|  |  |  |
| --- | --- | --- |
| 17/04/2016 |  | Projet annuel : QuickMind |
|  |  |  |

À

[sananes@esgi.fr](mailto:sananes@esgi.fr)

De

[quickmindesgi@gmail.com](mailto:quickmindesgi@gmail.com)

CC

[henryantoine@icloud.com](mailto:henryantoine@icloud.com)   
[l.rodrigues.david@gmail.com](mailto:l.rodrigues.david@gmail.com)  [melodieb1105@yahoo.fr](mailto:melodieb1105@yahoo.fr)

Re

Point du 17 avril 2016

Sujet définitif :

Blind-tests autour de trois catégories : musiques, films et images/photos.

**Fonctionnalités du logiciel :**

* Plusieurs modes de jeu
  + Chaque mode de jeu est accessible depuis la page d’accueil
  + **Mode challenge** : les trois catégories réunies et mélangées
  + Plusieurs niveaux de difficulté
* Une interface joueur personnalisée
  + **Espace joueur** (profil, grade, historiques, nb de points, classements actuels, avatar, association au compte Facebook…)
  + **Les grades** permettent de déverrouiller de nouvelles fonctionnalités (modes de jeu, difficultés supplémentaires, possibilité de soumettre du contenu, etc…)

**Base de données :**



* + Fonctionnalité d’**addition de contenu** par les utilisateurs (soumis à validation ou au vote des autres joueurs)
  + **Connexion aux réseaux sociaux** Facebook, Google+, Twitter (partage des scores, invitations…)
  + **Liens entre joueurs** (recherche en fonction du pseudo, ajout en ami, possibilité de comparer les historiques effectués etc.)

**Remarques et questions :**

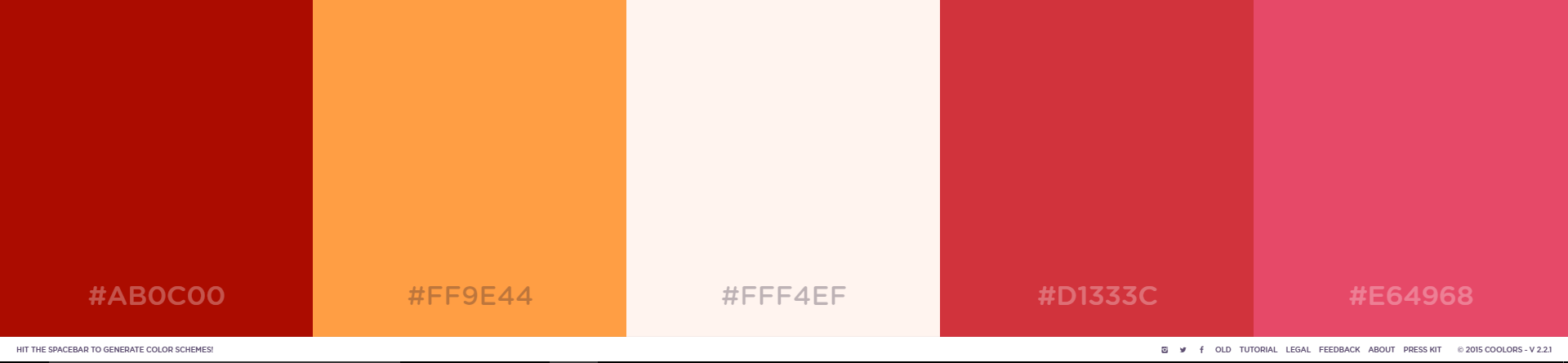
Ajout en ami : système de notification push pour indiquer à l’utilisateur une nouvelle demande

Principe BDD ou JavaScript ?

