



Région académique  
NORMANDIE

académie  
Rouen



# RAPPORT DE STAGE

Réalisé par :

Oliwer SKWERES

Encadrant :

Lucas SCHNEKENBURGER

Stage effectué chez Altameos pour une durée de 5 semaines.

BTS SIO - Année : 2021/2022

# SOMMAIRE

0/REMERCIEMENT .....	3
1/Présentation de l'entreprise.....	4
A/L'entreprise en elle-même.....	4
B/Les moyens informatique .....	4
C/Accueil.....	4
2/Présentation du projet .....	5
A/Le client.....	5
B/Le projet.....	5
C/Les outils utilisés .....	5
3/Mise en place .....	6
A/Les tâches confiées .....	6
B/La méthode utilisée .....	6
C/La base de données.....	7
4/Le projet .....	8
A/Le commencement .....	8
B/Page de connexion.....	10
C/ Page d'administration.....	11
A/Personnalisation du site .....	11
B/Gestion question / réponses.....	12
D/ Le jeu.....	13
A/Salon avant partie.....	13
B/Plateau .....	14
E/Sauvegarde partie .....	17
F/Modifications du code.....	19
A/HTML / PHP.....	19
B/CSS.....	19

## 0/Remerciment

Je voudrais avant tout remercier l'entreprise Altameos, spécialement Monsieur Lucas Schnekenburger, qui a pu m'accueillir dans son entreprise et me transmettre son savoir, de m'avoir fait confiance sur la réalisation du projet et de m'avoir accompagné tout au long de mon stage me permettant d'améliorer mes connaissances.

Je voudrais aussi remercier Mme Hellard, gérante du BTS SIO au lycée Saint Adjutor à Vernon, qui m'a permis de rentrer en contact avec l'entreprise Altameos afin de réaliser mon stage pour l'année 2021-2022 en BTS SIO.

# 1/Présentation de l'entreprise

## A/L'entreprise en elle-même

Altameos Multimédia est une agence WEB fondée en 2003 dont le but est de réunir graphistes, programmeurs et pros du web pour répondre au besoin d'entreprise sur Internet en général. Elle se situe à Evreux, au 9b Rue du Puits Carré. Altameos propose plusieurs services comme le Web Design, qui permet de créer l'image de l'entreprise sur Internet, le référencement afin d'améliorer la visibilité des clients sur Internet et le conseil numérique afin d'apporter le savoir pour la réalisation des projets aux clients. L'agence travaille notamment avec des grandes entreprises comme VINCI ou encore Renault.

L'entreprise est une PME dont le dirigeant est Monsieur Lucas SCHNEKENBURGER, qui était aussi mon maître de stage pour la période 2021-2022.

## B/Les moyens informatique

Afin de parvenir à satisfaire les clients, Altameos utilise les logiciels Visual Studio Code afin de développer les sites web pour les clients. Le logiciel Visual Studio Code fourni par Microsoft permet de traiter du code sur ordinateur et de l'exécuter. Ce logiciel permet d'utiliser plusieurs langages comme le HTML, JavaScript, PHP et bien plus encore. Ce logiciel flexible permet l'utilisation de bibliothèque comme JQuery ou AJAX pour permettre une meilleure gestion du site ou encore l'installation d'outils tiers pour nous aider à développer. Altameos utilise aussi MYSQL afin de créer et gérer des bases de données. Les requêtes SQL sont envoyées depuis les sites grâce au langage PHP.

## C/Accueil

Etant donnée que j'ai effectué mon stage depuis chez moi, on m'a accueilli directement sur Teams. J'ai pu parler avec mon maître de stage afin qu'il m'explique le projet et ce qu'il attendait de moi pour ce projet et pour qui on le réalisait.



*Logo de Altameos*

## 2/Présentation du projet







### A/Le client


Dans le cadre de ce projet, le client est un hôpital à Evreux qui s'occupe des personnes atteintes d'Alzheimer. Le client exige un site Internet qui permettrait la gestion des patients mais aussi un jeu intégré dans le site, une sorte de Trivial Pursuit / Monopoly, qui permettrait aux personnes atteintes d'Alzheimer d'entraîner leurs mémoires avec des questions sur les cases à la place des propriétés.

### B/Le projet

Le but de ce projet est de proposer au bout de 5 semaines, un site fonctionnel pour le client. Ce site peut être modifié ensuite pour le revendre à une autre entité que l'hôpital comme une école. Le site se diviserait en plusieurs catégories : L'accueil, l'administration, le jeu, la gestion de compte et la page contact. On doit pouvoir sécuriser la connexion entre l'utilisateur et le site afin d'éviter de possibles piratages. Il faut tout d'abord développer le back du site puis ensuite le front. Le back du site permettra d'assurer le bon fonctionnement du site, gérant la saisie des requêtes SQL, la création de parties et la connexion tandis que le front sera centré sur l'aspect graphique du site, permettant une navigation facile à prendre en main.

### C/Les outils utilisés

Le site sera développé en HTML  en utilisant bien évidemment du PHP  , JavaScript  et CSS  afin de rendre le site dynamique et de gérer l'aspect graphique. L'utilisation des bibliothèques comme AJAX  et JQuery  peut être nécessaire pour la gestion du jeu. La base de données une base MySQL qui sera utilisé pour gérer la base de données et d'y insérer les requêtes. MYSQL Workbench permettra de créer la base de données plus aisément.

Pour coder, on utilisera le logiciel Visual Studio Code  et pour utiliser la base de données et pouvoir consulter le site, on utilisera PHPMyAdmin, bien que on aurait pu utiliser MAMP ou encore XAMPP.

## 3/Mise en place

### A/Les tâches confiées

Après un entretien via Teams, on m'a fait part des tâches à faire sur le site. Je devais créer le site petit à petit, avec possiblement des tâches qui se rajouteront en plus. Les tâches qui m'ont été attribuées sont :

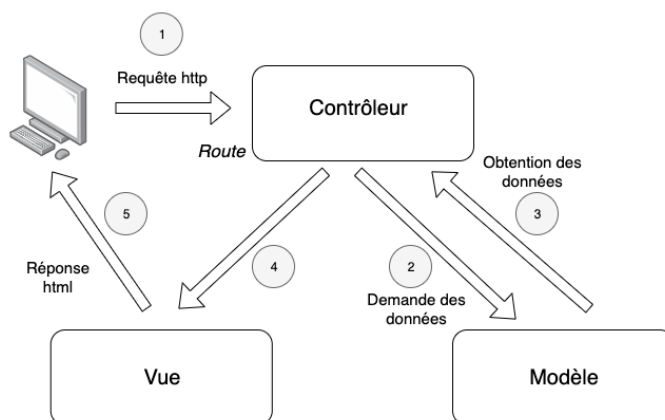
- Création de la base de données
- Création d'une page d'accueil
- Création d'une page de connexion au site
- Création d'une page d'administration
  - Gestion des patients (Ajout et suppression)
  - Ajout de questions et réponses (avec ou sans image)
- Ajout de la fonctionnalité « Mot de passe oublié » à la page de connexion
- Création d'une page nouvelle partie
- Ajout d'une page mon compte

### B/La méthode utilisée

Pour commencer, il fallait établir une méthode de travail. 2 choix s'offraient à nous : Faire les pages dans un seul et même dossier ou alors utiliser la méthode MVC (Modèle – Vue – Contrôleur). J'ai décidé d'opter pour la 2<sup>nd</sup> option, car je la trouve beaucoup plus pratique.

