomas / etc.)

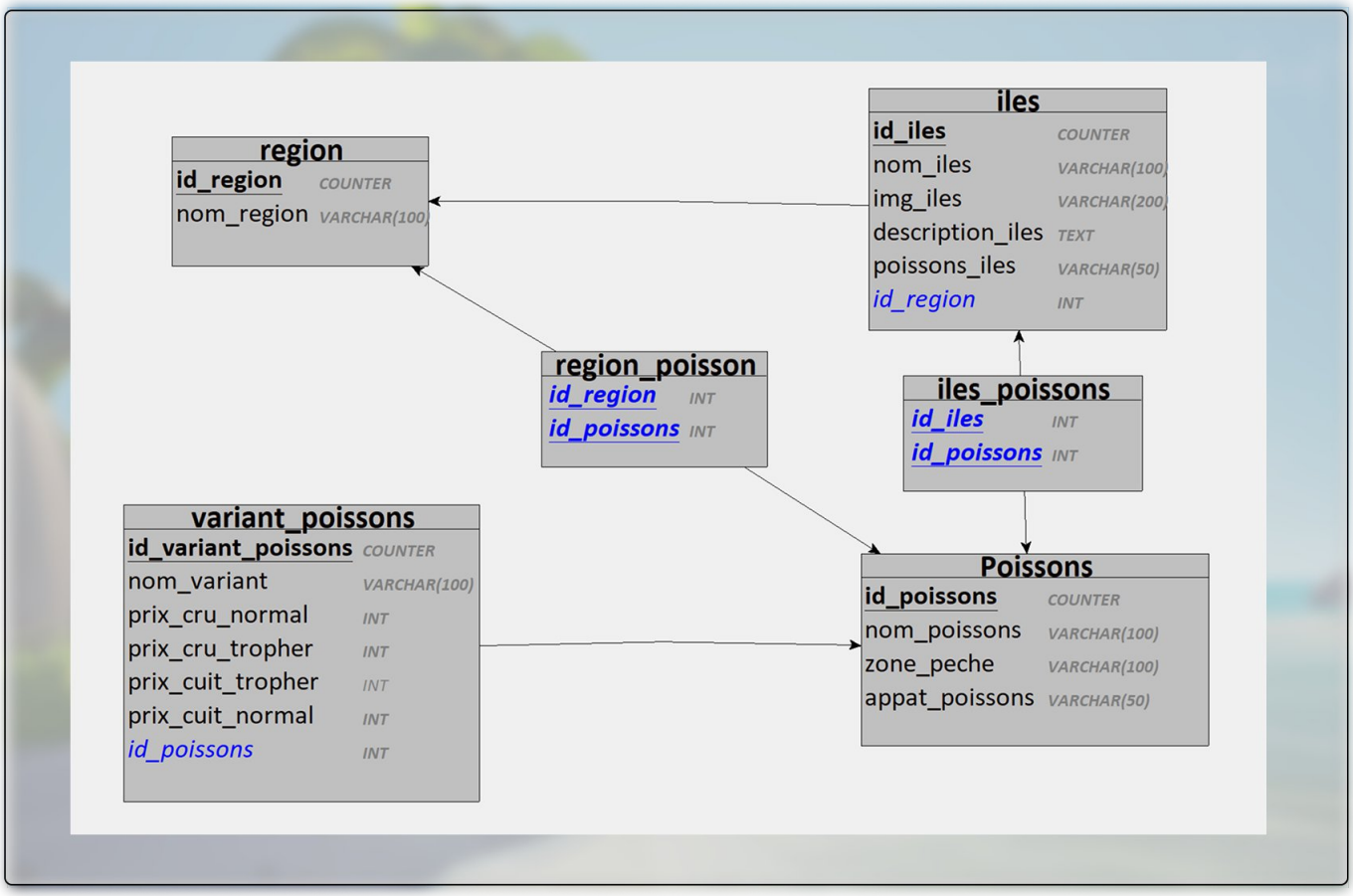
Vous pouvez aussi voir en bas de la table « id_role », qui a le rôle de clef étrangère. Cette dernière a pour but de renseigner la clef primaire d'une autre table pour faire le lien entre ces tables. Par exemple, ici ,« id_role » amène les colonnes de la table « role » pour l'utilisateur. En d'autres termes, grâce à la clef étrangère, l'utilisateur peut récupérer les informations de la table « role » et en bénéficier. L'utilisateur aura donc un rôle.

De plus, vous pouvez voir la flèche de relation entre les 2 tables où l'utilisateur pointe vers la table rôle car il vient récupérer les colonnes de cette dernière.

A droite de chaque colonne se trouvent des informations comme **VARCHAR(50) / COUNTER / INT**. Ce sont des informations de typage. Elles renseignent sur ce que vont et peuvent contenir les colonnes. Par exemple, VARCHAR veut dire **variable caractère**, la colonne va donc contenir une chaîne de caractères et le (50) signifie le nombre maximum de caractères possibles et autorisés dans cette colonne.

Il existe donc plusieurs types comme **INT (integer), TEXT (texte), DATE (date)**. Ils sont liés directement à la base de données, ils permettent de savoir quel type va être stocké en base de données. Cela permet lors de la création de la BDD de bien construire chaque table et colonne.

Je viens donc de vous expliquer le principe du MLD et dans le même temps de vous montrer le fonctionnement de mes utilisateurs sur mon site. Maintenant, je vais donc vous présenter, comme pour le MCD, la partie principale de mon site.



Vous retrouvez donc la table « **region** » avec sa clef primaire **« id_region »** et le **« nom_region »**. Cette table est récupérée par la table Îles.

La table « **iles** » avec dans sa colonnes la clef primaire **« id_îles »** possède un nom / les URL des images / une description /des poissons , et un lien avec la table « Region » grâce à la **clef étrangère « id_region »**.

Nous voilà dans un cas que je ne vous ai pas encore présenté, la table **« iles_poissons »** est une **table d'association**. Elle se crée car nous nous retrouvons dans une situation de **relation type n** ou une îles peut posséder plusieurs poissons et un poisson peut posséder plusieurs îles. Dans cette situation une table de liaison est nécessaire pour faire le lien entre ces deux tables. La **table d'association** possède donc les **deux clefs étrangères des deux tables en relation** et ainsi elle fait le lien.

Nous avons la même situation pour la table **« region_poissons ».**

Nous finissons donc avec la table **variant_poissons** qui a pour **clef primaire « id_variant_poissons »**. Elle possède les prix des poissons normaux crus et cuits et le prix des poissons trophées crus et cuits, suivi de la **clef étrangère « id_poissons »** car elle est en lien direct avec la table **poissons** puisque chaque poisson possède plusieurs variants.

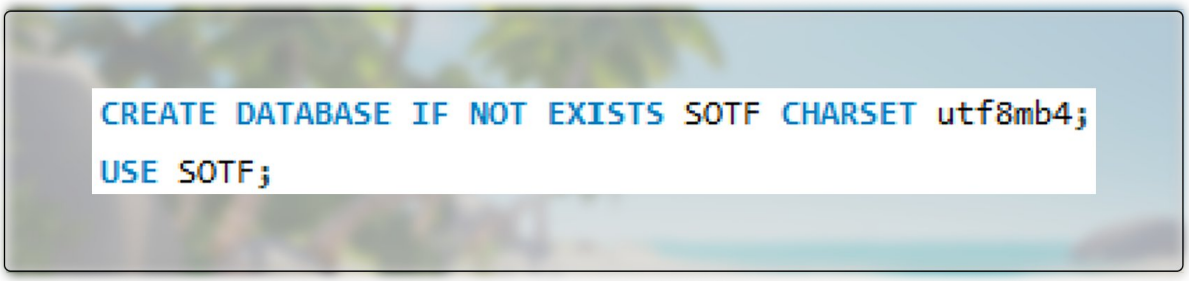
5.3 SQL

Quand le **MCD** et **MLD** sont finis, nous allons pouvoir utiliser le MLD pour créer le code SQL.

SQL est un langage de programmation qui sert à créer des bases de données avec des relations. Il permet aussi de les gérer et de les modifier. J'ai donc utilisé ce langage pour créer la base de données relationnelle de mon site.

Je me suis servie de **WORKBENCH** pour mettre en application ce que j'ai mis en place sur mon MLD.

Pour commencer j'ai déjà créé la base de données. Ici nous l'appellerons **SOTF**.

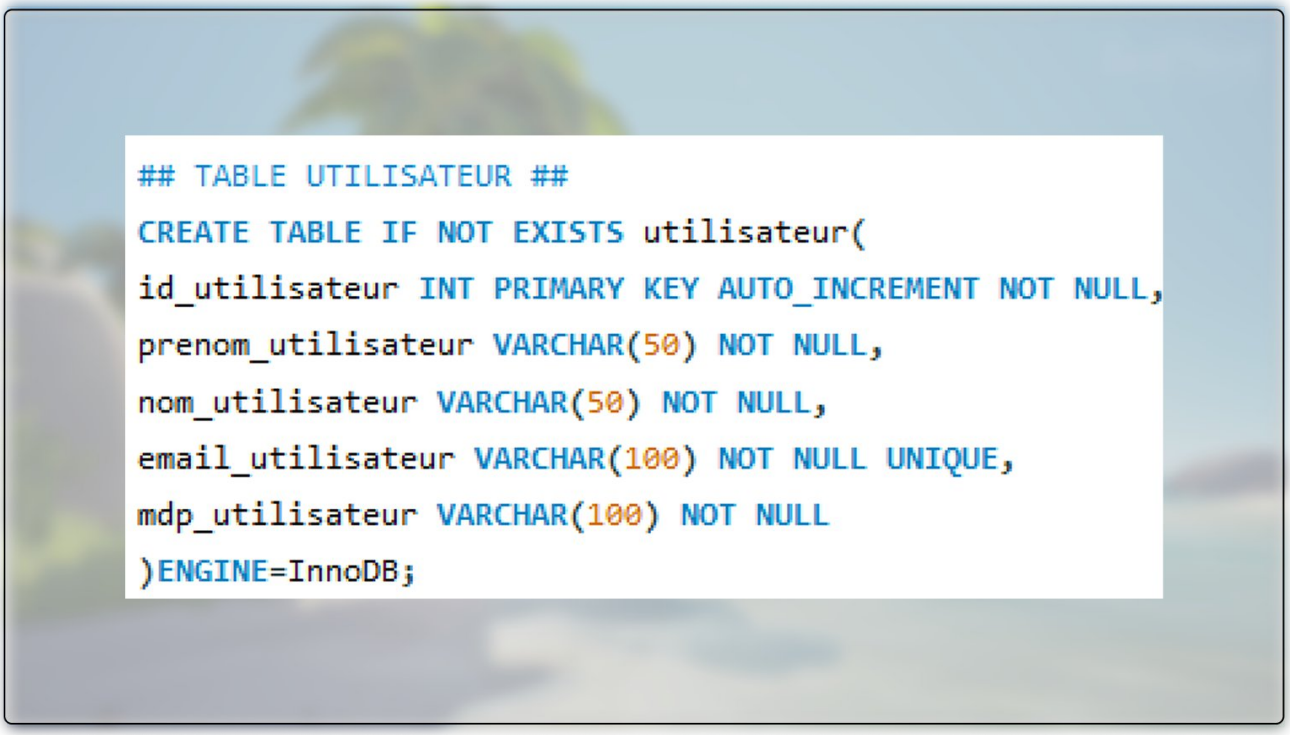


```
CREATE DATABASE IF NOT EXISTS SOTF CHARSET utf8mb4;
USE SOTF;
```

