



---

**MU4IN507 - PC3R**

**Micro-Projet Web : Quiz Of Legends**

---

**Rapport - Rendu Final**

**Membres du projet :**

Floria LIM (28706087)  
André VICENTE (3808685)

Année 2023-2024  
Sorbonne Université - Master STL

# SOMMAIRE

<b>1. Description générale de l'application.....</b>	<b>2</b>
<b>1.1. API choisie.....</b>	<b>2</b>
<b>1.2. Fonctionnalités.....</b>	<b>2</b>
<b>1.3. Schéma global du système.....</b>	<b>3</b>
<b>1.4. Cas d'utilisations classiques.....</b>	<b>3</b>
<b>1.5. Maquette des plans du site.....</b>	<b>4</b>
<b>2. Données stockées.....</b>	<b>6</b>
<b>2.1. Liste des différentes collections.....</b>	<b>6</b>
<b>2.2. Mise à jour des données et appel à l'API externe.....</b>	<b>8</b>
<b>3. Description du Serveur.....</b>	<b>9</b>
<b>3.1. Choix de l'approche utilisée.....</b>	<b>9</b>
<b>3.2. Liste des ressources.....</b>	<b>9</b>
<b>3.3. Description des requêtes et réponses Client-Serveur.....</b>	<b>10</b>
<b>4. Description du client.....</b>	<b>13</b>
<b>4.1. Technologies clients utilisées.....</b>	<b>13</b>
<b>4.2. Plan du client et appels AJAX au serveur.....</b>	<b>13</b>
<b>4.3. Intégration CSS grâce à des sources tierces.....</b>	<b>19</b>
<b>5. Hébergement et Scalabilité.....</b>	<b>20</b>
<b>6. Conclusion.....</b>	<b>20</b>
<b>7. Bibliographie.....</b>	<b>21</b>

## 1. Description générale de l'application

Notre application web Quiz of Legends (<http://13.60.55.141:8080/>) propose un quiz interactif basé sur le jeu populaire League Of Legends développé par Riot Games, où les utilisateurs pourront tenter de répondre à des questions par rapport à un match en fonction des informations de ce dernier. En plus du jeu de quiz, les utilisateurs pourront créer un compte sur le site, se connecter et se déconnecter à tout moment. En participant au quiz, les joueurs gagnent des points d'expérience, ce qui leur permettra de monter de niveau sur le site.

### 1.1. API choisie

Nous avons choisi d'utiliser l'API publique de Riot Games, l'éditeur du jeu League Of Legends où la documentation est disponible sur <https://developer.riotgames.com/apis> [7].

Plus précisément, nous avons utiliser les APIs suivantes :

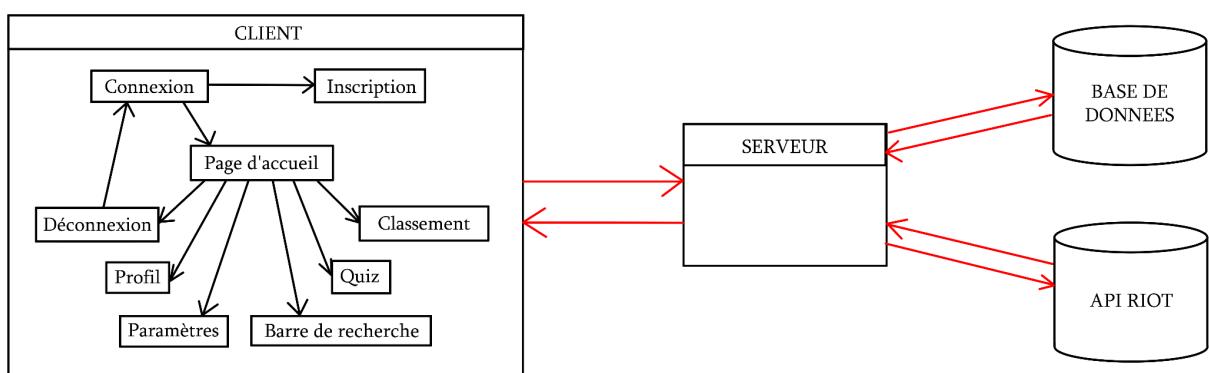
- **ACCOUNT-V1** : elle permet de récupérer l'identifiant unique d'un compte Riot (UUID) à partir du pseudonyme d'un joueur.
- **SUMMONER-V4** : elle permet de récupérer les informations d'un joueur à partir d'un identifiant Riot (UUID), en particulier l'identifiant de l'invocateur (summoner ID), qui sera utilisé pour référencer les joueurs de League of Legends dans les requêtes ultérieures.
- **LEAGUE-V4** : elle est utilisée pour récupérer le rang d'un joueur, ainsi que son nombre de victoires et de défaites.
- **MATCH-V5** : elle permet de fournir des données détaillées sur des matchs de League of legends, y compris les statistiques des joueurs, le résultat du match et d'autres détails pertinents. Nous l'utiliserons pour créer des questions dans le quiz où les utilisateurs devront deviner le dénouement ou le rang moyen d'un match en fonction de ces données du match.

### 1.2. Fonctionnalités

- **Inscription** : Les utilisateurs peuvent créer un compte en fournissant leur nom d'utilisateur, leur pseudo et leur mot de passe. Les informations d'inscription seront stockées dans la base de données.
- **Connexion** : Les utilisateurs enregistrés peuvent se connecter en utilisant leur nom d'utilisateur et leur mot de passe.
- **Déconnexion** : Les utilisateurs connectés peuvent se déconnecter de leur compte.
- **Quiz** : Les utilisateurs peuvent accéder au quiz où ils seront présentés avec des données d'un match de League of legends. Ils doivent répondre à une question sur ce match en fonction des informations fournies. Une fois qu'ils ont soumis leur réponse, l'application vérifie son exactitude et attribue des points d'expériences en fonction de la précision de leur estimation.
- **Profil utilisateur** : Chaque utilisateur aura un profil où son niveau actuel sera affiché. Ils pourront également consulter leur moyenne de réponses correctes, ainsi que leur historique de quiz.