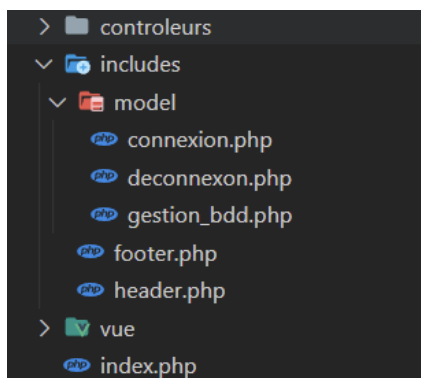
La méthode MVC est composée de 3 modules : les modèles, les vues et les contrôleurs. Le modèle contient les données à afficher, la vue représente l'interface graphique et le contrôleur représente toutes les actions effectuées par l'utilisateur. La méthode MVC est assez

simple à comprendre. Dans un premier temps, notre ordinateur va envoyer une requête au contrôleur qu'on a créé, ensuite, ce contrôleur va se diriger soit vers un autre contrôleur, soit directement faire une demande de données (par exemple une requête SQL). Ensuite, le contrôleur va afficher dans une vue, qui est une page internet, le résultat de notre demande. C'est ce qui permet de rendre les sites dynamiques et non statiques.



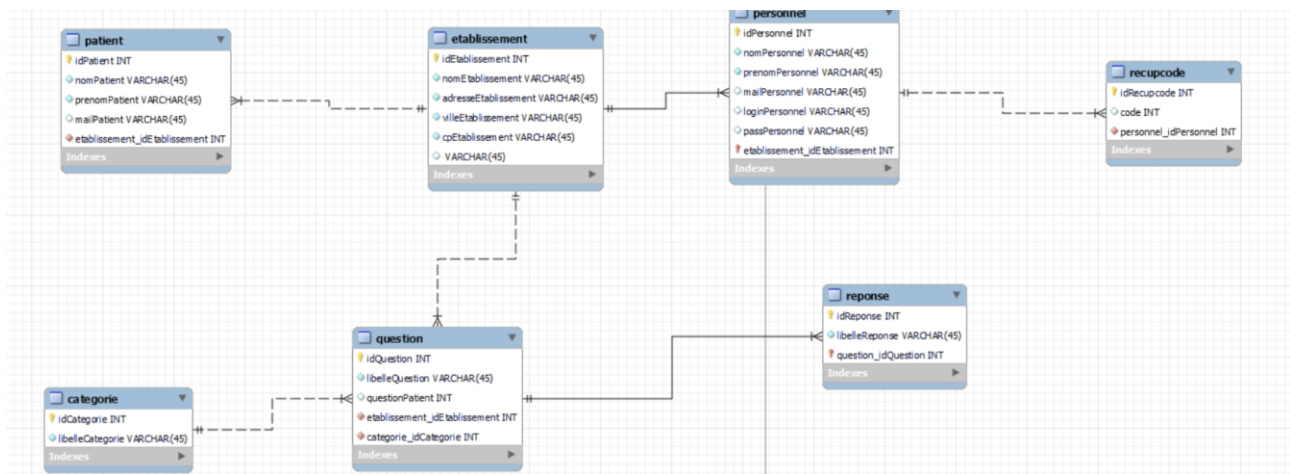
Utiliser la méthode MVC est très pratique, car tout ne se retrouve pas au même endroit. Chaque partie du code est séparée, si nous devons agir sur une requête SQL, nous allons directement dans les modèles, si c'est l'aspect graphique, ça sera la vue.

Donc pour commencer le projet, j'ai initialisé mon projet dans mon WampServer, sans ça, notre partie PHP ne s'affichera pas. J'ai donc créé plusieurs dossiers et fichiers afin de commencer. Un fichier `index.php`, qui contiendra seulement des includes, qui permettra d'appeler les fichiers `connexion.php`, `deconnexion.php`, `footer.php` et `header.php`. Ces fichiers aussi ont été créés. En plus des fichiers, viennent s'ajouter les dossiers contrôleurs, model ainsi que vue. L'architecture du projet à ce stade ressemble à ceci :



## C/La base de données

Après la mise en place du début du projet, j'étais obligé de faire avant tout la base de données, car c'est elle qui enregistrera nos données. J'ai utilisé MySQL Workbench, qui permet de créer des bases de données facilement. Tout d'abord, je me suis intéressé à l'aspect pratique, de ce que j'aurais besoin pour commencer, et je modifierai la base en fonction de mon besoin plus tard. Tout d'abord, j'ai créé une table établissement, qui contient un identifiant, un nom et une adresse. Cette table permet d'identifier l'établissement dans lequel on se trouvera une fois sur le site. Ensuite, j'ai créé la table personnelle, avec un identifiant pour le personnel, un nom et prénom, un mail, un login et mot de passe. Cette table personnelle est reliée à l'établissement et l'identifiant de l'établissement est une clé étrangère dans la table personnelle, car un personnel peut faire partie d'un seul établissement, tandis qu'un établissement peut contenir plusieurs personnes. J'ai ensuite créé la table patiente, avec un identifiant, nom, prénom, mail et dans la même logique que le personnel, l'identifiant de l'établissement est dans la table patient. J'ai ensuite créé une table question, qui regroupera les questions liées à un établissement grâce à la clé étrangère. Cette table est aussi liée à la table catégorie, permettant de donner une catégorie définie à une question. La table réponse contient les réponses possibles à la question posée. Comme plusieurs réponses sont possibles à une seule question, j'ai décidé de mettre une clé étrangère dans la table réponse. Ceci est donc la première version de la base de données, me permettant de commencer à travailler sur le site. Sur la page suivante, on retrouve un screen du début de la base de données.