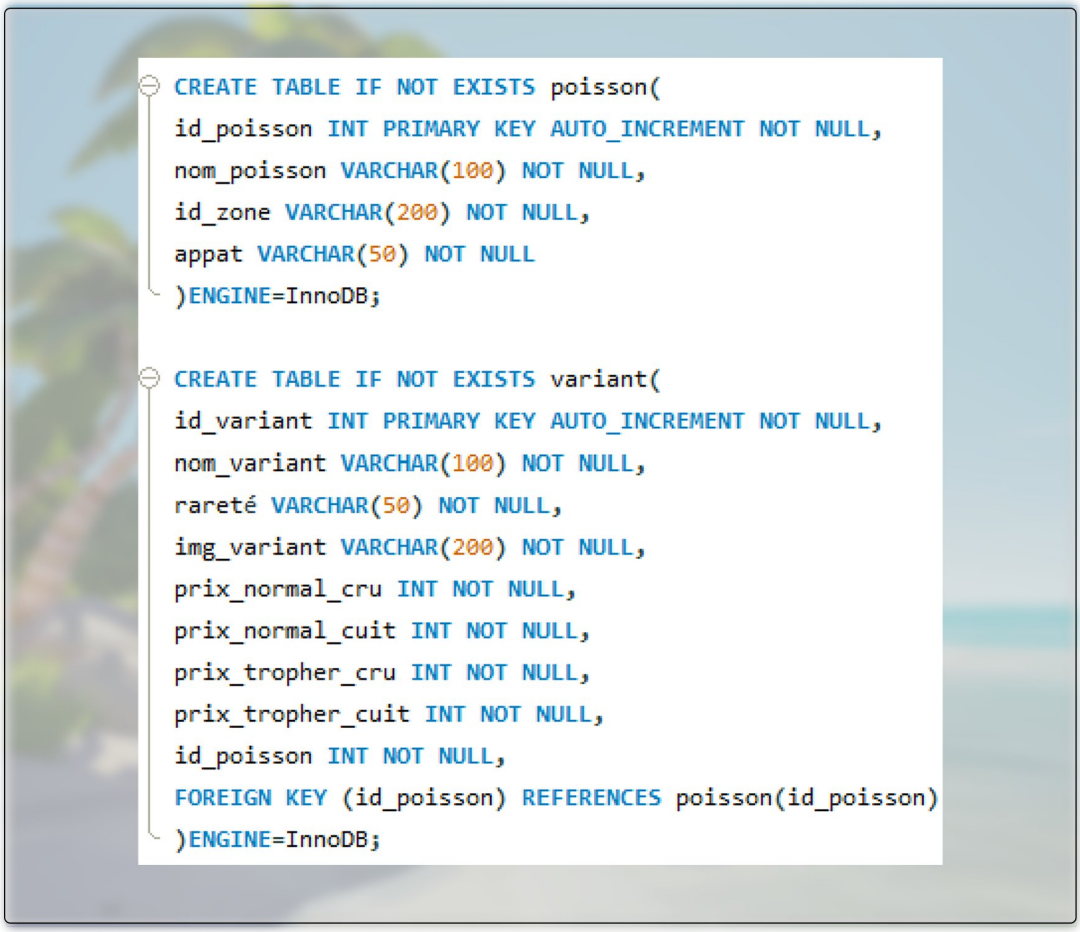
Pour reprendre le MLD, nous allons créer ensuite la table Utilisateur. Pour rappel, elle contient une **clef primaire id_utilisateur**, mais avec SQL, nous allons lui ajouter **AUTO_INCREMENT** pour qu'à chaque fois qu'un utilisateur est créé, la clef primaire change pour qu'il n'y en ait pas deux identiques.

On retrouve ensuite les autres colonnes avec le **VARCHAR**, mais aussi le **NOT NULL**, qui signifie que, lors de la création d'un utilisateur, on ne peut pas laisser ce champ vide : il doit être obligatoirement rempli.

Vous pouvez voir aussi à la fin de la table **ENGINE=InnoDB** : ENGINE sert à indiquer quel moteur de stockage sera utilisé. Il sert à gérer la manière dont seront stockées les données et comment elles seront manipulées.



```
## TABLE UTILISATEUR ##
CREATE TABLE IF NOT EXISTS utilisateur(
id_utilisateur INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
prenom_utilisateur VARCHAR(50) NOT NULL,
nom_utilisateur VARCHAR(50) NOT NULL,
email_utilisateur VARCHAR(100) NOT NULL UNIQUE,
mdp_utilisateur VARCHAR(100) NOT NULL
)ENGINE=InnoDB;
```



```sql
CREATE TABLE IF NOT EXISTS poisson(
id_poisson INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
nom_poisson VARCHAR(100) NOT NULL,
id_zone VARCHAR(200) NOT NULL,
appat VARCHAR(50) NOT NULL
)ENGINE=InnoDB;

CREATE TABLE IF NOT EXISTS variant(
id_variant INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
nom_variant VARCHAR(100) NOT NULL,
rareté VARCHAR(50) NOT NULL,
img_variant VARCHAR(200) NOT NULL,
prix_normal_cru INT NOT NULL,
prix_normal_cuit INT NOT NULL,
prix_tropher_cru INT NOT NULL,
prix_tropher_cuit INT NOT NULL,
id_poisson INT NOT NULL,
FOREIGN KEY (id_poisson) REFERENCES poisson(id_poisson)
)ENGINE=InnoDB;
```

J'ai ensuite créé les **tables « poisson »** et **« variant »** dans le même fonctionnement que la table précédente. Cependant, une nouvelle spécificité s'ajoute dans la table « variant ».

La **FOREIGN KEY (clef étrangère)** elle renseigne la clef étrangère avec laquelle la table est en relation. Dans l'exemple, la clé étrangère est id_poisson et il référence l'id de la table poisson. Elle permet donc de lier la table variant à la table poisson, comme ça depuis un variant, on peut savoir quel poisson est concerné en le retrouvant via son ID.

Voici maintenant le visuel partiel des tables « poisson » et « variant » fini en base de données.

id_poisson	nom_poisson	id_zone	appat
1	Branchies Belliqueses	Proche des Forts	Larves
2	Déferlantes	Région " The Wilds "	Vers De Terre
3	Écailles Anciennes	Région " The Ancient Isles "	Sangsues
4	Écumeurs des îles	Certaines îles	Aucun
5	Houles	Eau Douce	Aucun
6	Nageoires d'abondance	Région " The Shores Of Plenty "	Vers de Terre
7	Naufrageurs	Proche des épaves	Vers de Terre
8	Poissons Diable	Région "The Devil's Roar"	Larve
9	Queues éclaboussantes	Partout	Aucun
10	Tempêtes de Mers	Dans les Tempêtes	Sangsues

id_variant	nom_variant	rareté	img_variant	prix_normal_cru	prix_normal_cuit	prix_tropher_cru	prix_tropher_cuit	id_poisson
1	Jade	Très Commune	./public/image/img_poisson...	563	848	1410	2115	1
2	Cieux	commune	./public/image/img_poisson...	675	1013	1688	2535	1
3	Rhum	Rare	./public/image/img_poisson...	788	1185	1973	2963	1
4	Sablonneuse	Très Rare	./public/image/img_poisson...	5625	8438	14063	21098	1
5	Aigre-Douce	Nocturne	./public/image/img_poisson...	675	1013	1688	2535	1
6	Mordorée	Très Commune	./public/image/img_poisson...	338	510	848	1275	2
7	Sablonneuse	Commune	./public/image/img_poisson...	450	675	1125	1688	2
8	Océanique	Rare	./public/image/img_poisson...	563	848	1410	2115	2
9	Vaseuse	Très Rare	./public/image/img_poisson...	4500	6750	11250	1688	2
10	Corallienne	Nocturne	./public/image/img_poisson...	450	675	1125	1688	2
11	Amande	Très Commune	./public/image/img_poisson...	338	510	848	1275	3
12	Saphire	Commune	./public/image/img_poisson...	450	675	1125	1688	3
13	Fumée	Rare	./public/image/img_poisson...	563	848	1410	2115	3
14	Osseuse	Très Rare	./public/image/img_poisson...	4500	6750	11250	1688	3
15	Étoilée	Nocturne	./public/image/img_poisson...	450	675	1125	1688	3

6. Organisation et mise en place du code

Je vais maintenant pouvoir vous parler de la partie code de mon site web. Quel langage j'ai utilisé, quelle structure et quelle organisation pour concevoir ce projet.

6.1 Les langages utilisés

Pour ce site web j'ai utilisé plusieurs langages. Pour commencer, je vais vous parlé d'un langage dont je vous ai déjà parler pour la base de données le langage SQL avec Workbench.

Pour ce qui est du langage principal, j'ai utiliser PHP, du HTML, du CSS pour le style et du JavaScript pour le côté dynamique.

6.2 Architecture MVC

Pour donner du contexte, je vais d'abord vous expliquer quel est le but d'une architecture dans le code.

L'architecture, c'est la façon dont va fonctionner le code et la manière dont il va être organisé, les relations entre les différents éléments du code et la manière dont ces derniers communiquent entre eux. Tout cela relève de l'architecture.

Pour ma part j'ai opté pour une architecture MVC (modèle - vue - contrôleur). Comme son nom l'indique, le MVC est composé de 3 éléments.

- **Le modèle :** comme dit plus haut, j'ai décidé d'utiliser le langage PHP pour mon projet. Ce langage permet de faire ce que l'on appelle de la **programmation orientée objet**. Je parlerai de **POO** pour la suite. Pour faire de la POO, nous utilisons des classes comme dit plus haut avec le MCD – MLD. Le modèle a pour but de stocker ces classes et tout ce qu'elles contiennent. Pour résumer, le modèle a pour rôle de gérer les données et la logique du code.
- **Vue :** la vue du MVC, comme son nom l'indique, va avoir pour mission l'affichage du code pour l'utilisateur. On y mettra donc tous les éléments HTML, avec des éléments de PHP pour afficher les données et appliquer la logique du modèle.
- **Contrôleur :** pour terminer, nous avons le Contrôleur. Son rôle est crucial dans le MVC car c'est lui qui va faire le lien entre le modèle et la vue. C'est à partir du contrôleur que l'on va gérer les affichages et les relations entre les éléments de notre code.