- **Recherche** : Les utilisateurs pourront utiliser la fonctionnalité de recherche pour trouver rapidement le profil d'un autre joueur en saisissant son nom d'utilisateur dans la barre de recherche.
- **Statistiques globales** : L'application affichera les meilleurs joueurs du site en fonction de leur niveau et de leur moyenne de réponses correctes.
- **Paramètres** : L'application offre une page de paramètres permettant aux utilisateurs de modifier leur pseudo et leur mot de passe, ainsi que supprimer leur compte.

### 1.3. Schéma global du système



### 1.4. Cas d'utilisations classiques

- **Inscription** : L'utilisateur remplit le formulaire d'inscription en fournissant un pseudo, un nom d'utilisateur et un mot de passe. Après avoir créé un compte, il se connecte via le formulaire de connexion, puis il est redirigé vers la page d'accueil où il peut commencer un quiz ou bien consulter son profil.
- **Jouer au quiz** : L'utilisateur se connecte et est redirigé vers la page d'accueil. Il clique sur le bouton de quiz pour se diriger vers la page de quiz. Il répond à une question du quiz en sélectionnant sa réponse puis la soumet. Il peut ensuite voir si sa réponse est correcte, et peut choisir de rejouer ou de revenir à la page d'accueil.
- **Consulter le profil** : L'utilisateur se connecte et est redirigé vers la page d'accueil. Il accède ensuite à son profil pour voir son niveau et sa moyenne de réponses correctes.
- **Consulter le profil d'un autre utilisateur** : L'utilisateur se connecte et est redirigé vers la page d'accueil. Il consulte le profil d'un autre utilisateur en recherchant son pseudonyme dans la barre de recherche. En visitant le profil d'un autre joueur, il peut voir son niveau et sa moyenne de réponses correctes.
- **Consulter le classement** : L'utilisateur se connecte et est redirigé vers la page d'accueil où est affichée une version miniature du classement des utilisateurs en fonction de leur niveau. Il accède ensuite au classement entier depuis la page d'accueil pour voir liste complète des meilleurs utilisateurs.

- **Modifier les paramètres du compte :** L'utilisateur se connecte et est redirigé vers la page d'accueil. Il accède ensuite à la page des paramètres où il a la possibilité de modifier son pseudo, son mot de passe ou de supprimer son compte.
- **Déconnexion :** L'utilisateur se connecte et est redirigé vers la page d'accueil. L'utilisateur peut se déconnecter à tout moment en cliquant sur le bouton de déconnexion, où il est redirigé vers la page de connexion pour se reconnecter.

## 1.5. Maquette des plans du site

Voici les maquettes des écrans du site que nous avons réalisées lors du début de la conception de notre application pour avoir une idée de l'aspect visuel et les fonctionnalités de l'application :

- [Ecran de connexion et d'inscription](#)

QUIZZ	QUIZZ
<b>Login</b> Nom d'utilisateur : <input type="text"/> Mot de passe : <input type="password"/> <input type="button" value="OK"/> <input type="button" value="Créer un compte"/>	<b>Inscription</b> Pseudo : <input type="text"/> Nom d'utilisateur : <input type="text"/> Mot de passe : <input type="password"/> Confirmation : <input type="text"/> <input type="button" value="OK"/> <input type="button" value="Déjà inscrit ?"/>

Ces écrans permettent aux utilisateurs de se connecter à leur compte en saisissant leur nom d'utilisateur et leur mot de passe, ou bien de s'inscrire pour les nouveaux utilisateurs en fournissant un nom d'utilisateur unique et un mot de passe.

- [Écran d'accueil](#)

Déconnexion	QUIZZ	Profil
<input type="text"/> @		
Leader Board 1. Farminator 2. Yellow 3. Eldritch	<input type="button" value="JOUER"/>	/ 
<input type="button" value="Paramètres"/>		

Sur cet écran, les utilisateurs sont accueillis avec plusieurs options. Ils peuvent lancer un quiz pour commencer à jouer immédiatement, accéder à leur profil pour consulter leurs statistiques et leur historique de jeu, ou se déconnecter de leur compte. Aussi, un petit classement des meilleurs joueurs peut être affiché à gauche, permettant aux utilisateurs de voir les meilleurs joueurs du site. Une barre de recherche est également disponible pour permettre aux utilisateurs de rechercher des profils de joueurs spécifiques.