## 4/Le projet

### A/Le commencement

Pour commencer le projet, j'ai décidé de faire un peu de HTML et de CSS afin de faire un beau site. J'ai vite compris que faire quelque chose de beau me prendrait trop de temps, et ne me ferait pas avancer assez vite. Pour la page d'accueil, j'ai fait quelque chose de simple :

EPHAD SAINT MICHEL

[Accueil](#)
[Partie](#)
[Administration](#)
[Contact](#)
[Mon compte](#)
[Déconnexion](#)

# TRIVIAL

Nouvelle partie

Pour cette page de connexion, j'ai donc utilisé la méthode MVC. Quand on tape dans « localhost/trivial » la méthode MVC est exécutée. Cette URL se transforme alors en <http://localhost/trivial/index.php?uc=accueil&action=afficher>

Comment ça marche ? Lorsqu'on tape localhost/trivial, ce que va faire le navigateur, c'est ouvrir notre page index.php, dans cette page, nous avons plusieurs includes. Les includes permettant d'appeler une autre page php. Sur ce screen page à la page suivante, nous pouvons voir que notre index.php appelle le contrôleur c\_principal, qui est en quelque sorte le rond-point de tous nos contrôleurs pour le projet.



```

index.php X
index.php
1  <?php
2  include "includes/modeles/connexion_bdd.php"
3  include "includes/modeles/gestion_bdd.php" ;
4  session_start();
5  include "includes/footer.php" ;
6  include "includes/header.php" ;
7  include "contrôleurs/c_principal.php";
8

```

Lorsque on se rend dans notre contrôleurs principal, on voit ceci :

```

index.php  c_principal.php ●
contrôleurs > c_principal.php > ...
1  <?php
2  if (!isset($_REQUEST['uc'])) {
3      $uc = "accueil" ;
4  }
5  else {
6      $uc = $_REQUEST['uc'] ;
7  }
8
9  switch ($uc)
10 {
11     case 'accueil' : { include "c_page.php" ; break ;
12                     };
13
14     case 'auth' : { include "c_auth.php" ; break ;
15                  };
16
17     case 'gestion' : { include "c_gestion.php" ; break ;
18                      };
19
20

```

De la ligne 2 à 8, on peut voir un bout de code qui permet d'attribuer une valeur par défaut à « \$uc », qui est accueil, si aucune valeur n'est attribuée. Ensuite, le « switch(\$uc) » permet de faire des vérifications sur la valeur de notre \$uc. Ici, comme nous venons seulement d'arriver sur le site, notre \$uc = accueil par défaut. Donc, le switch(\$uc) va vérifier chaque cas (case) que nous avons renseigné. On peut voir à la ligne 11, que si \$uc = accueil, nous allons charger le contrôleur «c\_page.php », ce qui est notre cas. L'url commence alors à se structurer (<http://localhost/trivial/index.php?uc=accueil>) on retrouve dans notre URL, notre uc qui est égal à accueil.

Ensuite, dans le contrôleur c\_page.php, ce sera la même logique que pour le contrôleur principal.

```

index.php  c_principal.php ●  c_page.php X
contrôleurs > c_page.php > ...
1  <?php
2  if (!isset($_REQUEST['action']))
3      $action = "afficher" ;
4  else
5      $action = $_REQUEST['action'] ;
6
7
8  switch ($action)
9  {
10
11     case "afficher":
12         require "vues/v_index.php";
13         break;
14
15

```

La valeur de base de \$action sera affichée, et lorsque dans le switch(\$action) on trouvera la bonne valeur, ici afficher, on affichera une vue, ici v\_index.php, ce qui nous affichera notre vue :

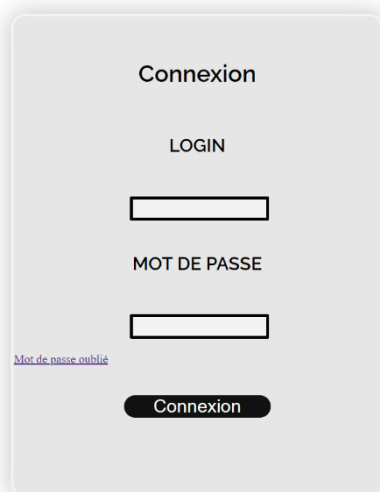
```
index.php  c_principal.php  v_index.php
vues > v_index.php > ...
1  <html>
2  <body>
3  <?php
4      if(!isset($_SESSION['connecter'])) {
5          header('Location:index.php?uc=auth');
6      } else {
7          ?>
8      }
9      <div id="container">
10         <div id="presentation">
11             <h1>TRIVIAL</h1>
12             <form method="post" action="index.php?uc=partie&action=newgame">
13                 <input type="submit" value="Nouvelle partie"></input>
14             </form>
15         </div>
16     </div>
17 <?php } ?>
18 </body>
19 </html>
20
```

Ce sera la même logique pour chaque page, on passera toujours par la méthode MVC.

## B/Page de connexion

Une fois la page d'accueil créée, j'ai fait la page de connexion. Nous sommes obligés de nous connecter au site pour y accéder. Sur cette page on retrouve seulement un formulaire de connexion.

Elle ressemble à ceci :



Pour faire cette page, j'ai fait une nouvelle vue v\_connexion.php, dans lequel se trouve le formulaire. Pour envoyer des informations vers notre contrôleur, on utilise la méthode POST afin de récupérer le mot de passe et le login. Cette page possède aussi une fonction « mot de passe oublié » qui, grâce à un token généré et envoyé par mail, permet de réinitialiser le mot de passe. Le token a une date d'expiration ainsi qu'une seule utilisation. Le fait de mettre un login obligatoire avant l'accès à cette page, permet de configurer le site comme bon nous le semble grâce à un compte Admin. On peut par exemple changer la couleur d'un site sans affecter la couleur d'un autre établissement.

```
$idPersonnel = getIdByMail($mail);
$token = bin2hex(random_bytes(16));
$date = date("Y-m-d H:i:s", strtotime('+30 minutes'));
codeOubliInsert($token,$date,$idPersonnel['idPersonnel']);
```

*Création (ligne2) et insertion (ligne 4) du token dans la base de données*

## C/ Page d'administration

Après avoir créé cette page, j'ai créé une page qui permettait de gérer justement le site, les patients et nos questions.

La page en entier ressemble à ceci :

The image shows three screenshots of the administration page forms, all with a light blue background and white text.

- Ajout patient:** A form with fields for 'NOM', 'PRENOM', and 'MAIL' (marked as optional). There is an 'Ajouter' button at the bottom.
- Suppression patient:** A form with a dropdown menu labeled '--- CHOISIR UN PATIENT ---' and a 'Supprimer' button.
- PERSONNALISATION DU SITE:** A form with a dropdown menu for 'Couleur du site' (labeled '--- Choix couleur ---'), a text field for 'Nom du site', and a 'LOGO' section with a file upload button 'Choisir un fichier'. There is an 'Enregistrer' button at the bottom.