Pour résumer, quand nous travaillons avec une architecture en MVC, le but premier est de sectoriser notre code afin d'avoir une meilleure structure et organisation, ainsi meilleure maniabilité, tout en ayant une possibilité d'évolution dans le temps pour implanter de nouveaux éléments au site.

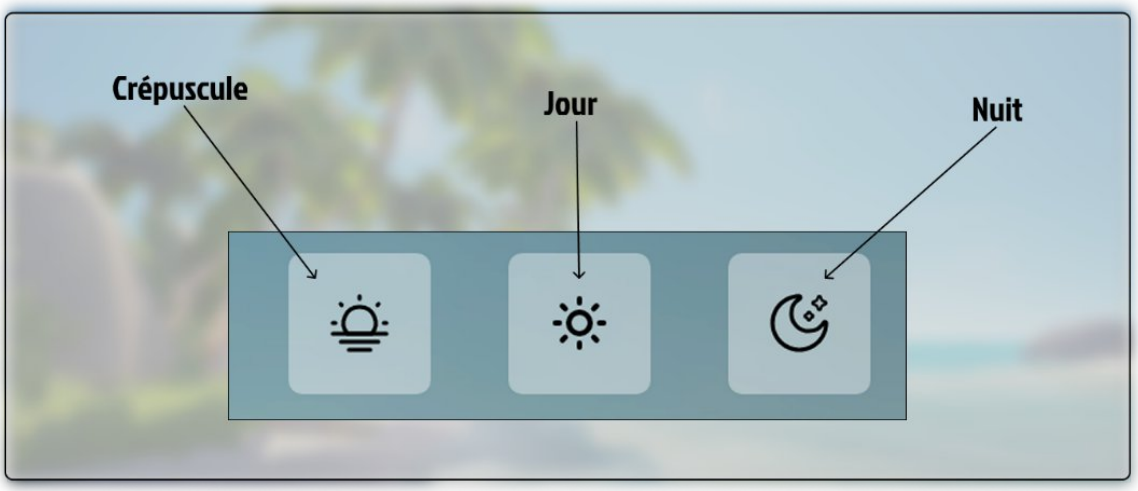
7. Front - end

7.1 Mises en contexte

Pour la partie front j'ai décidé de créer une fonctionnalité qui gère le background du site de manière dynamique. Je voulais que le background de mon site soit légèrement animé pour donner plus de vie.

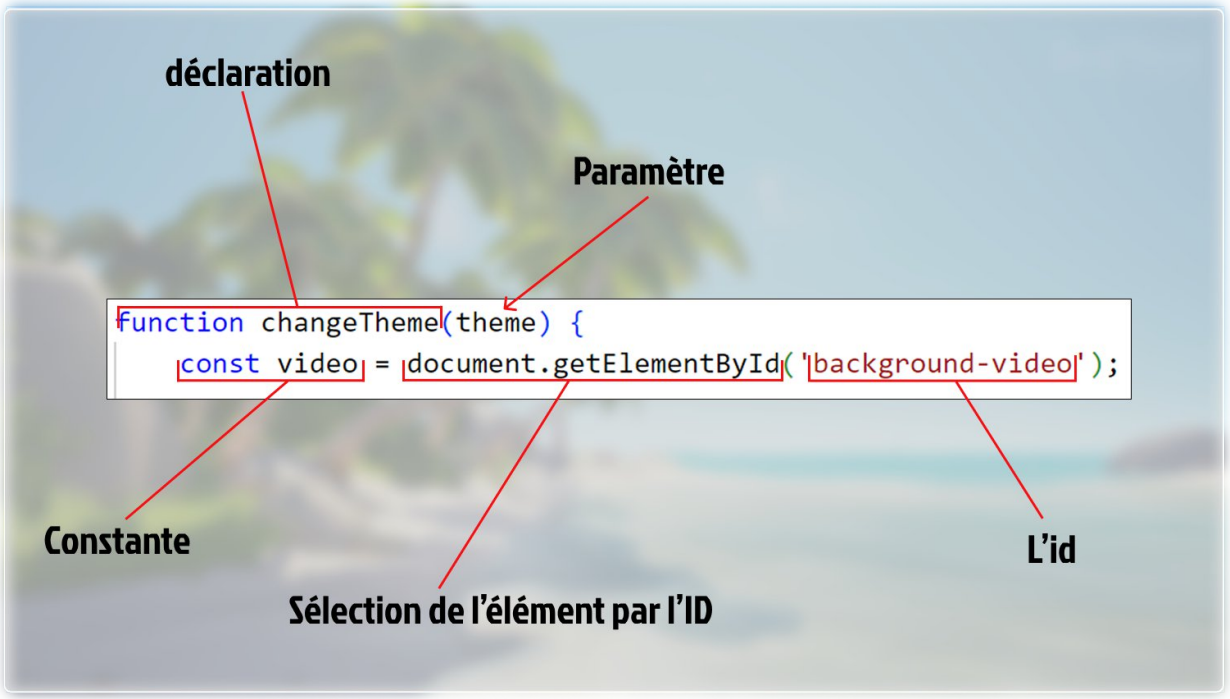
Je voulais également qu'il soit adaptable pour l'utilisateur et surtout que si l'utilisateur fait un choix, qu'il reste sauvegardé même après avoir fermé le site. Il y aura donc 3 thèmes pour les backgrounds, : nuit, jour, crépuscule.

Le thème est sélectionnable via 3 boutons en haut à droite de la page. Vous pouvez donc choisir votre thème et il reste enregistré pour vos prochaines visites sur le site. Chaque bouton a un symbole pour indiquer quel thème va être activé.



7.2 Création de la fonction

Pour commencer il faut que l'on crée la fonction que nous allons utiliser pour changer le thème du background de façon dynamique. pour cela nous allons utiliser le langage JavaScript.



Nous allons effectuer plusieurs étapes. Tout d'abord, il faut faire la déclaration de la fonction et qu'elle contienne un paramètre, le paramètre servira à déterminer quelle vidéo de fond sera utilisée.

La fonction a pour but de récupérer le bon élément vidéo pour l'afficher en background. pour ce faire nous allons créons une **constante « vidéo »** qui récupère l'élément vidéo dans la page HTML en le trouvant grâce à son ID qui est **« background-vidéo »**. Tout sera stocké dans la constante « vidéo ». J'ai opté pour une constante et non une variable car la constante ne peut être modifiée alors que la variable si.

Pour pouvoir gérer les thèmes, j'ai ajouté des conditions (if / else if / else) pour déterminer quel thème est sélectionné. Dans les conditions que j'ai mis en place, il faut que le thème soit strictement égal à la chaîne de caractères portant le nom du thème. Cette condition est visible grâce au triple égale.

Vient ensuite la gestion d'enregistrement en local du thème sélectionné grâce à **« localeStorage.setItem »**. Grâce à cela, le thème est stocké dans la clef « thème » qui est mise en paramètre de notre fonction. Ce qui permet au thème de rester enregistré dans le stockage du site web pour qu'à chaque visite sur le site le thème choisi reste le même.

Dans notre fonction, la seule chose qui change, c'est l'élément vidéo. Nous changeons la source de la vidéo afin qu'elle corresponde au thème sélectionné. Donc, pour chaque condition, une vidéo a donc une source différente grâce à **« video.src »** qui renseignera quelle vidéo sera affichée.

Nous avons également une condition **« else »** qui a pour mission de prendre le relais si aucune condition **« if »** n'est réalisable, elle affiche un message d'erreur dans la console pour nous avertir du problème rencontré.

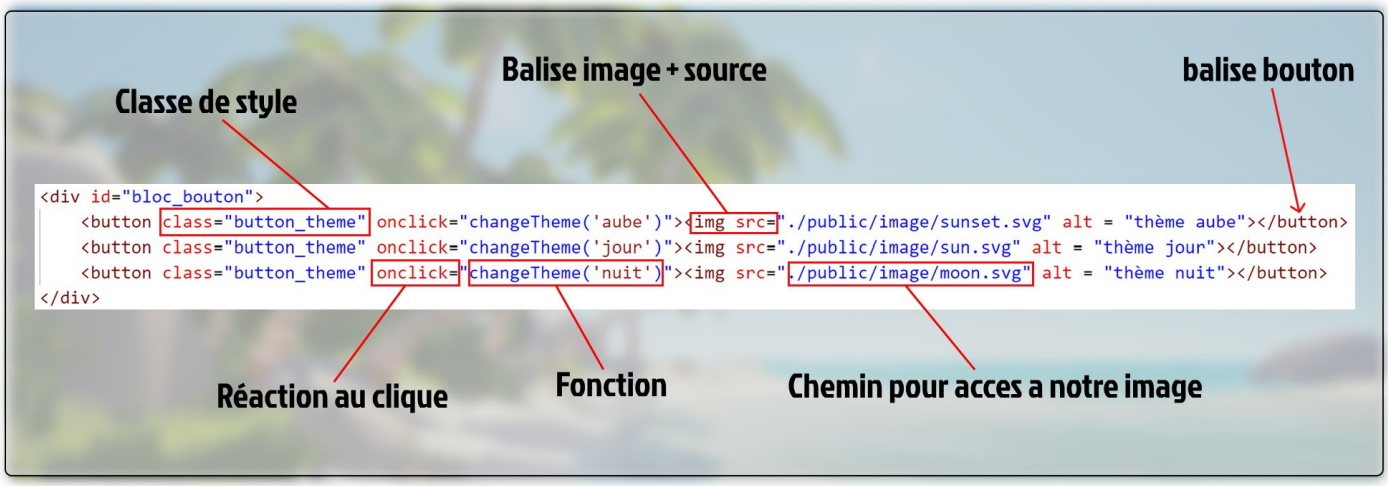
7.3 Application dans la vue

Nous venons de créer la fonction **« changeTheme »**. Il faut maintenant l'utiliser et pour cela nous allons créer des boutons sur notre page pour pouvoir choisir et changer le thème de notre background.