- Écran de quiz

Déconnexion	QUIZZ	Profil																																			
Quelle équipe a gagné ?																																					
<table border="1"> <thead> <tr> <th>Equipe bleue</th> <th>KDA</th> <th>Objets</th> </tr> </thead> <tbody> <tr><td><input type="radio"/> Adversary</td><td>-1</td><td></td></tr> <tr><td><input type="radio"/> Azeral</td><td>383</td><td></td></tr> <tr><td><input type="radio"/> El Julio</td><td>4.2</td><td></td></tr> <tr><td><input type="radio"/> Yellow</td><td>999</td><td></td></tr> <tr><td><input type="radio"/> Aykumir</td><td>-999</td><td></td></tr> </tbody> </table>	Equipe bleue	KDA	Objets	<input type="radio"/> Adversary	-1		<input type="radio"/> Azeral	383		<input type="radio"/> El Julio	4.2		<input type="radio"/> Yellow	999		<input type="radio"/> Aykumir	-999		<table border="1"> <thead> <tr> <th>Equipe rouge</th> <th>KDA</th> <th>Objets</th> </tr> </thead> <tbody> <tr><td><input type="radio"/> Random1</td><td>1.0</td><td></td></tr> <tr><td><input type="radio"/> Random2</td><td>1.0</td><td></td></tr> <tr><td><input type="radio"/> Random3</td><td>1.0</td><td></td></tr> <tr><td><input type="radio"/> Random4</td><td>1.0</td><td></td></tr> <tr><td><input type="radio"/> Random5</td><td>1.0</td><td></td></tr> </tbody> </table>	Equipe rouge	KDA	Objets	<input type="radio"/> Random1	1.0		<input type="radio"/> Random2	1.0		<input type="radio"/> Random3	1.0		<input type="radio"/> Random4	1.0		<input type="radio"/> Random5	1.0	
Equipe bleue	KDA	Objets																																			
<input type="radio"/> Adversary	-1																																				
<input type="radio"/> Azeral	383																																				
<input type="radio"/> El Julio	4.2																																				
<input type="radio"/> Yellow	999																																				
<input type="radio"/> Aykumir	-999																																				
Equipe rouge	KDA	Objets																																			
<input type="radio"/> Random1	1.0																																				
<input type="radio"/> Random2	1.0																																				
<input type="radio"/> Random3	1.0																																				
<input type="radio"/> Random4	1.0																																				
<input type="radio"/> Random5	1.0																																				
<input type="button" value="1. Bleue"/>		<input type="button" value="2. Rouge"/>																																			
Paramètres																																					

Cet écran présente une question à l'utilisateur. Il affiche les informations pertinentes sur un match de League of Legends, suivie de plusieurs choix de réponses parmi lesquels l'utilisateur doit sélectionner celle qu'il estime être correcte.

- Écran de profil

Déconnexion	QUIZZ	Profil
 <p>Azeral Niveau 99 Réponses correctes : 1000 Réponses total : 1001 Précision : 99%</p>		
Paramètres		

Sur cet écran, les utilisateurs peuvent consulter les détails de leur profil. Cela inclut leur pseudonyme, leur niveau actuel, leur moyenne de réponses correctes et d'autres informations pertinentes liées à leur expérience de jeu sur l'application.

- Écran LeaderBoard (classement)

Déconnexion	QUIZZ	Profil
<div style="border: 1px solid black; padding: 10px;"> <h3>Leader Board</h3> <ul style="list-style-type: none"> <li>1. Farminator</li> <li>2. Yellow</li> <li>3. Eldritch</li> <li>4. El Julio</li> <li>5. Aykumir</li> <li>6. Azeral</li> <li>...</li> </ul> </div>		
Paramètres		

Cet écran affiche la liste des X premiers utilisateurs ayant obtenu en fonction de leur niveau et de leur nombre de réponses correctes.

- Écran des paramètres

Déconnexion	QUIZZ	Profil
<input type="button" value="Modifier le pseudo"/> <input type="button" value="Modifier le mot de passe"/> <input type="button" value="Supprimer le compte"/>		
Paramètres		

Cet écran permet aux utilisateurs de modifier les données de leur compte, tel que le pseudonyme et le mot de passe. Il offre également la possibilité de supprimer le compte de l'utilisateur de la base de données.

## 2. Données stockées

Nous avons choisi d'utiliser MongoDB [9] comme base de données pour notre application. MongoDB est une base de données NoSQL orientée document, connue pour sa flexibilité et sa capacité à gérer de grandes quantités de données non structurées. Nous avons opté pour MongoDB car nous étions déjà familiers avec cette technologie, elle est gratuite et facile à utiliser, ce qui en fait une solution pratique pour notre projet.

Notre base de données, nommée “DB”, stocke les données des utilisateurs de notre application, mais également les données de plusieurs joueurs de League of Legends et de leurs matchs qui apparaîtront dans les questions de notre quiz, agissant ainsi comme un cache pour limiter les requêtes vers l'API RIOT.

### 2.1. Liste des différentes collections

**Collection users** : Stocke les données d'un utilisateur

- **\_id** (string) : identifiant unique dans la base de données
- **pseudo** (string) : pseudonyme de l'utilisateur
- **username** (string) : nom d'utilisateur
- **password** (string) : mot de passe de l'utilisateur
- **token** (string) : token d'authentification pour authentifier l'utilisateur lorsqu'il s'est connecté
- **experience** (int) : points d'expérience accumulés par l'utilisateur
- **correctAnswers** (int) : nombre total de réponses correctes répondues
- **nbAnswers** (int) : nombre total de questions répondues
- **picture** : image de profil de l'utilisateur

➔ **Exemple d'entrée :**

```
{  
    "_id": {"$oid": "661842c774ea5f52e38b58e9"},  
    "pseudo": "Stinky Waccoon",  
    "username": "stinkywaccoon",  
    "password": "azertyuiop",  
    "token": "2024-04-12T15:05:23Zstinkywaccoon",  
    "experience": {"$numberInt": "925"},  
    "correctAnswers": {"$numberInt": "68"},  
    "nbAnswers": {"$numberInt": "90"},  
    "picture": "fizz.png"  
}
```

**Collection players** : Stocke les données d'un joueur de League Of Legends

- **\_id** (string) : identifiant unique dans la base de données
- **id** (string) : identifiant de l'invocateur (summoner ID)
- **accountid** (string) : identifiant unique régional du compte Riot