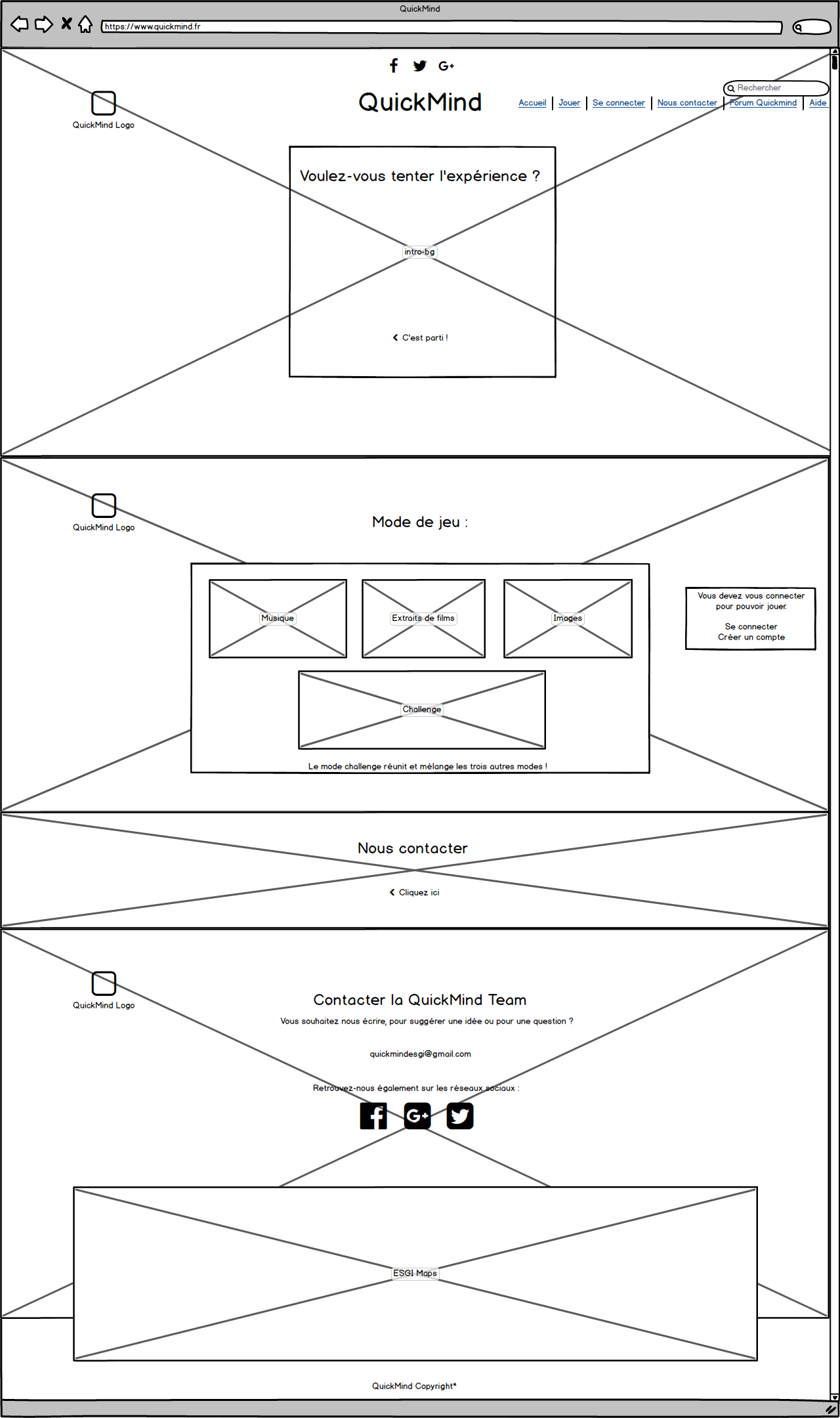
* + **Forum** accessible aux joueurs
* Un système de classement par points
  + Le calcul des points **s’adapte au temps et à la difficulté**.
  + Les points attribuent des grades.
  + **Classements de points inter-joueurs selon catégorie/pays**
  + **Classement général** (total des points cumulés)
* Back-office
  + **Gestion des utilisateurs**
  + **Ajout/Modification/Suppression de contenu**

**Charte graphique :**

Palette de couleurs :