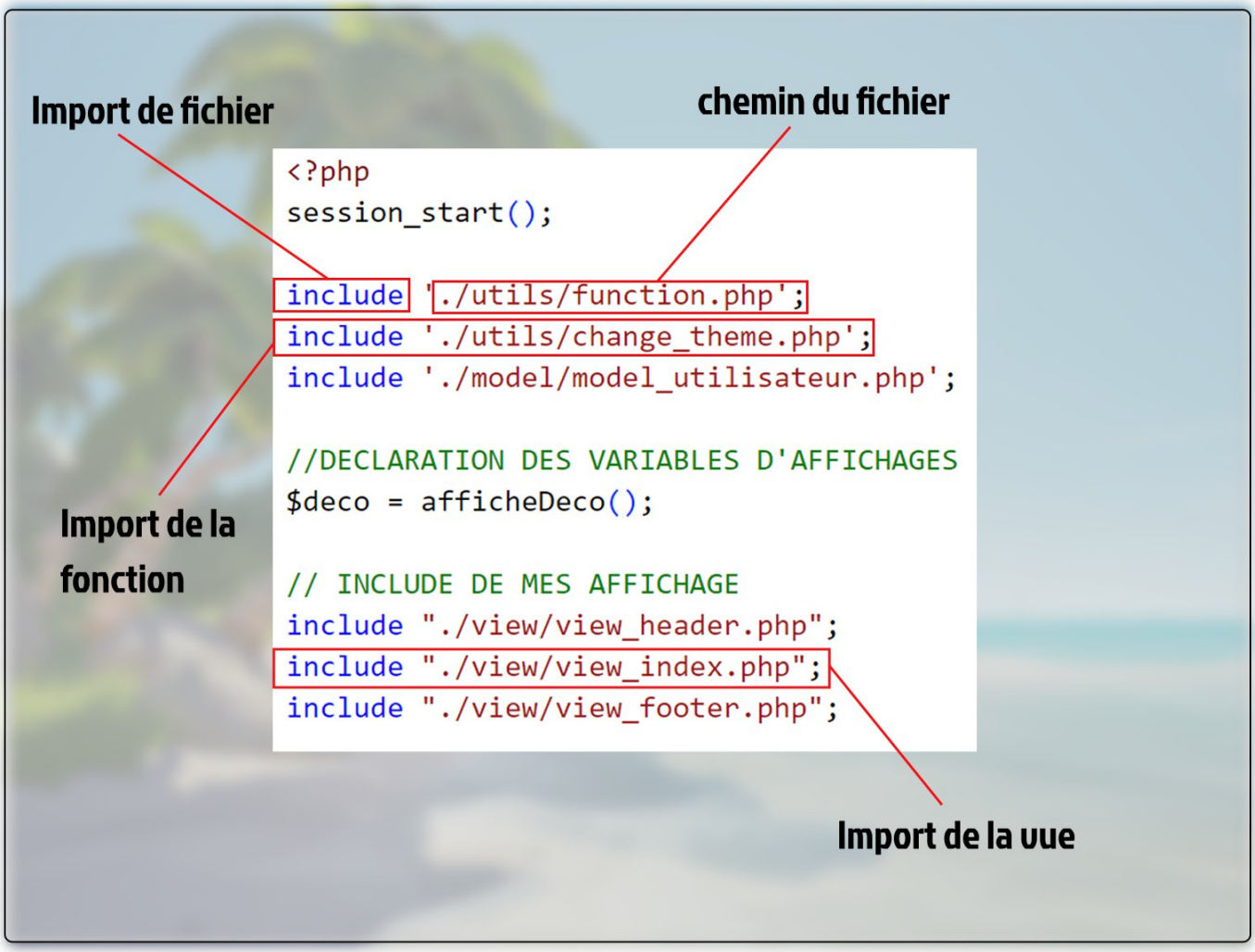
Pour créer nos boutons, nous allons donc nous diriger sur la partie **Vue** de notre **MVC** et utiliser des balises HTML **« Button »**. On leur assigne une classe de style qui permet de modifier le visuel de nos boutons dans le langage «CSS» pour que l'utilisateur du site comprenne quel bouton correspond à quel thème.

Le but de nos boutons est qu'ils réagissent et appliquent notre fonction quand on clique dessus. Alors, nous leurs attribuons **« onclick »,** fonction native dans laquelle nous assignons notre fonction "changeTheme". De cette manière, au clic sur le bouton la fonction s' exécutera et appliquera le thème.

Vous pouvez également voir la balise « img » qui correspond à une image. Il faut renseigner la source de l'image pour qu'elle s'affiche dans nos boutons



7.4 Création des liens avec le contrôleur



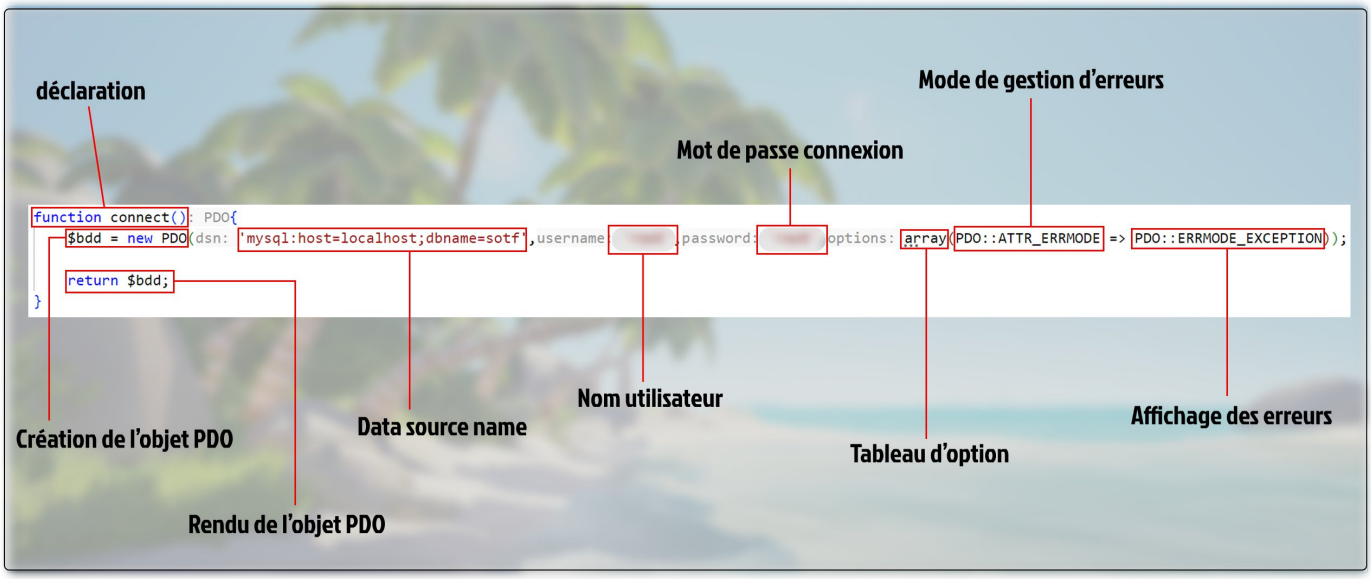
Maintenant que nous avons créé la fonction et mis en place les boutons pour l'accueillir, il faut que nous les lions grâce au contrôleur. Voici le code du contrôleur pour que la liaison se fasse. Nous utilisons des **« include »**, ils servent à appeler des fichiers dans les différentes parties du MVC. Ce qui est grandement utile avec les includes, c'est qu'ils permettent d'appeler des pages ou des fonctions dans plusieurs pages. Dans notre schéma, vous pouvez voir les includes du « header » et « footer ». Ce sont des pages que je vais pouvoir appeler pour chaque page les nécessitant. Cela évite de devoir refaire le même élément à chaque page créée.

8. Back – end

8.1 Fonction connect

Pour la fonctionnalité back-end, je vais vous présenter la création, la connexion et l'affichage d'un compte sur mon site web.

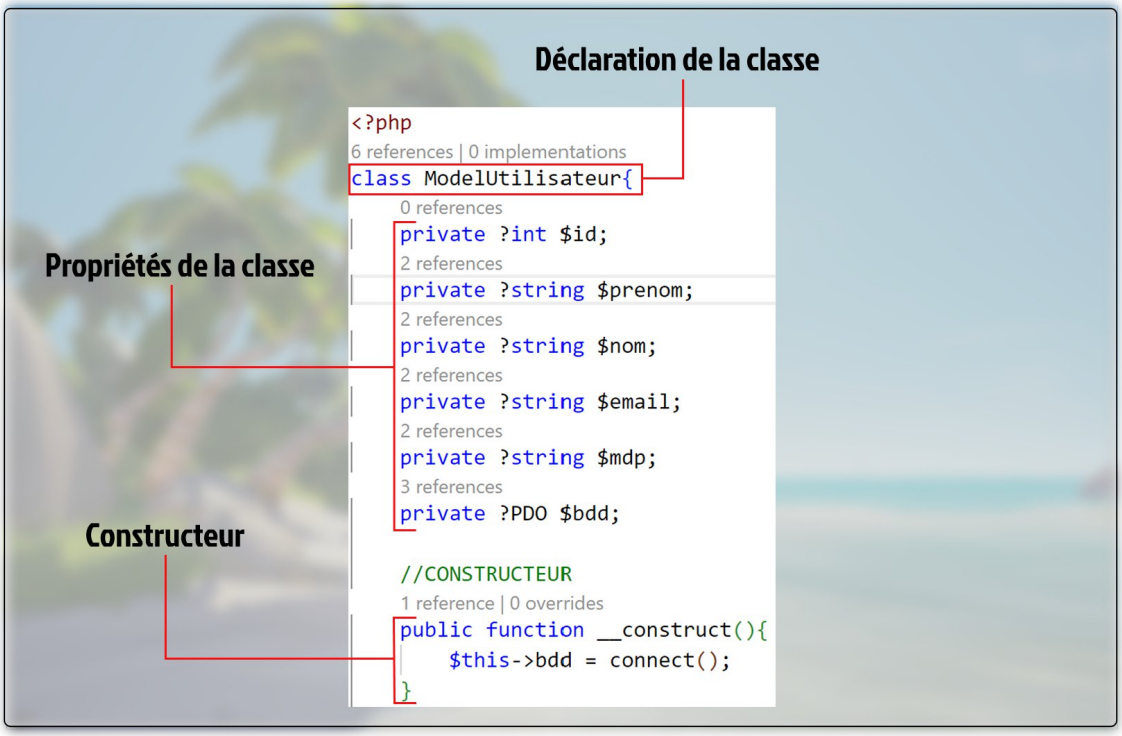
Pour commencer, je vais créer la fonction qui va permettre de lier mon site web à ma base de données.