## A/Personnalisation du site

Chaque onglet a donc sa propre utilisation, on peut ajouter un patient, supprimer un patient, ajouter une question ou encore personnaliser le site. La personnalisation permet de choisir une couleur parmi 10 proposées et de changer le nom du site. Pour changer la couleur des headers et des boxs, j'ai décidé de mettre une balise <style> dans mon header.php. Dans ce fichier, on attribue une couleur de base, une couleur sombre et une couleur clair comme ceci :

```
<?php
$leSite = getInfoSite($_SESSION['etabPerso']);
$color = $leSite['couleurSite'];
$nomSite = $leSite['nomSite'];
if($color == "#ff3300") { $baseColor = "#ff3300"; $brightColor = "#ff6666"; $darkColor = "#800000"; } // ROUGE
elseif($color == "#ff9900") { $baseColor = "#ff9900"; $brightColor = "#ffb84d"; $darkColor = "#b36b00"; } // ORGANGE
elseif($color == "#ffff00") { $baseColor = "#ffff00"; $brightColor = "#ffff80"; $darkColor = "#cccc00"; } // JAUNE
elseif($color == "#66ff33") { $baseColor = "#66ff33"; $brightColor = "#9fff80"; $darkColor = "#33cc00"; } // VERT CLAIR
elseif($color == "#009933") { $baseColor = "#009933"; $brightColor = "#00e64d"; $darkColor = "#006622"; } // VERT FONCE
elseif($color == "#33ccff") { $baseColor = "#33ccff"; $brightColor = "#80dfff"; $darkColor = "#0099cc"; } // BLEU CLAIR
elseif($color == "#3366ff") { $baseColor = "#3366ff"; $brightColor = "#809fff"; $darkColor = "#0039e6"; } // BLEU
elseif($color == "#6600ff") { $baseColor = "#6600ff"; $brightColor = "#a366ff"; $darkColor = "#4700b3"; } // VIOLET
elseif($color == "#ff00ff") { $baseColor = "#ff00ff"; $brightColor = "#ff80ff"; $darkColor = "#800080"; } // ROSE
elseif($color == "#f2f2f2") { $baseColor = "#e6e6e6"; $brightColor = "#f2f2f2"; $darkColor = "#cccccc"; } // BLANC
?>
```

On peut ensuite réutiliser les variables avec des balises PHP ou on le souhaite.

```
#box1Admin {
  position: absolute;
  top: 20%;
  left: 8%;
  width: 500px;
  height: 570px;
  background: <?= $baseColor; ?>;
  box-shadow: 0em 0em 2em <?= $darkColor; ?>;
  border-radius: 20px;
  border: 4px solid <?= $brightColor; ?>;
}
```

Ici, ce bout de CSS permet de gérer les carrés bleus qui contiennent nos formulaires. On peut ensuite répéter la même chose pour les autres box, l'header ou encore autres choses. Je n'ai pas trouvé d'autres solutions que celle-ci, mais je trouve qu'elle est très efficace, car elle s'applique partout.

## B/Gestion question / réponses

Pour le formulaire, on peut saisir une question tout en haut, ensuite, selon le nombre de réponses que l'on souhaite proposer, on écrit dans les `<input>` prévu à cet effet. On doit cocher le petit bouton radio, qui permet de sélectionner la réponse qui sera vraie. Si l'on saisit qu'une seule réponse, lorsqu'arrive le moment de répondre aux questions, nous aurons un simple input, tandis que s'il y a plusieurs réponses saisies, on aura des boutons radio avec les propositions.

En plus de ces questions-réponses, on peut ajouter un fichier réponse un et fichier question, qui peuvent être une image ou vidéo. Lorsque nous mettons un fichier réponse, ce fichier s'affichera lorsque nous aurons répondu à la question, et si nous saisissons un fichier question, il s'affichera lorsque la question sera posée. Il faut modifier cependant le fichier `php.ini` afin de pouvoir envoyer des fichiers volumineux, car sans cette modification, nous ne pouvons pas uploader les vidéos.

Pour finir, chaque question appartient à une catégorie. Sans la saisie de catégorie, nous ne pouvons pas enregistrer la question. Nous avons aussi la possibilité de lier une question à une personne, c'est-à-dire que c'est seulement cette personne qui aura cette question, les autres joueurs ne pourront pas tomber sur cette question. Si la question est liée à aucun joueur, elle sera disponible à tous les joueurs bien évidemment. Le code à ce stade n'est pas encore optimisé, je pense faire donc des optimisations et supprimer des éléments non essentiels au bon fonctionnement de l'application à la fin de mon stage.

## D/ Le jeu

### A/Salon avant partie

Lorsque nous voulons créer une nouvelle partie, nous arrivons dans ce salon :

#### SALON

Liste joueurs

Selection patient:

--- CHOISIR PATIENT ---

Combien de question ?

Ex:10

Je n'ai pas fait de CSS sur le jeu, car j'attendais les maquettes finales pour faire le CSS par la suite. Dans ce salon, nous pouvons ajouter des joueurs à la liste, on peut aussi les supprimer et on peut définir le nombre de questions souhaitées. A savoir que le nombre de questions n'est par personne. C'est-à-dire que si on précise 4 questions, ce sera 4 questions par personne. Le menu <select> est alimenté automatiquement avec du PHP comme le montre le code ce dessous :

```
<form method="post" action="index.php?uc=partie&action=ajoutJoueur">
  <p>LISTE JOUEURS </p><select name="ajoutPatient" >
    <option value="0"> Choisir un patient </option>
  <?php
    $cpt = 0;
    foreach($_SESSION['joueursPotentiel'] as $unJoueur) {
      echo '<option value="'. $unJoueur. '">'. $unJoueur. ' </option>';    $cpt++;
    }
  ?>
  <input id="buttonAdd" type="submit" name="submit" value="Ajouter" />
```

\$\_SESSION['joueursPotentiel'] contient la liste des patients. On parcourt ensuite cette liste grâce à un foreach, qui signifie "Pour chaque" et on identifie un joueur de cette liste comme \$unJoueur. On ajoute ensuite automatiquement les joueurs dans le menu select. Ce qui est pratique, c'est que cette méthode permet de ne pas répéter plusieurs fois le même code. Voici une comparaison :

```
<?php
$cpt = 0;
foreach($_SESSION['joueursPotentiel'] as $unJoueur) {
  echo '<option value="'. $unJoueur. '">'. $unJoueur. ' </option>';    $cpt++;
}
?>
```

*Avec un foreach*

```
<?php

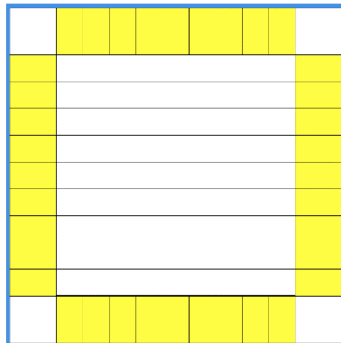
echo '<option value="Oliwer SKWERES">Oliwer SKWERES</option>';
echo '<option value="Yanis KARA">Yanis KARA</option>';
echo '<option value="Mohammed BADACHE">Mohammed BADACHE</option>';
echo '<option value="Eva BREAK">Eva BREAK</option>';

?>
```

*Sans foreach*

Imaginons que nous avons 50 patients, si on devait rajouter à la main, ça prendrait une éternité, alors qu'avec une boucle, ça se fait tout seul.