**Bootstrap choisi :** StartBootstrap Grayscale 1.0.6



* **Inscription du joueur** (<https://www.quickmind.fr/subscribe.php>)
* L’utilisateur aura la possibilité de s’inscrire via un formulaire d’inscription HTML, ou en utilisant son compte Facebook, Twitter ou Google.
* Le formulaire HTML contiendra les informations suivantes :
* Pseudo (***pseudo***) : <input type=’text’ name=’pseudo’>
* Nom (**name**) : <input type=’text’ name=’name’>
* Prénom (***surname***) <input type=’text’ name=’surname’>
* Email (***email***) <input type=’email’ name=’email’>
* Sexe (***gender***) <input type=’radio’ name=’gender’> x2
* Pays (***country***) <select name=’country’><option /><option />…</select>
* Conditions Générales d’Utilisation <input type=’checkbox’ name=’cgu’>
* Captcha <?php include ‘captcha.php’ ; ?>
* La validation des champs s’effectuera **en JavaScript** afin de ne pas générer de nouveau la page en cas d’erreur de saisie. Les champs devront respecter les règles suivantes :
* **Pseudo** : 4 caractères minimum, aucun espace, pas encore présent en base.
* **Nom** : Uniquement des lettres, pas d’espaces en début de champ.
* **Prénom** : Uniquement des lettres, différent du nom, pas d’espace en début de champ.
* **Email** : Obligatoirement valide, pas encore présent en base.
* **Pays** : Doit retourner une valeur comprise dans les propositions suggérées.
* **CGU** : Doit être coché.
* **Captcha** : Se régénère à chaque tentative échouée, le captcha sera composé d’une combinaison de 4 chiffres aléatoires.
* Un token est généré grâce à la fonction generateAccessToken() lorsque le formulaire soumis ne comporte aucune erreur. Celui-ci permettra d’identifier l’utilisateur de manière sécurisée. Nous renouvellerons ce token à chaque fois qu’il est utilisé pendant la navigation sur QuickMind.
* La requête SQL suivante permettra d’injecter dans la BDD les informations saisies lors de l’inscription de l’utilisateur.

INSERT INTO users (pseudo, name, surname, email, password, country, accesstoken) VALUES ($\_POST[‘pseudo’], $\_POST[‘name’], $\_POST[‘surname’], $\_POST[‘email’], $\_POST[‘password’], $\_POST[‘country’], $accesstoken) ;

* Une fois l’inscription terminée, un mail est envoyé grâce à la fonction mail() à l’utilisateur afin que celui-ci confirme son inscription via un lien de validation, généré en fonction du nouvel accesstoken de l’utilisateur ( $url = ”<https://www.quickmind.fr/validemail.php?accesstoken=>”.$accesstoken ; ).

$from = "quickmindesgi@gmail.com";

$to = $\_POST['email'];

$subject = "Validation de l'inscription";

$body = "Veuillez cliquer sur ce lien pour activer votre compte. ".$url;

$headers = 'From: '. $from . "\r\n" . 'Reply-To: '. $from . "\r\n" . 'X-Mailer: PHP/' . phpversion();

mail($to, $subject, $body, $headers);

* La confirmation de l’inscription s’effectuera sur la BDD grâce à une requête SQL (**bool *is\_valid*** de 0 à 1) :

<?php require "init.php";

if( !isset($\_GET['accesstoken']) || strlen($\_GET['accesstoken']) != 32 ){

die("error");

}

$db = connectDb();

$query = $db 🡪 prepare( "SELECT pseudo FROM users WHERE accesstoken=:accesstoken");

$query->execute(["accesstoken" => $\_GET['accesstoken']]);

$results = $query 🡪 fetch();

if( !empty($results) ){

$query = $db 🡪 prepare(" UPDATE users SET is\_valid = 1 WHERE accesstoken=:accesstoken");

$query 🡪 execute(["accesstoken" => $\_GET['accesstoken']]);

$\_GET[‘accesstoken’] = generateAccessToken($\_GET[‘accesstoken’]) ;

header("Location: index.php");

}else{

die("error"); } ?>

* **Back-office**

Le back-office est un **espace d’administration** du site QuickMind.