- **puuid** (string) : identifiant unique global du compte Riot (PUUID)
  - **pseudo** (string) : pseudonyme du joueur
  - **rank** (string) : rang du joueur
  - **matchIDList** (List[string]) : liste des identifiants des derniers matchs du joueur
- Exemple d'entrée :

```
{
    "_id": {"$oid": "6604235651e615bb8c71b05a"},
    "id": "v5sHuv_svlnFJAry8OBPFeKwLGtkf8DeLWJ_jlA5FFGzGOo",
    "accountid": "N_A9K0sgXbkyy_vgq3lbB0UgXbo57B0oUNNPIMliVLPDIA",
    "puuid": "o2RVmT9OvP6Glm1YSptsL2fQHV43_n5p31JqRjL3dfj_e6yhg-cvgU4n3nmLYfj
        kfgEJzy80pbg2eA",
    "pseudo": "Azeral",
    "rank": "PLATINUM I",
    "matchIDList": ["EUW1_6858526396", "EUW1_6827474512", "EUW1_6827401533"]
}
```

**Collection matchs** : Stocke les données d'un match de League Of Legends

- **\_id** (string) : identifiant unique dans la base de données
  - **metadata** (object) : contient l'identifiant du match et des participants
  - **info** (object) : contient toutes les informations du match (duration, mode, performances de chaque participant, gagnant...)
- Exemple d'entrée :

```
{
    "_id": {"$oid": "660600f028a68c6c28623b77"},
    "metadata":
    {
        "dataversion": "2",
        "matchid": "EUW1_6827474512",
        "participants": (...),
        ...
    },
    "info":
    {
        "gamecreation": {"$numberLong": "1708637287898"},
        "gameduration": {"$numberLong": "1969"},
        ...
        "gametype": "MATCHED_GAME",
        "gameversion": "14.4.561.3953",
        "participants": (...),
        "mapid": {"$numberInt": "11"},
        ...
    }
}
```

## 2.2. Mise à jour des données et appel à l'API externe

Pour mettre à jour les données et appeler l'API externe, nous avons dû gérer les limitations strictes imposées par l'API de League of Legends. En effet, afin de pouvoir réaliser une question de notre quiz, nous devons récupérer les informations d'un match ainsi que le rang des 10 joueurs présents dans ce dernier. Ainsi, la récupération de toutes les données nécessaires pour un match demande au total 11 requêtes vers l'API (1 pour le match, 10 pour tous les participants du match). Étant limités à 20 requêtes par seconde et 50 requêtes toutes les deux minutes, ce quota peut être rapidement atteint.

Pour éviter toute surcharge, nous avons mis en place une stratégie de mise en cache des données dans notre base de données : nous maintenons une liste de joueurs, qui apparaîtront dans les questions de notre quiz, et stockons leurs matchs les plus récents dans notre base de données. Toutes les minutes, la base de données sélectionne un joueur au hasard parmi cette liste et vérifie si son historique de matchs a changé (c'est-à-dire s'il a joué un nouveau match). Si un nouveau match est détecté, il est ajouté à la base de données.

Notre système de mise à jour des données fonctionne comme suit :

- On choisit un des joueurs présent dans notre base de donnée.
- On récupère la liste des derniers matchs du joueur : requête sur **MATCH-V5**.
- Si on détecte un nouveau match qui n'est pas présent dans le cache :
  - ➔ On ajoute le nouveau match dans la base de données.
  - ➔ On récupère le rang de tous les joueurs du match avec **LEAGUE-V4**, que l'on ajoute dans le champ "rank" de la liste "participants" situé dans "info" de la collection matchs.

## 3. Description du Serveur

### 3.1. Choix de l'approche utilisée

Pour la conception de notre serveur, nous avons opté pour une approche basée sur les ressources plutôt que sur les services, ce qui correspond bien à notre architecture RESTful où chaque ressource représente une entité dans notre système, telle qu'un utilisateur ou un match. Cette approche nous permet de structurer notre API de manière claire et cohérente, en utilisant des routes et des endpoints spécifiques pour chaque ressource.

De plus, nous avons choisi de développer notre serveur en suivant une architecture monopage. Cela signifie que l'ensemble de l'application côté serveur est conçu pour servir une seule page web à nos clients, qui interagissent ensuite avec l'API via des requêtes HTTP. Cette approche simplifie la gestion des routes et des états côté serveur, tout en offrant une expérience utilisateur homogène et fluide.

Notre serveur a été implémenté en utilisant le langage de programmation Go et le package standard `net/http`. Pour commencer notre serveur, nous avons repris le template fourni par MongoDB [1] pour connecter une application Go à un cluster MongoDB. Cette base nous a permis d'établir rapidement une connexion stable et sécurisée à notre base de données, facilitant ainsi le développement de notre API RESTful.

### 3.2. Liste des ressources

- **Utilisateur (/api/user) :**

Cette ressource gère l'ensemble des opérations relatives aux utilisateurs de l'application. Elle permet la création de nouveaux comptes, la connexion et la déconnexion des utilisateurs, la modification des informations d'un utilisateur, la récupération des détails des utilisateurs ainsi que la suppression de comptes. Cette ressource est essentielle pour afficher les profils des utilisateurs et générer le classement des utilisateurs de l'application en fonction de leur niveau.

- **Match (/api/match) :**

Cette ressource gère les données des matchs enregistrés. Elle est utilisée pour sélectionner aléatoirement un match à intégrer dans les questions du quiz, garantissant ainsi la diversité des questions.