- Je commence par la déclaration de ma fonction, elle s'appelle **« connect »**
- Vient ensuite la création de l'objet **PDO** (**PHP Data Object**) qui va avoir pour rôle de nous connecter à la base de données **MySQL**.
- Le data source name englobe plusieurs éléments :
  - **« MySQL »** qui sert à indiquer le type de base de données à laquelle nous allons nous connecter.
  - **« host=localhost »** qui indique quel type de serveur nous avons; ici c'est un serveur local.
  - **« dbname=soft »** renseigne le nom de la base de données avec laquelle nous allons interagir.
  - Suit ensuite le nom de l'**utilisateur** pour la connexion, ici il est flouté pour des raisons de sécurité.
  - **Le mot de passe** pour la connexion à la base de données lui aussi est flouté, toujours pour des raisons de sécurité.
  - **« array »** a pour mission de créer un tableau qui va stocker plusieurs options pour notre connexion à la base de données
  - **« PDO ::ATR_ERRMODE »** est la première option de notre tableau, elle permet de définir le mode de gestion des erreurs.
  - **« PDO ::ERRMODE_EXCEPTION »** est la deuxième option de notre tableau, elle permet d'afficher les erreurs pour que l'on puisse les identifier et comprendre la cause d'un éventuel problème.
  - Une fois terminé **« return $bdd »** a pour but de retourner le résultat de la connexion, grâce à cela nous passons d'une fonction qui fait une action à une fonction qui fournit une ressource réutilisable.

8.2 Création de la classe utilisateur

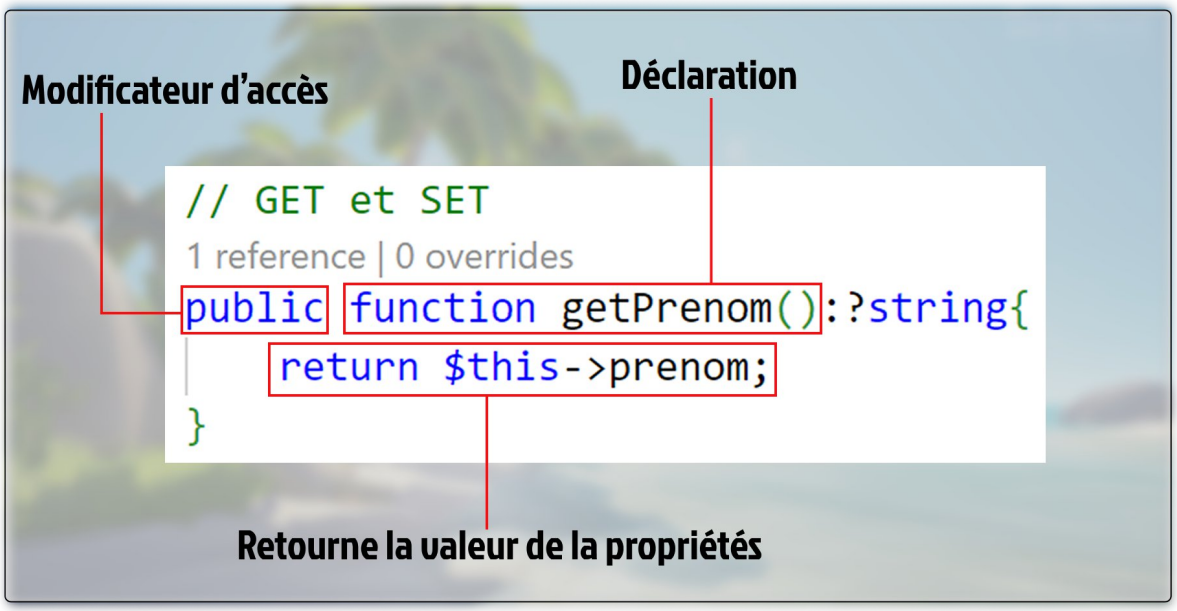
Je vais maintenant créer la classe de mon utilisateur, elle se trouve dans le modèle utilisateur car je suis en MVC.



- La première étape est de déclarer la classe. Ici elle s'appelle **« modelUtilisateur »**.
- Pour déclarer les propriétés à notre classe il faut renseigner certains éléments :
  - **« private »** signifie que la propriété ne sera accessible que depuis la classe elle-même pour des raisons de sécurité.
  - **« ? »** informe que la propriété peut être nulle, ce qui signifie qu'elle peut être aussi bien saisie que vide.
  - **« int » « string » « PDO »** sont ce que l'on appelle le typage. Ils renseignent le format attendu lors de la saisie. Dans notre cas par exemple la première ligne **« private ?int $id ; »** signifie que le format attendu est un chiffre ou nombre entier.
  - **« $id » « $prenom »** sont les noms des propriétés et les variables qui vont accueillir les données de l'utilisateur.
- **Le constructeur** a un rôle fondamental dans la création de classes et la **programmation orientée objet**, ici il va initialiser la connexion à la base de données grâce à 2 étapes :
  - Il appelle la **fonction « connect »** grâce à la **méthode magique « __construct »** .
  - Puis il stocke le résultat de l'objet PDO dans la propriété «bdd » grâce au « this », ce qui permet que la connexion soit disponible pour tous les futurs utilisateurs.

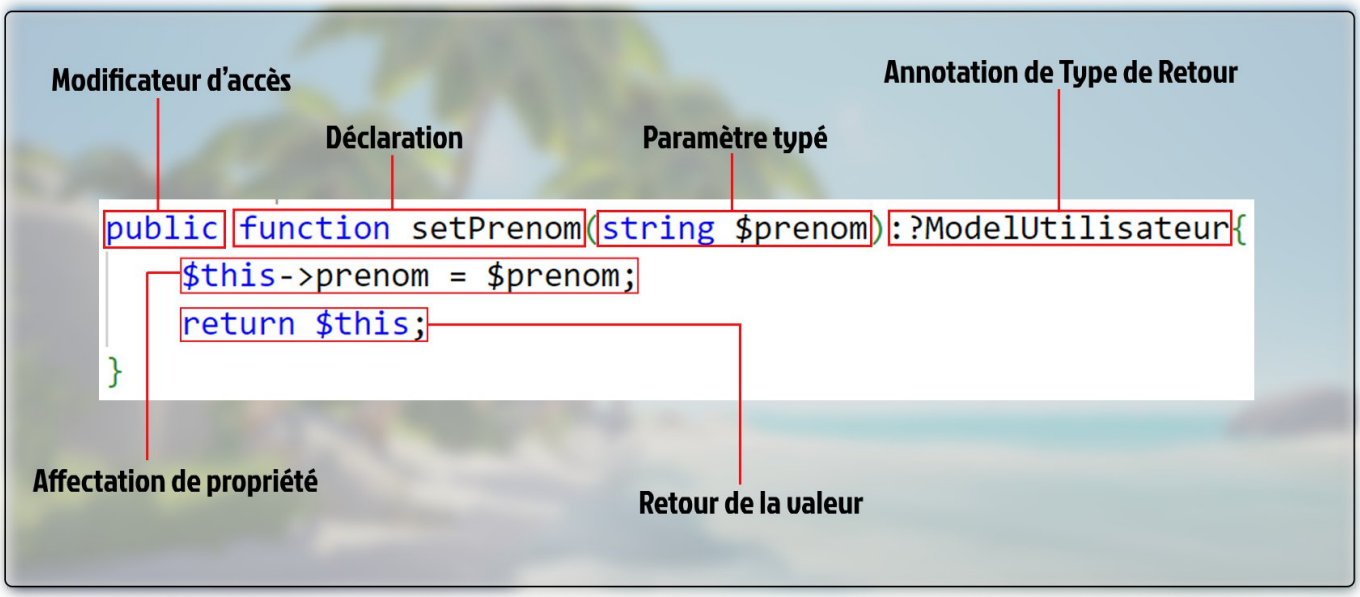
8.3 Getter et setter

Comme nous l'avons vu juste avant, dans une classe, nous avons des propriétés. Pour contrôler ces dernières (ex : les afficher, les modifier), les getters et setters ont pour rôle d'**encapsuler les données de l'objet utilisateur**. L'encapsulation sert à protéger les données d'une classe en les rendant privées pour qu'elles ne soient atteignables et modifiables que via les getters et setters. Je les ai donc mis en place dans ma classe utilisateur.



Pour mettre en place un getter, il faut le déclarer de manière publique pour que l'on puisse y avoir accès à l'extérieur de la classe, car notre propriété est privée. Pour la déclarer de manière publique, il faut mettre un modificateur d'accès **« public »**.

Le but premier d'un getter est d'accéder à la valeur de la propriété, et pour cela on demande de nous la retourner grâce au **« return $this->prenom »**. Ici nous avons l'exemple pour la propriété « prenom » mais chaque propriété a son propre getter.



Comme pour le getter, quand on déclare le setter, il faut le faire avec un modificateur d'accès pour pouvoir l'atteindre en dehors de la classe. Il faut ensuite ajouter en paramètre le typage. Ici, nous attendons une chaîne de caractères **«string »** .

L'annotation de type de retour indique que ce qui sera retourné sera **« nulle »** ou la classe « **ModelUtilisateur ».**

Vient ensuite l'affectation à la propriété, et pour terminer le setter, on retourne la valeur pour que l'on puisse l'utiliser par la suite via **« return $this »**.

8.4 Récupération des information utilisateur

Pour récupérer les informations d'un utilisateur, nous mettons en place un formulaire qui se trouve dans la vue. Ce formulaire est en HTML

```
<div id="main_profil">

    <form method ="POST" id ="inscription">
        <h1>Inscription</h1>

        <h3>Prenom </h3>
        <input type="text" name="prenom" class="inputP" placeholder="Entrez votre prenom">
        <h3>Nom </h3>
        <input type="text" name="nom" class="inputP" placeholder="Entrez votre nom">
        <h3>Adresse email </h3>
        <input type="email" name="email" class="inputP" placeholder="Entrez votre email">
        <h3>Mot de passe </h3>
        <input type="password" name="mdp" class="inputP" placeholder="Entrez un mot de passe">
        <h3>Confirmer votre mot de passe </h3>
        <input type="password" name="mdp2" class="inputP" placeholder="Confirmer votre mot de passe"> <br>

        <button type="submit" name = "inscription" class = "btnP">inscription</button>

        <h3><?php echo $message ?></h3>

    </form>
```

Tout d'abord, ce formulaire est en méthode « POST » pour pouvoir envoyer les informations à la base de données de manière sécurisée.

Pour ce qui est du formulaire en lui-même, je vous présente le formulaire d'inscription de mon site. Ici, il y a une balise « h1 » pour identifier le titre du formulaire, elle est suivi de « H3 » pour identifier chaque champ à remplir. J'ai décidé d'utiliser des « H3 » et non des labels pour mes champs par pur esthétisme, car toutes les informations importantes mais secondaires sont des «H3 » pour pouvoir créer une harmonisation des éléments sur toutes les pages.

Pour ce qui est des **« input »**, ils on tous un type « texte » pour les saisies sauf celui de l'e-mail pour s'assurer d'une bonne saisie et minimiser les erreurs de l'utilisateur.