Lorsque on aura finalement choisi le nombre de questions ainsi que les joueurs qui participent à au jeu, nous pouvons lancer la partie. On a alors un plateau qui apparaît. C'est de loin la tâche la plus compliquée à faire selon moi, car elle implique l'utilisation de JavaScript, que nous n'avons pas étudié en cours et mes connaissances en Node.JS ne seront pas forcément utiles. Voici à quoi ressemble le plateau d'essai, sans vraiment de CSS :



Le plateau ressemble à celui du Monopoly, les joueurs apparaissent en haut à droite et doivent lancer un dé pour avance. Cependant, sur la case 0, 5, 10, 15... on retrouve les questions personnelles justement. Il faut donc gérer ce paramètre aussi.

Ce plateau a été fait avec des `<table>`, `<tr>` et `<td>`. Je n'ai pas spécialement cherché à optimiser la génération du tableau, je le ferais plus tard car j'avais vraiment envie d'avancer sur le jeu.

La génération du plateau ressemble à ceci. J'ai décidé d'attribuer un numéro à chaque case du tableau, afin de pouvoir y déplacer les pions. Pour afficher les pions, j'ai décidé d'utiliser des balises `<p>` avec le numéro de la case. Par exemple : `<p32>` pour la case 32 ou `<p2>` pour la case 2. Cet élément prend ton importance lorsqu'on commence à creuser en JavaScript sur une méthode pour faire déplacer le pion. Le style qu'il y a dans les cases va aussi disparaître lors de la refonte du CSS. J'ai mis directement dedans pour effectuer les premiers tests, sachant que le CSS n'affecte pas le fonctionnement du reste.

```
<td id="37" style="background-color: yellow;" width = 140px; height = 80px;><p id="p37"></p>
<td style="width:720px;"></td>
<td id="13" style="background-color: yellow;" width = 140px; height = 80px;><p id="p13"></p>
</tr>
<tr>
<td id="36" style="background-color: yellow;" width = 140px; height = 80px;><p id="p36"></p>
<td style="width:720px;"></td>
<td id="14" style="background-color: yellow;" width = 140px; height = 80px;><p id="p14"></p>
</tr>
</table><table>
<tr>
<td id="35" style="background-color: yellow;" width = 140px; height = 80px;><p id="p35"></p>
<td style="width:720px;"></td>
<td id="15" style="background-color: yellow;" width = 140px; height = 80px;><p id="p15"></p>
</tr>
<tr>
<td id="34" style="background-color: yellow;" width = 140px; height = 80px;><p id="p34"></p>
<td style="width:720px;"></td>
<td id="16" style="background-color: yellow;" width = 140px; height = 80px;><p id="p16"></p>
</tr>
</table><table>
<tr>
<td id="33" style="background-color: yellow;" width = 140px; height = 80px;><p id="p33"></p>
<td style="width:720px;"></td>
<td id="17" style="background-color: yellow;" width = 140px; height = 80px;><p id="p17"></p>
</tr>
<tr>
<td id="32" style="background-color: yellow;" width = 140px; height = 80px;><p id="p32"></p>
<td style="width:720px;"></td>
<td id="18" style="background-color: yellow;" width = 140px; height = 80px;><p id="p18"></p>
</tr>
</table>
```

Génération du plateau

Pour commencer, il fallait que j'apprenne un peu de JavaScript. Je me suis donc rendu sur W3School afin de connaître quelque syntaxe, notamment ceux des tableaux.

Je suis parti du principe que les pions et les positions seront stockées en base de données et JavaScript directement dans un tableau qui correspond à chaque joueur.

```
let pion = [['●', 0], ['●', 0], ['●', 0], ['●', 0], ['●', 0], ['●', 0]];
```

Dans cet array, on a chaque profil qui est stocké, avec une couleur de pion et sa position. Chaque joueur sera défini par la couleur de son pion.

Lorsqu'on lance une partie, une fonction va s'exécuter et attribuer une couleur de pion à chaque joueur.

```
function startGame(x) {  
  for(var i = 0; i < x; i++) {  
    players.push(pion[i]);  
    $('#p0').html(pion[i][0])  
  }  
  currentPlayer = 0;  
  localStorage.setItem("players", JSON.stringify(players));  
  localStorage.setItem("currentPlayer", JSON.stringify(currentPlayer));  
}
```

Avant d'exécuter cette fonction, on crée une liste de joueurs « players » vide. Ensuite, lorsque la fonction est lancée, nous allons ajouter un pion et sa position dans la liste des joueurs. C'est comme si Oliwer SKWERES devenait « ['●', 0] » et on le fait pour chaque joueur, de la ligne 2 à la ligne 5. La ligne 3 permet d'afficher le pion du joueur sur la case 0, ici "p0", d'où l'importance lors de la création du tableau, d'avoir mis un numéro pour chaque balise <p>. La 6eme ligne définit le joueur qui va jouer en premier, c'est-à-dire le joueur 0 et les lignes 7 et 8 permettent de stocker localement les informations sur la partie et on verra pourquoi plus tard.

C'est ce qui se passe au lancement du jeu.

Après avoir chargé le jeu, on se retrouve sur le plateau. On peut ensuite lancer un dé. Le dé est lancé grâce à un "OnClick" sur un bouton. Lorsque le bouton est appuyé, on lance une fonction qui s'appelle RollDice, celle qui est à droite. Cette fonction permet de générer un nombre aléatoire entre 1 et 6.

```
dice = getRandomInt(1,6);
```

```
function getRandomInt(min, max) {  
  return Math.floor(Math.random() * (max - min + 1) + min);  
};
```

Une fois le chiffre généré, on déplace notre pion case par case grâce à une boucle for. Pour ajouter un effet d'animation à notre pion, comme quoi il se déplace case par case, j'ai décidé que la fonction RollDice serait une fonction asynchrone. Pour résumé, la fonction va continuer de s'exécuter si une promesse « Promise » est retourné. J'ai donc décidé de mettre un delay, à chaque fois que l'on rajoute 1 dans notre boucle, afin de rendre notre

Le code commenté ce retrouve en dessous, afin de rendre l'explication du déplacement du pion plus facile.

```
//Pour x = 0, tant que x < dice, on ajoute 1 à x.
for(x = 0; x < dice; x++) {

    // On récupère la case du joueur et on supprime son pion, afin d'éviter de complètement vider la case si d'autres joueurs se trouvent sur la même case.
    var text = $('#p'+Math.floor(players1[currentPlayer][1])).text()
    var text = text.split(players1[currentPlayer][0])

    // On réattribue à la case, le contenu de celle ci sans le pion du joueur
    $('#p'+Math.floor(players1[currentPlayer][1])).html(text)

    // On enregistre ensuite la position du joueur en incrémentant de 1
    players1[currentPlayer][1] = Math.floor(players1[currentPlayer][1]) + 1;

    // Si la position du joueur est supérieur à 39, soit le nombre maximum de case, on le place à la case numéro 1
    if(Math.floor(players1[currentPlayer][1]) > 39) { players1[currentPlayer][1] = 0; }

    // Ensuite, on fait avancer le pion. On utilise .append afin de pas remplacer toute la valeur de notre case
    $('#p'+Math.floor(players1[currentPlayer][1])).append(players1[currentPlayer][0])

    // On attends 500ms avant de relancer cette fonction
    await delay(500);
}
}
```