- **Quiz (/api/quiz) :**

Cette ressource gère les réponses envoyées par l'utilisateur. Elle vérifie la justesse des réponses pour attribuer des points d'expérience en conséquence. Elle met également à jour les données de l'utilisateur, notamment leur nombre de réponses correctes et répondues.

### 3.3. Description des requêtes et réponses Client-Serveur

- **Endpoints de l'API utilisateur :**

Nom de l'endpoint	URL	Description du service	Paramètres en entrée	Format de sortie	Erreurs possibles
CreateUser	PUT /api/user/createUser	Créer un nouveau utilisateur dans la base de données	pseudo (string), username (string), password (string)	{           "status": int,           "message": string,           "InsertedID": string         }	405 : Méthode non autorisée 400 : Nombre de paramètres incorrect 409 : Nom d'utilisateur déjà utilisé 500 : Erreur interne du serveur
GetAll Users	GET /api/user/getall/	Récupération de tous les utilisateurs de l'application	—	[           {             "ID": string,             "Pseudo": string,             ...           }, ...         ]	405 : Méthode non autorisée 500 : Erreur interne du serveur
GetUser Token	GET /api/user/getUserToken	Permet de récupérer les informations d'un utilisateur à partir de son token d'authentification	Le token d'authentification est transmis dans l'en-tête “Authorization”	{           "ID": string,           "Pseudo": string,           "Username": string,           "Password": string,           "Token": string,           "Experience": int,           "CorrectAnswers": int,           "NbAnswers": int,           "Picture": string         }	405 : Méthode non autorisée 401 : Token manquant 404 : Utilisateur non trouvé
Login	POST /api/user/login	Connexion d'un utilisateur et création d'un token d'authentification	username(string), password (string)	{           "status": int,           "message": string,           "userID": string,           "token": string         }	405 : Méthode non autorisée 400 : Nombre de paramètres incorrect 401 : Mauvais utilisateur ou mot de passe 500 : Erreur interne du serveur

<b>Logout</b>	POST /api/user/ logout/ {userid}	Déconnecte un l'utilisateur et supprime son token	—	{         "status": int, "message": string     }	405 : Méthode non autorisée 500 : Erreur interne du serveur
<b>SetPicture</b>	POST /api/user/ setPicture	Met à jour la photo de profil d'un utilisateur	userID (string), picture (string)	{         "status": int, "message": string     }	405 : Méthode non autorisée 500 : Erreur interne du serveur
<b>UpdateUser Pseudo</b>	PUT /api/user/ change Pseudo/ {userid}	Met à jour le pseudo d'un utilisateur	newPseudo (string)	{         "status": int, "message": string     }	405 : Méthode non autorisée 500 : Erreur interne du serveur
<b>UpdateUser Password</b>	PUT /api/user/ change Password/ {userid}	Met à jour le mot de passe d'un utilisateur	newPassword (string)	{         "status": int, "message": string     }	405 : Méthode non autorisée 500 : Erreur interne du serveur
<b>DeleteUser</b>	DELETE /api/user/ deleteUser/ {userid}	Supprime un utilisateur de la base de données	—	{         "status": int, "message": string     }	405 : Méthode non autorisée 500 : Erreur interne du serveur

- Endpoints de l'API match :

Nom de l'endpoint	URL	Description du service	Paramètres en entrée	Format de sortie	Erreurs possibles
<b>GetRandom Match</b>	GET /api/match/ getRandom Match	Récupère un match aléatoire dans la base de données	—	{         "metadata": {...} "info": {...}     }	405 : Méthode non autorisée 500 : Erreur interne du serveur

- **Endpoints de l'API quiz :**

Nom de l'endpoint	URL	Description du service	Paramètres en entrée	Format de sortie	Erreurs possibles
<b>Reponse1</b>	POST /api/quiz/reponse1	Envoie la réponse de l'utilisateur pour une question de type "Quelle équipe a gagné ?"	idUser (string), idMatch (string), winningTeam Number (int)	{           "status": int,           "message": string,           "correctAnswer Increment": int,           "experience Increment": int,           "winGuess": boolean,         }	405 : Méthode non autorisée 404 : Match non trouvé 500 : Erreur interne du serveur
<b>Reponse2</b>	POST /api/quiz/reponse2	Envoie la réponse de l'utilisateur pour une question de type "Quel est le rang moyen de cette partie ?"	idUser (string), idMatch (string), averageRank (string)	{           "status": int,           "message": string,           "correctAnswer Increment": int,           "correctRankAnswer": string,           "experience Increment": int,           "rankGuess": boolean,           "rankProximityScore": int         }	405 : Méthode non autorisée 404 : Match non trouvé 500 : Erreur interne du serveur

## 4. Description du client

### 4.1. Technologies clients utilisées

Pour notre application, nous avons choisi d'utiliser React [8], une bibliothèque JavaScript populaire pour la construction d'interfaces utilisateur. React possède une approche basée sur les composants, ce qui permet de créer des éléments modulaires et réutilisables à différents endroits de l'application, facilitant ainsi le développement et la maintenance de l'application.

React propose également des hooks tels que useState et useEffect pour gérer l'état et les effets de bord dans les composants fonctionnels. Nous utilisons ces hooks pour gérer l'état local de chaque composant, comme les données des utilisateurs et d'un match.

Nous bénéficions également d'un rechargement automatique du client grâce à React, ce qui nous permet de voir instantanément les modifications apportées au code dans le navigateur sans avoir à rafraîchir manuellement la page, accélérant ainsi considérablement l'efficacité du développement.