Ils sont également tous munis d'un « name » pour pouvoir par la suite indiquer quel champ devra être récupéré pour l'envoyer en base de données.

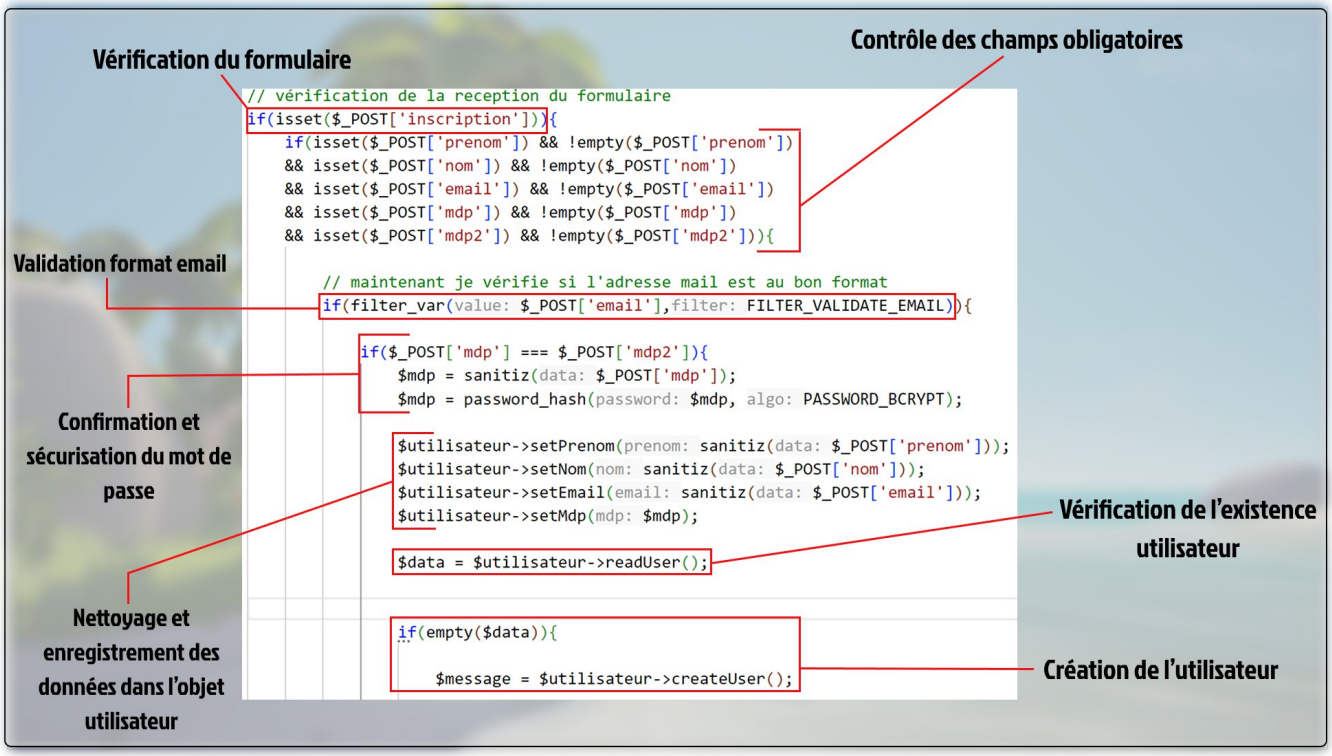
Pour le bouton d'envoi, il est de type « submit » afin qu'il ait pour rôle de déclencher l'envoi de toutes les informations du formulaire. Ce sera par ce bouton que nous pouvons vérifier si toutes les informations et saisies attendues sont correctes pour valider la création du compte utilisateur.

Le « H3 » est le renvoi des messages d'informations. Il permet d'indiquer les potentielles erreurs de saisie de l'utilisateur ou de la réussite de création de compte.

Pour cette partie, nous aurions pu améliorer certains points au niveau sécurité, comme des **« required »**, pour obliger l'utilisateur à saisir tous les champs et préciser quel champ n'a pas été rempli. De plus, un **REGEX** pourrait être utile pour interdire les mots de passe trop simples.

8.5 Contrôleur Profile

Le contrôleur va avoir pour rôle d'effectuer plusieurs vérifications et sécurisations des données envoyées pour que l'enregistrement de ces dernières se fasse en toute sécurité et sans erreurs.



Pour commencer, nous allons mettre en place une condition « if » pour vérifier si le formulaire a été envoyé en passant par le bouton «inscription». Si la condition est bien remplie, nous allons pouvoir passer aux autres vérifications.

Nous allons ensuite vérifier si tous les champs du formulaire sont existants et bien remplis grâce à la méthode **«$_POST »**. Chaque champ sera vérifié grâce au **« name »** que nous lui avons attribué.

Une fois tous les champs vérifiés, l'e-mail lui aussi sera contrôler pour confirmer le bon format de ce dernier.

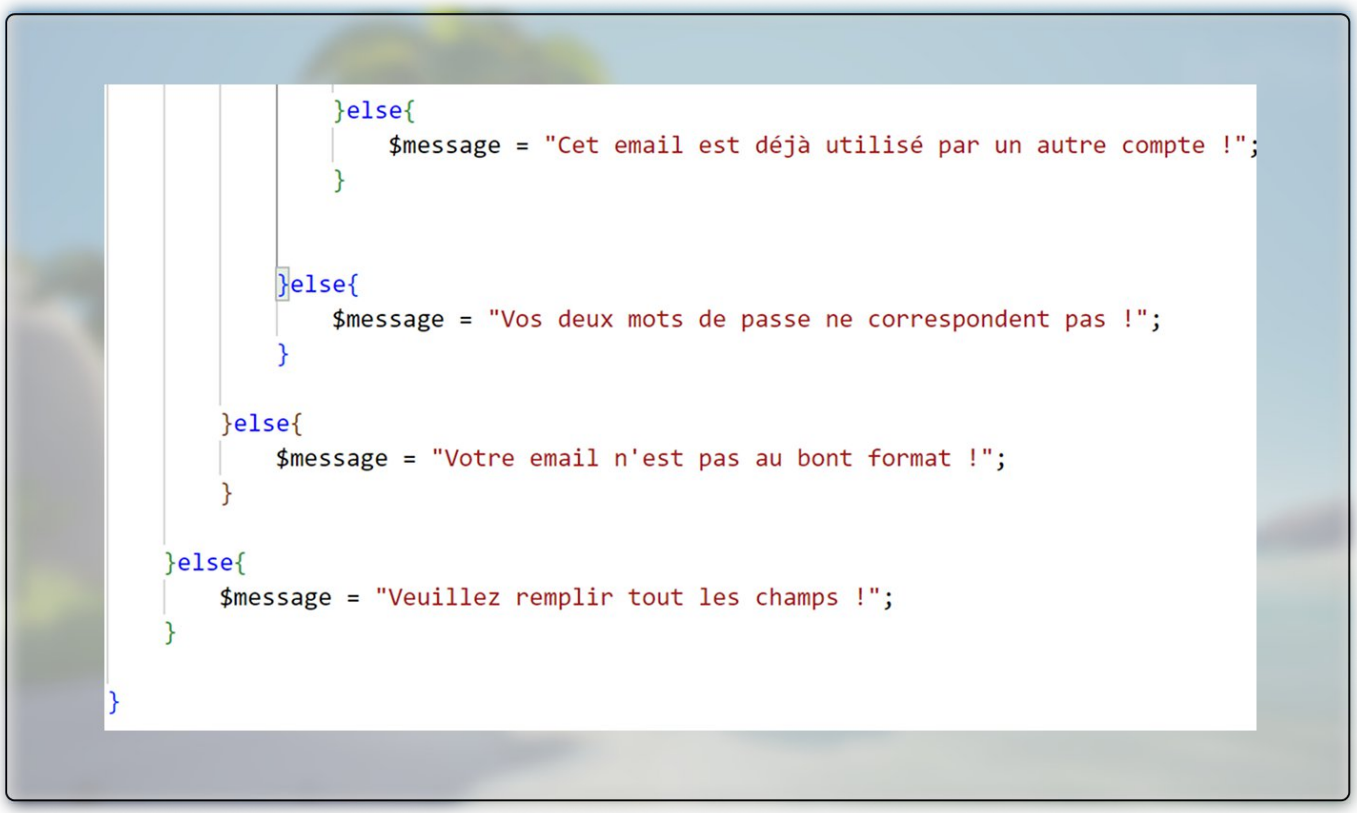
Dans notre formulaire, nous avons mis un champ de confirmation de mot de passe afin de pallier à une potentielle erreur de frappe de l'utilisateur lors de la saisie. Pour cela, nous effectuons une vérification des 2 champs pour une totale correspondance grâce au **« === »**. Toujours pour le mot de passe, nous procédons à un nettoyage des données pour éviter tout type d'attaque ou caractère non approprié et un **hachage de mot de passe** pour qu'il n'apparaisse pas de façon lisible en base de données.

Pour ce qui est du reste des informations utilisateur, elles sont elles aussi nettoyées et enregistrées dans notre objet utilisateur.

Nous terminons ce bloc en vérifiant si un utilisateur avec les mêmes informations n'existe pas déjà. Pour cela, nous avons créé une fonction qui va permettre de parcourir la table utilisateur.

Une fois toutes les conditions remplies, le compte utilisateur peut être enregistré et un message de validation est affiché pour en informer l'utilisateur.