Une fois que le pion a fini son déplacement, on regarde si le joueur se trouve sur une case multiple de 5. Si oui, on charge une question qui est liée à ce patient, sinon, on charge une question au hasard. Afin d'éviter d'utiliser les API, j'ai décidé de faire passer la position du joueur dans l'URL du site, et jsQuestion définit le type de question. Si jsQuestion = 0, c'est une question aléatoire, sinon c'est une question personnelle.

```
async function changeWindow(x) {
    await delay(500);
    pos = JSON.parse(localStorage.getItem("currentPlayer"))
    players1 = JSON.parse(localStorage.getItem("players"))
    window.location.href=`index.php?uc=partie&action=save&array=${players1[pos]}&jsquestion=0`;
    localStorage.removeItem("currentPlayer")
    currentPlayer = currentPlayer + 1;

    if(currentPlayer > players1.length - 1) { currentPlayer = 0 }

    localStorage.setItem("currentPlayer", JSON.stringify(currentPlayer));
}
}
```

On stock aussi localement les informations sur le joueur et je vais expliquer pourquoi un peu plus en dessous.

Après avoir choisi le type de question, on charge simplement la question, et on peut y répondre. S'il y'a une image, elle est chargée, s'il y'a une vidéo elle est aussi chargée. Si c'est une question à choix, nous avons des boutons radio qui s'affichent, sinon c'est un simple input texte qui s'affiche.

```
{ $questions[idcompte][5] != "" } {
    { substr($questions[idcompte][5],-3,1) == "mpa" } { substr($questions[idcompte][5],-3,1) == "mov" } {
        echo "<video width: 480px height: 480px src: res/questions/.$questions[idcompte][5].- type=video/.$questions[idcompte][5].-></video> ->";
    } else {
        echo "<img height=480px width=480px src= res/questions/.$questions[idcompte][5].->";
    }
}

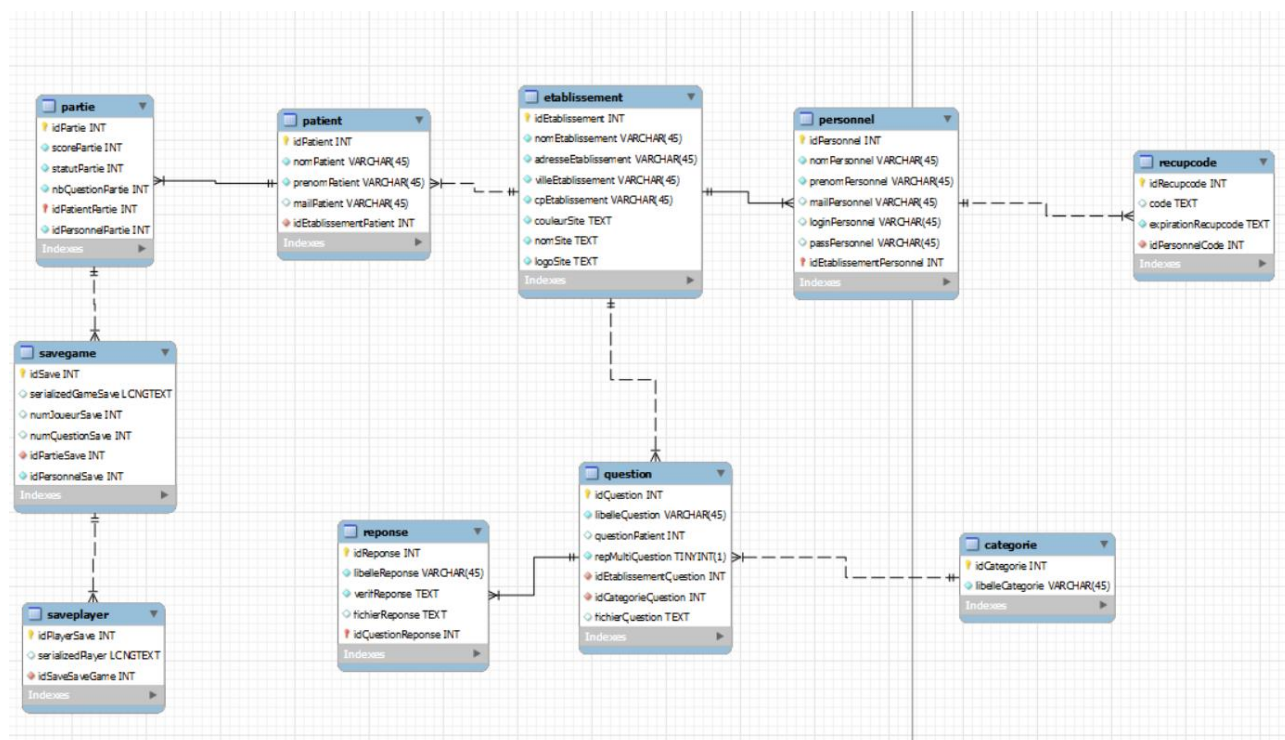
echo "<form method='post' action='index.php?uc=partie&action=verifreponse'>";
if($questions[idcompte][2] == 0) {
    echo "<input type='text' name='reponse'></input>";
    <input type="submit" name="verifreponse" value="Valider"></input>
</form>";
} else {
    echo "<form method='post' action='index.php?uc=partie&action=verifreponse'>";
    $rep = 0;
    while ($rep < $SESSION['reponses'][$idcompte]) {
        echo "<input type='radio' name='reponse' value='".$questions[idcompte][4][$rep].$questions[idcompte][4][$rep]."></input>";
        $rep++;
    }
    { (isset($SESSION['repvalide'])) {
        echo "<input type='submit' name='verifreponse' value='Valider'></input>";
    }
    echo "</form>";
}
```

Une fois avoir saisi notre réponse, on vérifie si la réponse saisie est bonne ou non grâce à du PHP. Si la réponse est bonne, on rajoute 1 au score du joueur, sinon on ne fait rien. On retourne ensuite sur la page du jeu afin de rendre dire si le joueur a eu bon ou non.



## E/Sauvegarde partie

Pour fini la partie jeu, on m'a demandé le fais de pouvoir enregistrer les parties. J'ai du modifier la base de données pour cela afin de



J'ai donc rajouté 3 tables : Partie, Savegame et Saveplayer.

Partie regroupera toutes les parties depuis le début du site, on pourra ainsi les consulter si la colonne « statutPartie » vaut 1 et les reprendre si elle vaut 0.