En plus de React, nous utilisons également Axios pour effectuer des requêtes HTTP, facilitant ainsi l'interaction avec notre backend et l'API RIOT. Axios simplifie la gestion des requêtes asynchrones et le traitement des réponses serveur car il gère automatiquement les en-têtes de requête, les corps de requête et de réponse, et offre des fonctionnalités avancées telles que les intercepteurs de requêtes et de réponses pour la gestion centralisée des erreurs et des authentifications.

En combinant ces technologies, nous avons créé une application moderne et réactive. Elle offre une expérience utilisateur fluide tout en garantissant une bonne interaction avec notre backend.

### 4.2. Plan du client et appels AJAX au serveur

Nous avons structuré notre application en plusieurs composants React, chacun représentant une partie spécifique de l'interface utilisateur.

- **Composant MainPage**

Le composant MainPage est le composant principal de notre application, regroupant toutes les pages et gérant leur transition. Il se charge de récupérer et de stocker les données requises pour les différentes pages de l'application. En plus de cela, il possède une entête qui s'affiche lorsque l'utilisateur est connecté qui lui permet de naviguer vers le profil, la page d'accueil et de se déconnecter facilement. Lorsque l'application démarre, le composant vérifie si l'utilisateur est déjà connecté en vérifiant si un token existe dans le stockage local. Ensuite, la fonction getConnected est appelée pour récupérer les informations de l'utilisateur connecté. Le composant MainPage définit également d'autres fonctions pour gérer les interactions avec l'utilisateur :

- setLogout permet à l'utilisateur de se déconnecter et de supprimer le token du stockage local.
- getAllUsers récupère tous les utilisateurs. Cette fonction est utilisée dans plusieurs pages pour obtenir la liste des utilisateurs.

- changePage permet de changer la page affichée en modifiant l'état de la variable page.
- calculateLevel est utilisée pour calculer le niveau de l'utilisateur en fonction de son expérience. Elle est utilisée dans plusieurs pages qui nécessitent un affichage du niveau d'un utilisateur.

Appels AJAX effectués au serveur :

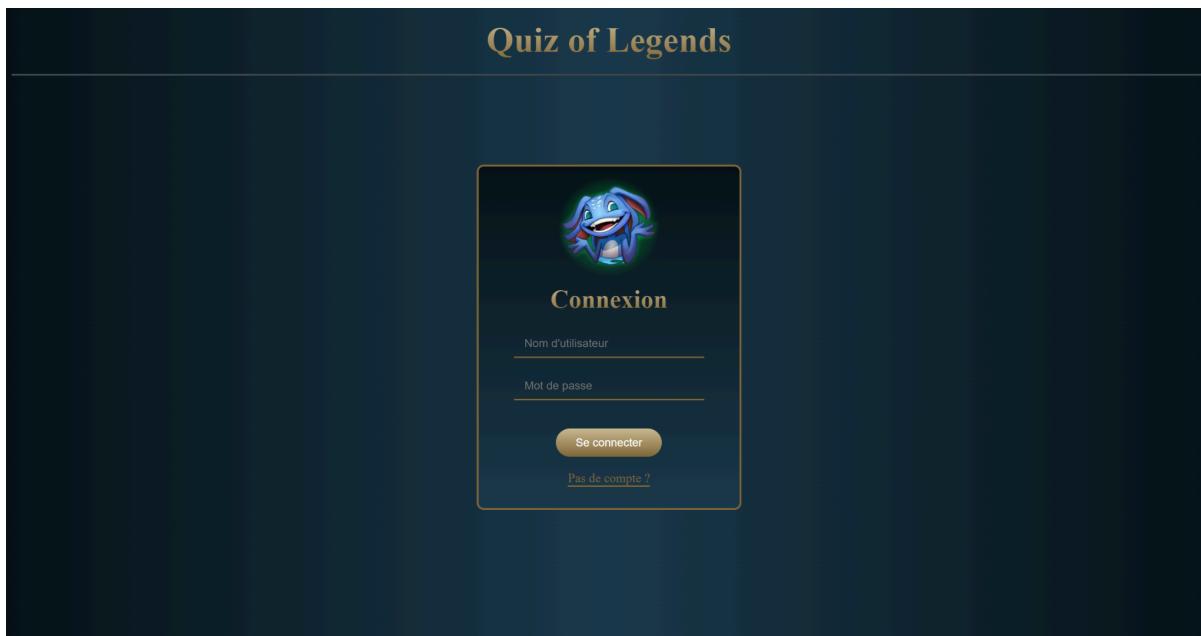
- **/api/user/getUserToken** pour vérifier si un token d'authentification existe et connecter automatiquement l'utilisateur si c'est le cas.
- **/api/user/logout/{userid}** pour gérer la déconnexion de l'utilisateur en cours.
- **/api/user/getall/** pour récupérer la liste de tous les utilisateurs de l'application

- **Composant Login**

Le composant Login représente la page de connexion de l'application. Lorsque l'utilisateur saisit son nom d'utilisateur et son mot de passe, des fonctions sont appelées pour mettre à jour les états correspondants. Ensuite, lorsque l'utilisateur soumet le formulaire de connexion, la fonction submissionHandler est déclenchée. Cette fonction vérifie d'abord si les champs sont remplis. Si les champs sont remplis, une requête AJAX est envoyée au serveur pour tenter de connecter l'utilisateur. En cas de succès, l'utilisateur est redirigé vers la page principale de l'application. En cas d'échec, des messages d'erreur appropriés sont affichés.

Appels AJAX effectués au serveur :

- **/api/user/login** pour gérer la connexion de l'utilisateur.