Le back-office est joignable depuis :

<https://www.quickmind.fr/admin>/

* + **Accès sécurisé**
    - Son accès est réservé aux administrateurs du site (**bool *is\_admin***).
    - Vérification du champ ***is\_admin*** de la table USERS en implémentant une fonction PHP **is\_admin()** comprenant la requête SQL. On prendra soin de régénérer un token après utilisation :

<?php require “init.php”;

function is\_admin() {

$error = FALSE ;

$msg\_error = “”;

If(isConnected()){

$db = connectDb() ;

$query = $db 🡪 prepare( “SELECT \* FROM users WHERE is\_admin = :is\_admin AND accesstoken = :mytoken”) ;

$query 🡪 execute([ “is\_admin” => $\_SESSION[‘is\_admin’], “mytoken” => $\_SESSION[‘accesstoken’]]) ;

$data = $query 🡪 fetch() ;

If($data[‘is\_admin’] != 1 {

$error = TRUE ;

$msg\_error .= “Seuls les administrateurs peuvent accéder à cette partie du site.”

}

If( ! error) {

$\_SESSION[‘accesstoken’] = generateAccessToken($\_SESSION[‘pseudo’])

header(“Location: /admin/index.php”) ;

}else{

echo $msg\_error ;

}

}

If( ! isConnected() ) {

$error = TRUE ;

$msg\_error = “Veuillez vous authentifier.” ;

echo $msg\_error ;

}

}

* + - **Authentification sécurisée par mot de passe** (***.htaccess***).

Du côté du serveur Debian, les administrateurs du site sont référencés dans le fichier **/var/www/pwd/admin**, grâce à la méthode **htpasswd**.

* + - Le **.htaccess** propose une authentification classique par mot de passe (***Basic***) pour un utilisateur valide (***valid-user*** présent dans le htpasswd), placé dans le dossier admin contenant le back-office.

<Directory /var/www/QuickMind/admin>

AuthType Basic

AuthName “Accès réservé aux administrateurs de QuickMind”

AuthFile /var/www/pwd/admin

Require valid-user

</Directory>

* + - L’idée, c’est de rediriger l’utilisateur sur la page d’administration lorsqu’il souhaite y accéder **si et seulement s’il est reconnu comme un administrateur et que son couple login/mot de passe est présent dans le fichier** référençant les administrateurs (situé en dehors du site).
  + **Une interface intuitive**

L’accueil du back-office propose à l’administrateur connecté deux grandes fonctionnalités d’administration du site :

* ***Gestion des utilisateurs*** *(https://www.quickmind.fr/admin/users.php)*

Panneau d’administration des utilisateurs du site.

Les utilisateurs sont affichés dans un tableau dynamique appelant la BDD.

<?php require "init.php";

if(isConnected()){

$db = connectDb();

$results = $db 🡪 query("SELECT \* FROM users");

?>

<table>

<tr>

<th>Pseudo</th>

<th>Nom</th>

<th>Prénom</th>

<th>Email</th>

<th>Pays</th>

<th>Date d'insertion</th>

<th>Validé ?</th>

<th>Modifier</th>

<th>Supprimer</th>

</tr>

<?php

while($data = $results 🡪 fetch()){

echo "<tr>";

echo "<td>".$data['pseudo']."</td>";

echo "<td>".$data['name']."</td>";

echo "<td>".$data['surname']."</td>";

echo "<td>".$data['email']."</td>";

echo "<td>".$data['country']."</td>";

echo "<td>".$data['date\_inserted']."</td>";

echo "<td>".$data['status']."</td>";

echo "<td name='modifybutton'><a href='modify.php?ps=".$data['pseudo']."'><button>Modifier</button></td>";

echo "<td name='deletebutton'><a href='delete.php?ps=".$data['pseudo']."'><button>Supprimer</button></td>";

echo "</tr>";

}

echo "</table>";

}

Chaque ligne concerne un unique utilisateur, et propose deux options grâce des boutons :

* + - * **Modifier un utilisateur**

On envoie dans l’URL de modify.php le pseudo de l’utilisateur que l’on souhaite modifier, afin de s’en resservir grâce à la méthode GET. On va ainsi pouvoir récupérer les informations de l’utilisateur rentrées en base afin de les modifier :

if(isset($\_GET['pseudo'])){

$db = connectDb();

$results = $db 🡪 prepare( "SELECT pseudo, name, surname, country FROM users WHERE pseudo=:pseudo");

$results->execute(["pseudo" => $\_GET['pseudo']]);

$\_POST = $results->fetch();

}

On propose à l’administrateur un nouveau formulaire HTML similaire à subscribe.php. Si les champs de mot de passe ne sont pas complétés, le mot de passe est inchangé.