Ensuite, savegame va contenir la sauvegarde de la partie, comme toutes les questions, le prochain joueur qui doit jouer et la prochaine question.

Cette table est liée à saveplayer, dans lequel on va stocker les informations à propos du joueur.

Afin de pouvoir stocker toutes ces informations, j'ai cherché plusieurs solutions et j'ai trouvé la fonction serialize et unserialize. Cette fonction permet de transformer un array en une série de lettre et de chiffre, permettant un stockage en base de données. Voici un exemple :

```
a:1:{i:0;a:6:{i:0;i:0;i:1;s:8:"X Oliwer";i:2;i:0;i:3;s:1:"1";i:4;s:4:"????";i:5;i:0;}}
```

Cette méthode permet de stocker alors les informations sur le joueur, et lorsque on unserialize, cette chaîne de caractères redeviens tout simplement un array.

Pour la reprise de partie, il suffit tout simplement de réattribuer les valeurs déjà faite avant, avec les éléments qui sont serialized.

```

case "repandre":
    $jeu = getSaveById($_SESSION['idPerso'], $_GET['partieId']);

    $_SESSION['joueursInGame'] = unserialize($jeu[4]);
    $_SESSION['decompte'] = $jeu[3];
    $_SESSION['numJoueur'] = $jeu[2];
    $_SESSION['question'] = unserialize($jeu[1]);
    $rep = unserialize($jeu[1]);
    $_SESSION['reponses'][$_SESSION['decompte']] = count($rep[$_SESSION['decompte']][4]);
    $lesJoueurs = $_SESSION['joueursInGame'];

    $players = array();
    $cpt = 0;

    foreach($_SESSION['joueursInGame'] as $unJoueur) {
        $p = $unJoueur[4];
        $players[$cpt][0] = json_decode("\"\".substr($p,0,5).\"\".substr($p,5,strlen($p)).\"\"");
        $players[$cpt][0] = $players[$cpt][0];
        $players[$cpt][1] = $unJoueur[5];
        $cpt++;
    }

    ?>
    <script type="text/javascript"> resumeGame(<?php echo $_SESSION['numJoueur'];?>,<?php echo json_encode($players);?>)</script>
    <?php
    require "vues/v_plateau.php";
    break;

```

Grâce à une requête SQL, on récupère la partie sauvegarder, et on réattribue les bonnes valeurs à chaque variable.

Le plus compliqué dans la sauvegarde de partie, était de passer du JavaScript en PHP. Ce que je pouvais faire, c'est apprendre AJAX, ce qui m'aurait permis d'utiliser une API, afin de passer les variables plus facilement entre PHP et JavaScript. Cependant, par manque de temps, je n'ai pas pu m'attarder là-dessus, car je voulais à tout prix finir le projet dans les délais. Pour contourner AJAX, j'ai donc utilisé la technique de passer les informations dans l'URL, afin de récupérer la position du pion et le pion du joueur. J'ai donc pu sauvegarder ceci. Cependant, pour reprendre le jeu, j'ai dû utiliser en JavaScript, le localStorage, qui permet de stocker des informations dans le navigateur. J'étais obligé d'utiliser cette méthode, afin de ne pas avoir à redéfinir les variables à chaque fois, car une fois un script en JavaScript exécuté, JavaScript ne retiens pas les valeurs, à moins qu'on utilise le localStorage. La dernière épreuve sur laquelle j'ai buté était le pion du joueur. Les émojis sont définis différemment en fonction du langage. Par exemple, ici on a un cercle rouge 🟠, sauf que JavaScript le définit comme \uD83D\uDD34, et PHP comme \u{1F534}. On a donc deux différences. Pour ce faire, j'ai dû enregistrer l'émoji JavaScript en enlevant les « \ », et si on reprend la partie, on aura juste à les remettre pour que JavaScript.

```

$players[$cpt][0] = json_decode("\"\".substr($p,0,5).\"\".substr($p,5,strlen($p)).\"\"");

```

Cette ligne permet justement de remettre le pion de PHP en JavaScript.

```

function resumeGame(nbJoueur, players1) {
    localStorage.removeItem("currentPlayer");
    localStorage.removeItem("players");
    console.log(players1)
    let players = [];
    for (var i = 0; i < players1.length; i++) {
        $('#p'+players1[i][1]).html(players1[i][0])
        players.push(players1)
    }
    console.log(players)
    localStorage.setItem("currentPlayer", JSON.stringify(nbJoueur));
    localStorage.setItem("players", JSON.stringify(players1));
}

```

Ensuite, pour reprendre la partie, on utilise une fonction resumeGame(), défini en PHP. Cette fonction permet de réattribuer les valeurs en JavaScript, d'où l'utilité des localStorage. Au cas où une partie avant a déjà été entreprise, on supprime tout du localStorage, afin de ne pas reprendre une autre partie. Ensuite, on remplace les pions des joueurs sur le plateau et on réattribue le pion et leur position, dans l'array « players ». Ensuite, on remet tout dans le localStorage afin de pouvoir rejouer la partie.

## F/Modifications du code

### A/HTML / PHP

Après avoir fini les fonctions principales du site, je suis repassé sur le code, afin de le rendre plus lisible à lire et retirer les choses inutiles, comme les « echo » et « var\_dump » qui m'ont servi à trouver les erreurs. Je suis aussi repassé sur le plateau, qui n'était pas très beau avec tous les <div>. J'ai donc refait le plateau proprement avec des boucles for. C'était aussi l'occasion d'apprendre du CSS. J'ai appris à utiliser le « display : flex » qui permet de créer des zones flexibles, plus facile à modifier, même si la prise en main a mis un peu de temps à arriver. Grâce à ça, j'ai pu restructurer la plupart des pages. On peut voir dans cette partie du

```
<div class="plateau">
  <div class="containerHaut">
    <div id="0" class="first"><p id="p0"></p></div>
    <?php
      for($i=1;$i<10;$i++) {
        if($i == 5) {
          <div id="5" class="third"><p id="p5"></p></div>
        }
        <?php
          } else {
            <div id="<?php echo($i); ?>" class="second"><p id="p<?php echo($i); ?>"></p></div>
          }
        }
      }
    <div id="10" style="padding-bottom:30px;" class="first"><p id="p10"></p></div>
  </div>
```

code, la génération de la ligne du haut du plateau. J'ai séparé le 0, 5 et 10, ça ce sont des cases qui auront une couleur différente par rapport au reste du tableau dans la version finale et qui contiendront les questions personnelles.