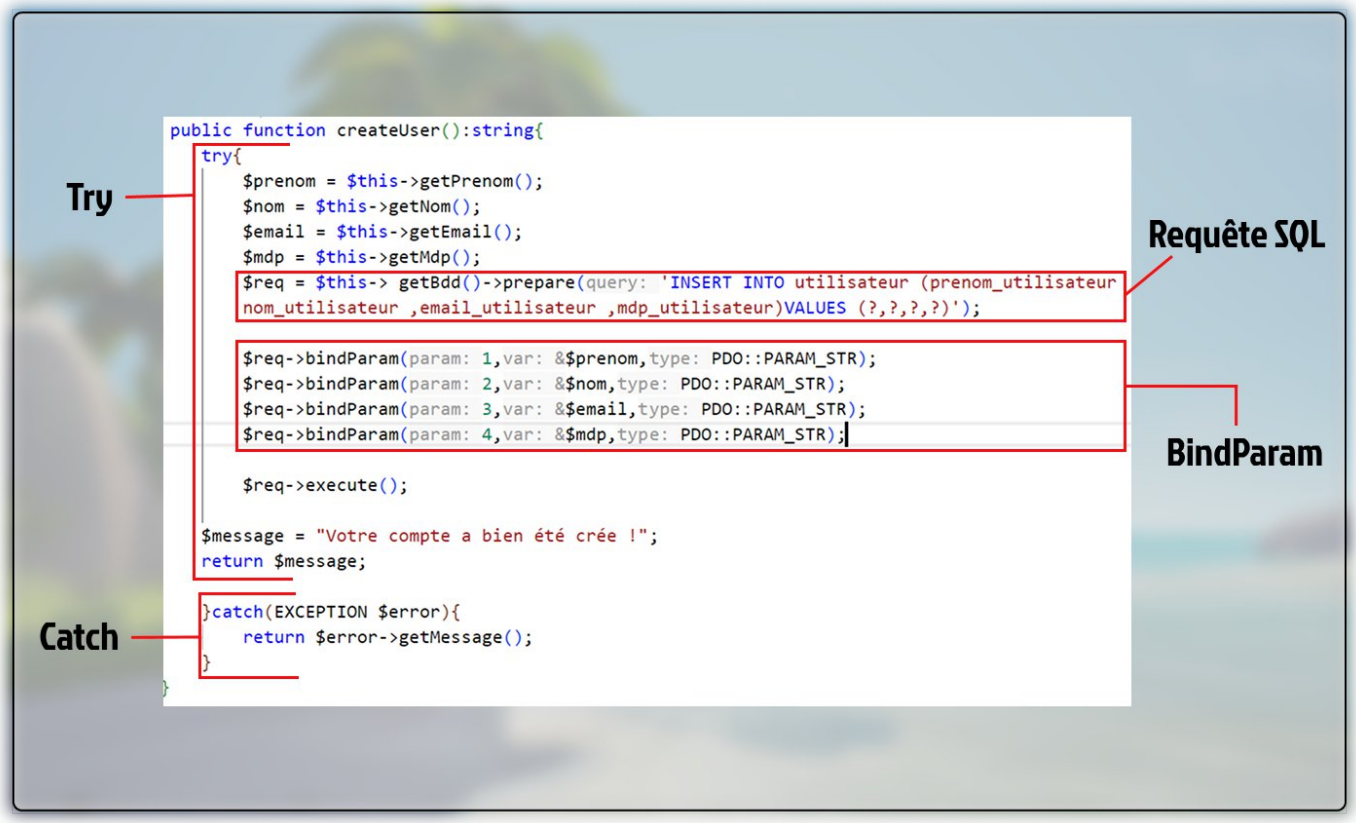
Cependant, si l'une des étapes n'est pas remplie ou qu'un problème survient, chaque « if » est accompagné de sa condition « else », elle permettra d'afficher un message d'erreur en fonction de l'étape qui a échoué afin d'avertir l'utilisateur pour qu'il puisse rectifier son erreur.



```
                }else{
                    $message = "Cet email est déjà utilisé par un autre compte !";
                }


            }else{
                $message = "Vos deux mots de passe ne correspondent pas !";
            }

        }else{
            $message = "Votre email n'est pas au bont format !";
        }

    }else{
        $message = "Veuillez remplir tout les champs !";
    }

}
```

8.6 Fonction creatUser

Revenons dans la classe utilisateur. Après avoir préparé la vue et le contrôleur nous allons maintenant déclarer la fonction qui va nous permettre de créer un utilisateur pour que l'envoi des données soit effectué.




```php
public function createUser():string{
    try{
        $prenom = $this->getPrenom();
        $nom = $this->getNom();
        $email = $this->getEmail();
        $mdp = $this->getMdp();
        $req = $this-> getBdd()->prepare(query: 'INSERT INTO utilisateur (prenom_utilisateur
        nom_utilisateur ,email_utilisateur ,mdp_utilisateur)VALUES (?,?,?,?)');

        $req->bindParam(param: 1,var: &$prenom,type: PDO::PARAM_STR);
        $req->bindParam(param: 2,var: &$nom,type: PDO::PARAM_STR);
        $req->bindParam(param: 3,var: &$email,type: PDO::PARAM_STR);
        $req->bindParam(param: 4,var: &$mdp,type: PDO::PARAM_STR);

        $req->execute();

    $message = "Votre compte a bien été crée !";
    return $message;

    }catch(EXCEPTION $error){
        return $error->getMessage();
    }
}
```

Dans cette fonction nous allons mettre en place une gestion des erreur **« try » - « catch »** .

Le bloc « try » a pour missions d'insérer un nouvel utilisateur en base de données il va récupérer et enregistrer toute les saisie effectuées dans le formulaire. Il va également faire une **requête préparer SQL** pour cibler la zone d'enregistrement.

Pour éviter tout type d'attaque nous allons mettre en place un **« bindParam »** pour chaque champ. Il va nous permettre de dissocier la requête SQL des paramètres. Ainsi la requête est pré compilée sans aucune valeur, et les paramètres sont ajoutés un par un comme de simples données et non comme du code exécutable. Cela nous permet de sécuriser nos requêtes contre les injections SQL .

Le bloc catch a pour rôle de contrôler l'affichage des erreurs de manière sécurisée. Il évite l'affichage de données sensibles tel que la structure de la base de données.

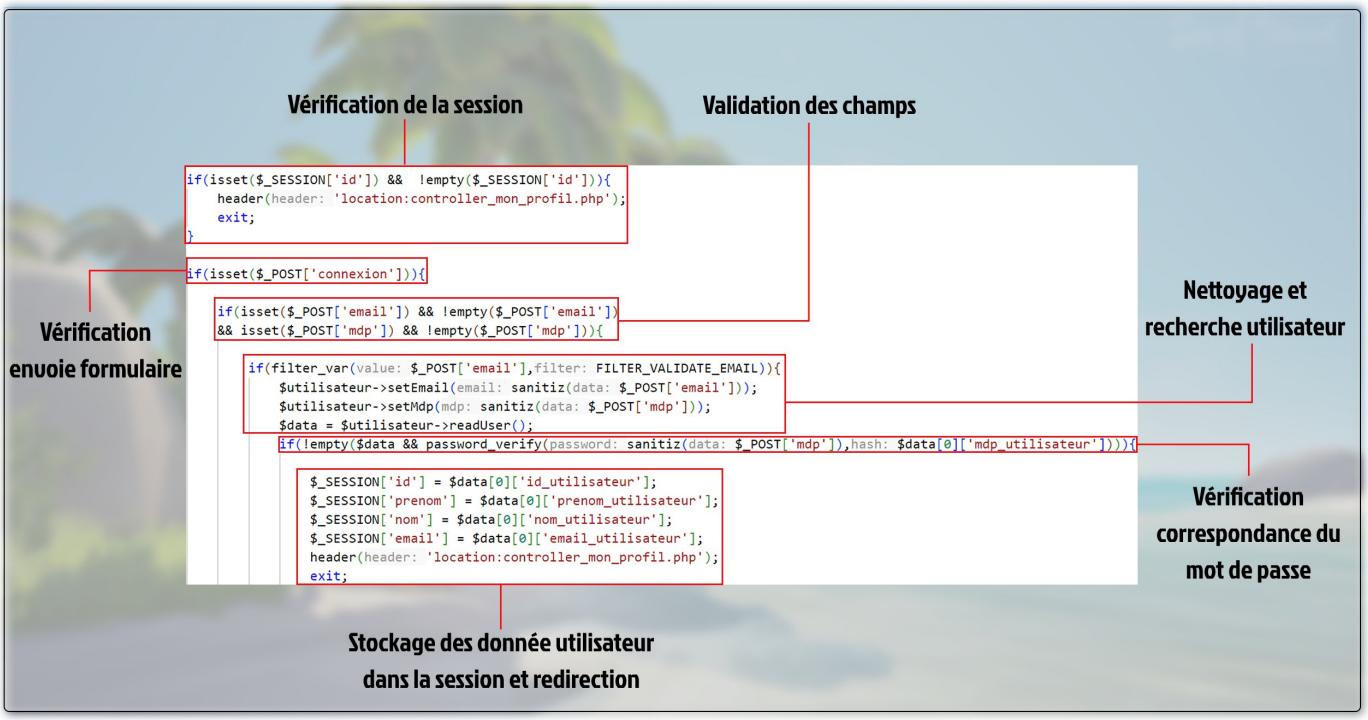
La création de notre utilisateur vient donc d'être validée voici le résultat en base de donnée.

id_utilisateur	prenom_utilisateur	nom_utilisateur	email_utilisateur	mdp_utilisateur
1	thomas	ciccarella	thciccarella@gmail.com	$2y$10$zYVAeQPF130YGJJMUlUOoeMExZdh6Ul...

8.7 Connexion

Maintenant que notre compte est créé, il va falloir nous connecter. La vue de la connexion est la même que l'inscription, elle demande l'e-mail et le mot de passe utilisateur, à la différence que dans l'inscription nous faisions une insertion en base de données ici nous parcourons cette dernière pour trouver les identifiant de l'utilisateur. La différence sera donc visible dans le contrôleur.

8.8 Contrôleur et vue profil




```php
if(isset($_SESSION['id']) &&  !empty($_SESSION['id'])){
    header(header: 'location:controller_mon_profil.php');
    exit;
}

if(isset($_POST['connexion'])){

    if(isset($_POST['email']) && !empty($_POST['email'])
    && isset($_POST['mdp']) && !empty($_POST['mdp'])){

        if(filter_var(value: $_POST['email'],filter: FILTER_VALIDATE_EMAIL)){
            $utilisateur->setEmail(email: sanitiz(data: $_POST['email']));
            $utilisateur->setMdp(mdp: sanitiz(data: $_POST['mdp']));
            $data = $utilisateur->readUser();
            if(!empty($data && password_verify(password: sanitiz(data: $_POST['mdp']),hash: $data[0]['mdp_utilisateur']))){

                $_SESSION['id'] = $data[0]['id_utilisateur'];
                $_SESSION['prenom'] = $data[0]['prenom_utilisateur'];
                $_SESSION['nom'] = $data[0]['nom_utilisateur'];
                $_SESSION['email'] = $data[0]['email_utilisateur'];
                header(header: 'location:controller_mon_profil.php');
                exit;
```

La première étape consiste à vérifier l'état de notre session. Cette vérification permet de savoir si nous sommes déjà connectés ou non. Si nous sommes connectés nous sommes renvoyés directement à la page profil où y seront présentes toutes les informations sur notre compte. A la fin de ce bloc vous pouvez voir **« exit »**. Il a pour rôle d'arrêter l'exécution du script PHP et empêche le code suivant de s' exécuter car il n'est pas utile.

Mais dans le cas où aucun utilisateur ne serait connecté nous allons comme pour l'inscription vérifier si le bouton « connexion » a bien soumis le formulaire.