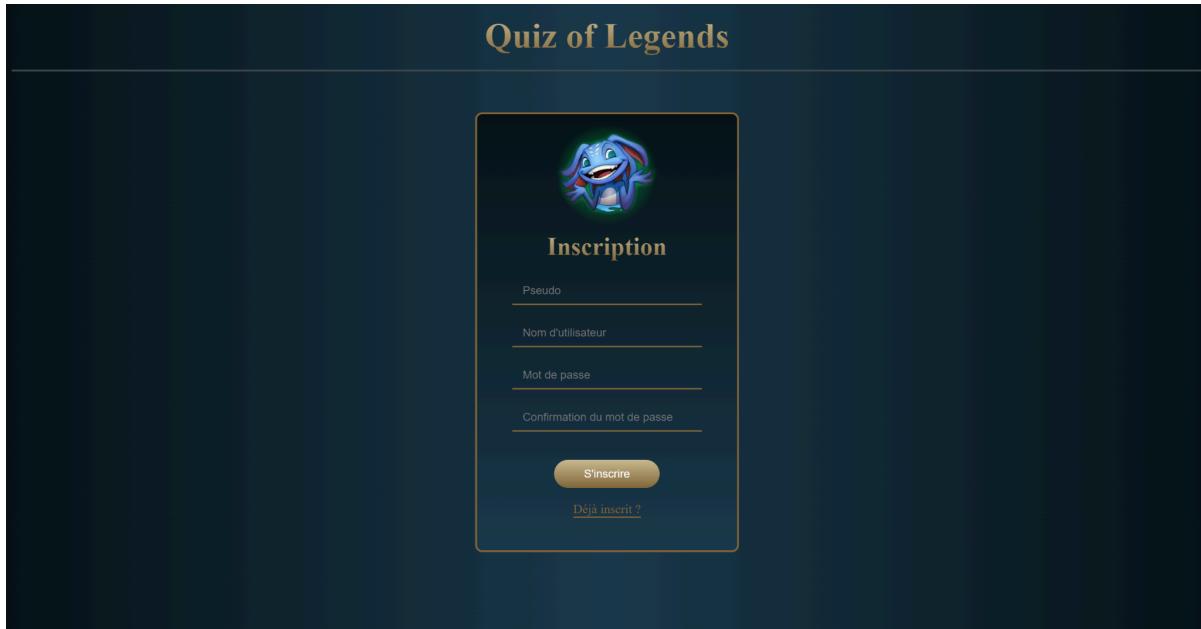


- **Composant Signin**

Le composant Signin représente la page d'inscription de l'application. Similairement au composant Login, la fonction submissionHandler s'assure que les informations saisies par l'utilisateur sont correctes avant de procéder à l'inscription. En cas d'erreur, des messages appropriés sont affichés à l'utilisateur. En cas de succès, l'utilisateur est redirigé vers la page de connexion.

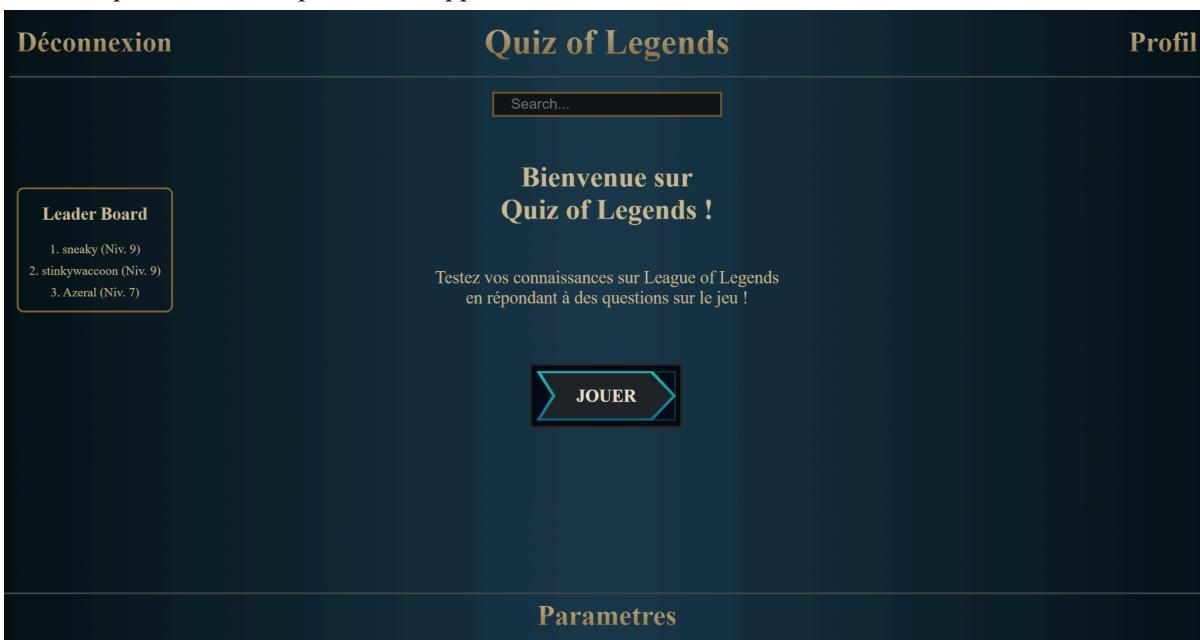
Appels AJAX effectués au serveur :

→ `/api/user/createUser` pour gérer l'inscription d'un nouvel utilisateur.



- Composant HomePage

Le composant HomePage affiche la page d'accueil de l'application. Il comprend une barre de recherche (SearchBar), une version miniature du classement des utilisateurs (LeaderBoard), un bouton pour accéder au quiz et un accès aux paramètres (Paramètres). Contrairement aux autres composants, HomePage n'effectue pas d'appels AJAX au serveur pour récupérer des données, car il s'agit principalement d'une page statique destinée à fournir des informations générales et des fonctionnalités d'accès rapide aux autres parties de l'application.