Je me suis dit que ce serait sympa de voir comment mettre en place des cookies, afin de garder l'utilisateur connecté pendant 48h, et le déconnecte si aucune activité à été faite sur le site. Je pensais que ce serait compliqué, mais non. Pour créer un cookie, c'est très simple. Il suffit de faire setcookie(nom, valeur, expiration, chemin, domaine, secure, httponly ?). Une fois ces données saisies, un cookie se crée dans notre navigateur. Pour le récupérer en PHP, rien de plus simple, il suffit de faire \$\_COOKIE['nom']. Grâce à ça, il suffit de regarder si lors de la connexion, le cookie existe encore ou non. Si celui-ci est expiré, on déconnecte l'utilisateur, afin qu'il se reconnecte.

```
setcookie('connect', "connexion", time()+3600*48, '/', '', false, false);
```

Ceci permet d'améliorer la sécurité du site, car si une personne malveillante se connecte sur l'ordinateur, les données des patients peuvent être volées.

```
if(isset($_COOKIE['connect'])) {
```

Sachant que c'est un cookie qui permet seulement l'authentification et non le traçage, nous n'avons pas besoin de créer une case afin d'accepter ou non les cookies. Ce cookie rentre dans la zone « exempté de consentement », défini par la CNIL, car le cookie permet le bon fonctionnement du site. Dans l'idéal, il faut préciser cependant dans les Conditions Générales d'Utilisation, que nous utilisons les cookies, les nommer et donner descriptif.

### B/CSS

Après avoir appris le « display : flex », j'ai pu commencer à retravailler le code de certaines pages ainsi que le CSS. J'ai appris que la structure était importante afin de rendre l'interface simple d'utilisation. Avec mon maître de stage, on a réalisé une maquette grâce à Adobe XD, qui est outil UI/UX, qui permet la conception d'application ou de site web. Cet outil est pratique car on peut créer notre site et avoir une image de ce que nous voulons faire.

J'ai donc commencé à refaire le CSS du site en entier, commençant par la page de connexion, qui était seulement une page blanche avec un login et mot de passe.

Avant / Après:



A simple, light gray login form with the title 'Connexion'. It contains fields for 'LOGIN' and 'MOT DE PASSE', a 'Connexion' button, and a small link 'Mot de passe oublié'.

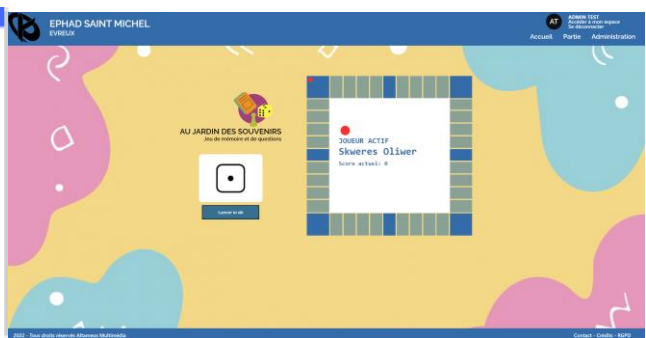
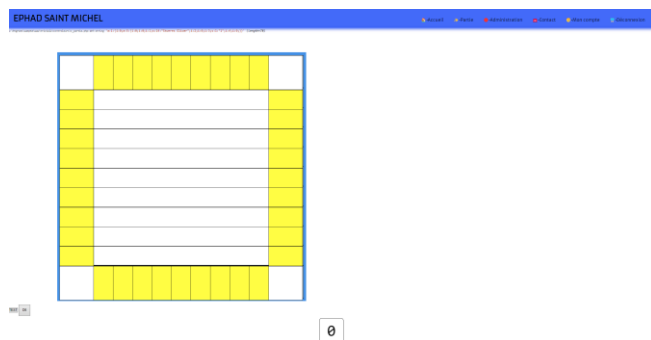


A more colorful and modern login form titled 'IDENTIFICATION'. It features a header 'AU JARDIN DES SOUVENIRS' with the subtitle 'Jeu de mémoire et de questions'. It includes fields for 'EMAIL' and 'MOT DE PASSE', a 'Mot de passe oublié' link, and an 'IDENTIFIER' button. The background has a vibrant, abstract pattern.

On peut voir une grosse différence entre les deux versions. C'est surtout mon maître de stage qui m'a expliqué qu'il fallait le rendre plus abordable aux personnes qui ne savent pas utiliser des ordinateurs, pour leur simplifier la chose.

J'ai aussi refait les pages des questions, le plateau et l'administration. La plus grosse différence pour moi ce fait sur le plateau, car il a beaucoup changé par rapport à la première version.

Avant / Après



Et voici la comparaison d'autres pages :

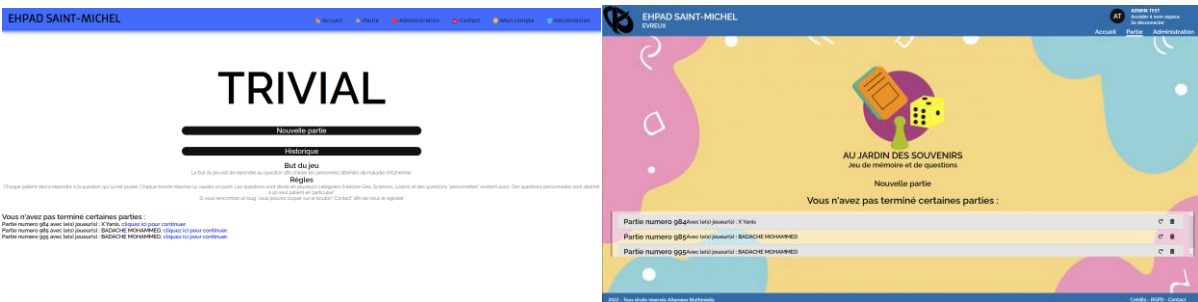
Accueil :



## Administration :



## Partie :



## 5/Bilan

### A/ Bilan du projet

Pour le côté projet, j'ai pu faire tout ce qui m'as été demandé dans les temps, finissant même une semaine en avance, me laissant donc retravailler le CSS. L'hébergement a pu se faire aussi, avec quelques problèmes du côté de l'hébergeur concernant les SESSIONS et COOKIES. A part ça, j'ai trouvé ce projet très sympa à faire, sachant que j'ai pu travailler le JavaScript, que j'avais abordé très peu côté Web, et j'ai aussi pu améliorer le côté graphique de mon site Web, en apprenant des notions que je ne connaissais pas. Lorsque je suis arrivé à la fin de mon projet, je me suis rendu compte que je manquais beaucoup de technique et qu'il il fallait que je travail ça pendant les 2 mois de vacances. J'ai aussi vu en cours de ce projet les frameworks comme Bootstrap et Bulma, qui auraient pu être très utile pour le côté CSS et la librairie React.js pour créer plus facilement des interfaces utilisateurs. La seule chose que je n'ai pas pu faire, c'est adapter le site pour une version mobile, ce que je trouve très dommage.

A titre personnel, je pense avoir tout de même progressé dans le raisonnement et la manière de faire. Je pense pouvoir faire beaucoup mieux pour mon 2<sup>ème</sup> stage ainsi que ma 2<sup>ème</sup> année de BTS SIO. Je suis satisfait de mon travail et je pense que mon maitre de stage aussi.