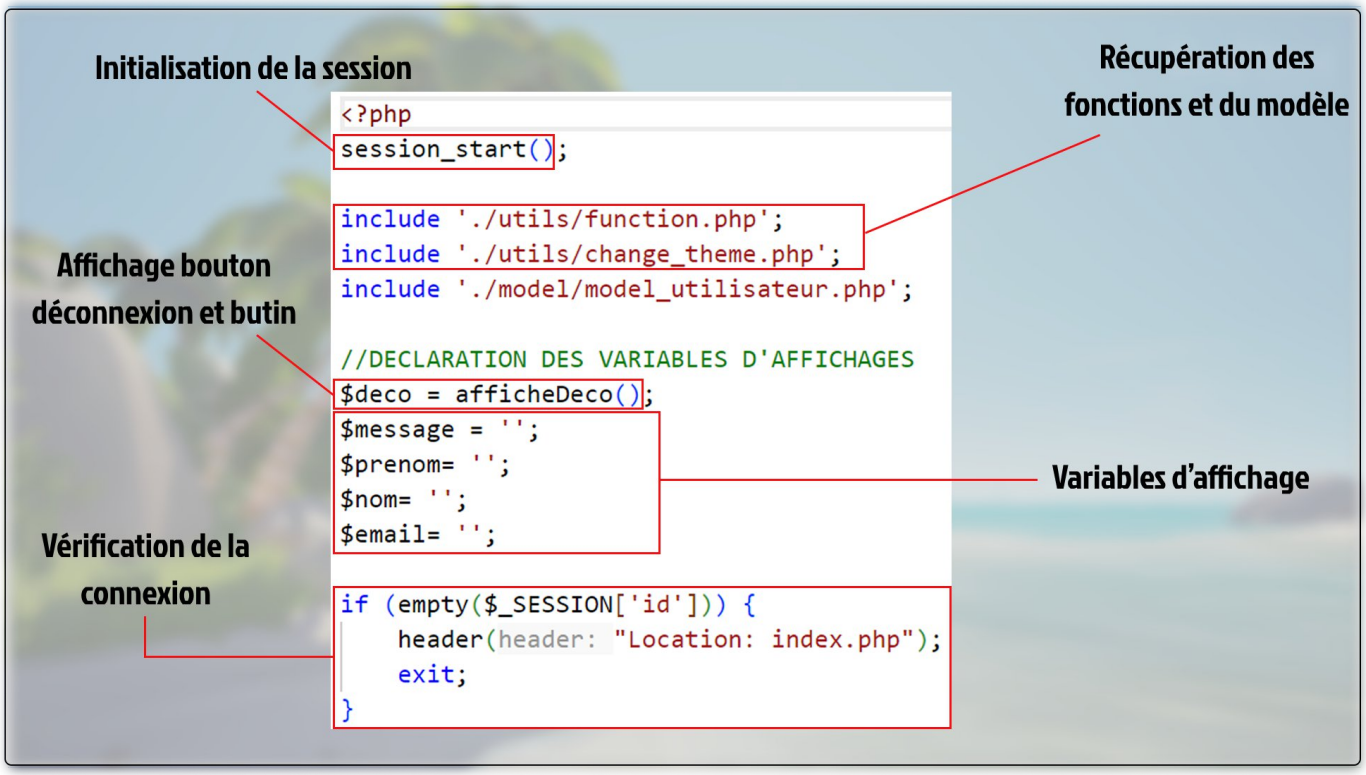
Si c'est le cas, nous allons maintenant vérifier si les 2 champs ont été remplis, puis pour des raisons de sécurité nous allons nettoyer les données saisies pour éviter tout type d'attaque.

Nous allons maintenant vérifier la correspondance entre le mot de passe saisi et le mot de passe haché en base de données.

Une fois toutes les conditions remplies avec succès, nous allons maintenant stocker toutes les données utilisateur de la base de données dans les **superglobales « $_SESSION »** .Cela va permettre à l'utilisateur de rester connecté sur toutes les pages du site web grâce à la fonction **« session_start »** présente dans toutes les pages du site. vous pouvez voir un exemple page 31 dans « Création des liens avec le contrôleur » .

Il faut tout de même prendre en compte que l'utilisateur aura la possibilité de rester connecté 24 minutes. Au-delà de ce délai, dans le cas où il serait inactif, il sera automatiquement déconnecté.

Nous finissons ce bloc avec une redirection de l'utilisateur vers la page **« mon profil »** , et, comme pour le contrôleur inscription, nous avons une gestion des erreurs possibles via des **« else »** pour informer l'utilisateur de la potentiel correction à effectuer. La une seule différence est que nous sommes un peu plus vague dans les codes erreur pour l'utilisateur pour des raisons de sécurité afin de ne pas aider un potentiel utilisateur malveillant à se connecter à un compte ne lui appartenant pas.

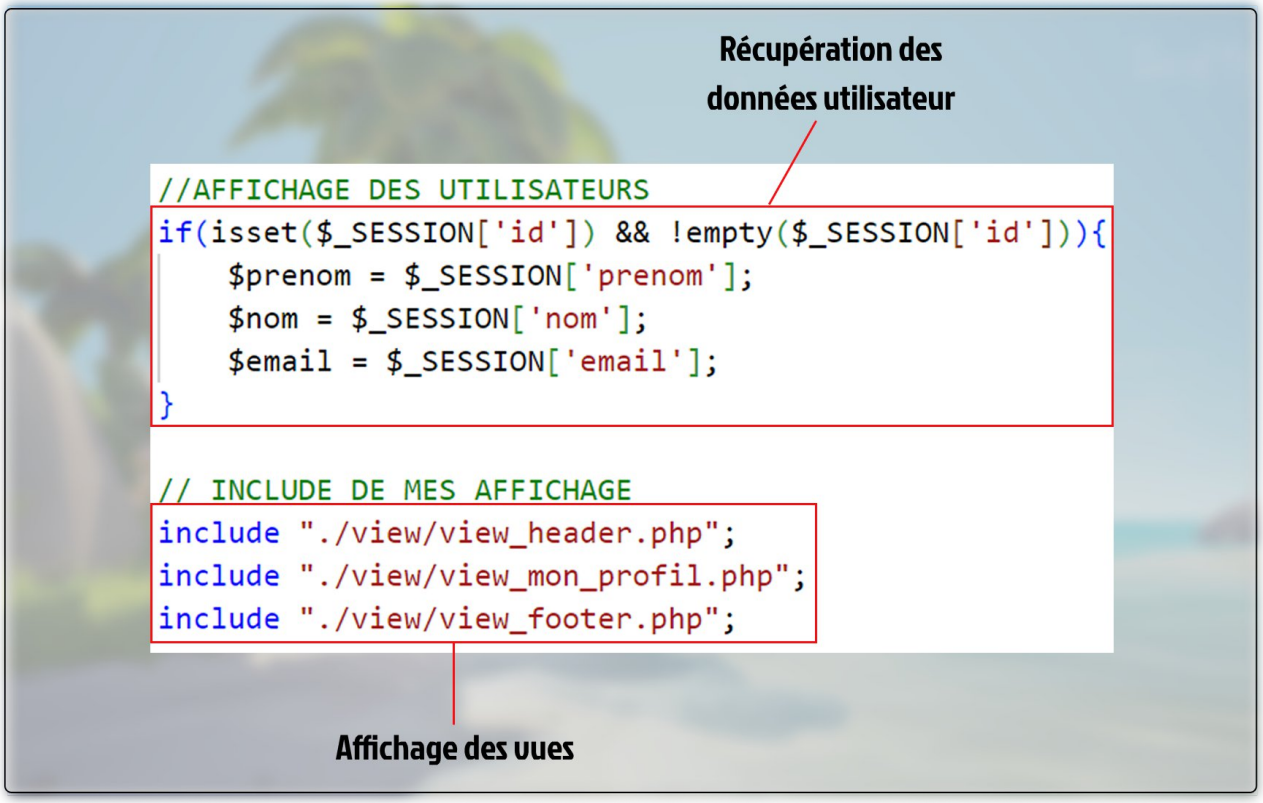


Comme pour toutes les pages le contrôleur profil va initialiser la session et récupérer les différentes parties de code dont-il a besoin. Ici nous appelons les fonctions et le modèle utilisateur.

Nous allons ensuite déclarer les variables d'affichage, elles auront pour rôle de stocker les données utilisateurs et quand nous les appellerons dans la vue elles afficheront ces dernières.

Dans les variables d'affichage vous pouvez voir la fonction **« afficheDeco »**. Elle a pour but d'afficher le bouton de déconnexion et butin dans le « header » quand l'utilisateur est connecté. La fonction n'affiche rien si elle ne détecte aucun compte connecter.

Pour ce qui est de la vérification de connexion, elle a pour but de rediriger l'utilisateur vers la page d'accueil si elle ne détecte aucun compte connecté.



Toujours dans le contrôleur, si l'utilisateur existe et n'est pas vide, nous allons récupérer les données de l'objet utilisateur grâce à la **superglobale « $_SESSION »** et les afficher dans la vue ci-dessous.

Grâce aux balises PHP, nous pouvons affichées les variables contenant les donnée utilisateur via « echo ».

9. Conclusion et suite

Ce projet a été un vrai plaisir à faire. Parler d'un jeu que j'affectionne particulièrement et surtout progresser tout au long de l'année en mettent en place ce que l'on apprenait a été un très enrichissant. Il est vrai que j'ai rencontré quelques difficultés mais cela m'a permis de découvrir de nouvelles façons de travailler et développer de nouvelles compétences en développement.

L'avancement de mon travail sur ce site est encore en cours et le sera toujours afin de l'améliorer continuellement. Le projet que j'ai mis en place pour mon stage chez CYBERTEK Labège, m'a pris énormément de temps et ne m'a pas permis de travailler en même temps sur le site que je vous ai présenté aujourd'hui.

J'aimerais vraiment remercier les formateurs de l'ADRAR, tout d'abord pour la confiance qu'ils m'ont accordé en m'acceptant dans la formation malgré mes lacunes académiques. Ils ont été un vrai soutien tout au long de l'année et ont joué un rôle crucial dans mon évolution.

Alors, un grand merci à eux !

Pour ce qui est de la suite, j'aimerais vraiment pouvoir continuer à me former et viser d'autres diplômes pour pouvoir un jour vivre de ce métier. Cette formation m'a permis de me rendre compte que ce métier dépasse toutes mes espérances, j'aimerais continuer d'apprendre pour développer de nouvelles compétences, qu'elles soient du coté front-end ou back-end. Les possibilités de création dans ces deux domaines m'intéressent haut plus au point ; ce qui était une simple pensée à la base est devenue un réel objectif aujourd'hui , alors je vais tout faire pour pouvoir le concrétiser. J'espère obtenir mon diplôme à la fin de la formation et postuler à la formation concepteur développeur d'application proposé par l'ADRAR.

Je vous remercie de l'attention que vous avez porté à mon mémoire et je vous souhaite une très bonne journée.

Thomas CICCARELLA