 A screenshot of the Quiz of Legends homepage. At the top, there are navigation links for "Déconnexion", "Quiz of Legends" (the main title), and "Profil". Below the title is a search bar with placeholder text "Search...". To the left, there is a "Leader Board" section showing three entries: 1. sneaky (Niv. 9), 2. stinkywaccoon (Niv. 9), and 3. Azeral (Niv. 7). The central part of the page features a welcome message "Bienvenue sur Quiz of Legends !" and a call-to-action button "JOUER". At the bottom, there is a "Paramètres" link.

- **Composant SearchBar**

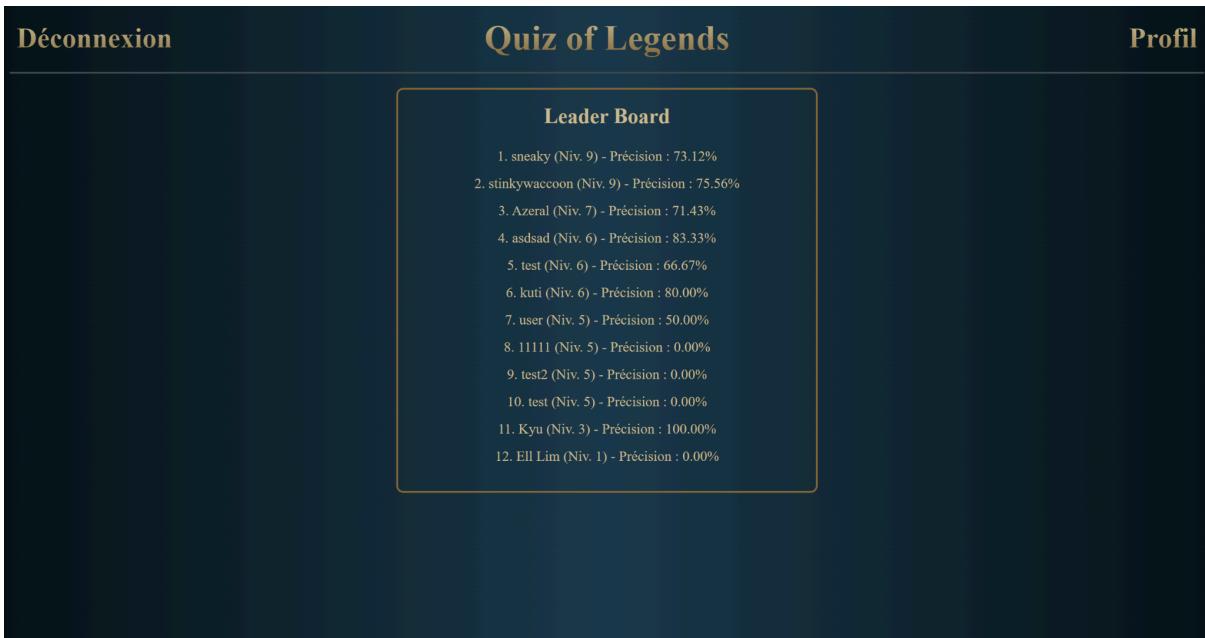
Le composant SearchBar correspond à la barre de recherche de l'application. Il permet à l'utilisateur de rechercher d'autres utilisateurs par leur pseudo ou leur nom d'utilisateur à mesure que l'utilisateur saisit du texte. Les résultats de la recherche sont affichés sous forme d'une liste, et lorsque l'utilisateur clique sur un résultat, le composant change de page pour afficher le profil de l'utilisateur sélectionné.

Appels AJAX effectués au serveur :

- `/api/user/getall/` pour récupérer la liste de tous les utilisateurs de l'application et la filtrer en fonction du texte saisi par l'utilisateur dans la barre de recherche.

- **Composant LeaderBoard**

Le composant LeaderBoard affiche le classement des utilisateurs de l'application. Les utilisateurs sont ensuite triés en fonction de leur nombre d'expérience et de leur précision dans les réponses données. Le composant LeaderBoard possède deux versions : une version complète et une version miniature. Lorsqu'il est utilisé sur la page d'accueil, la version miniature est activée, ce qui permet d'afficher uniquement les trois premiers utilisateurs du classement. L'utilisateur peut visiter le profil d'un utilisateur du LeaderBoard en cliquant sur l'un parmi la liste.



The screenshot shows a dark-themed user interface with three main navigation tabs at the top: "Déconnexion" (Logout), "Quiz of Legends" (highlighted in yellow), and "Profil". The central content area is titled "Leader Board" and displays a list of 12 users with their names and precision percentages:

Rank	User	Precision (%)
1.	sneaky (Niv. 9)	73.12%
2.	stinkywaccoon (Niv. 9)	75.56%
3.	Azeral (Niv. 7)	71.43%
4.	asdsad (Niv. 6)	83.33%
5.	test (Niv. 6)	66.67%
6.	kuti (Niv. 6)	80.00%
7.	user (Niv. 5)	50.00%
8.	11111 (Niv. 5)	0.00%
9.	test2 (Niv. 5)	0.00%
10.	test (Niv. 5)	0.00%
11.	Kyu (Niv. 3)	100.00%
12.	Ell Lim (Niv. 1)	0.00%

Appels AJAX effectués au serveur :

- `/api/user/getall/` pour récupérer la liste de tous les utilisateurs de l'application et effectuer le classement.

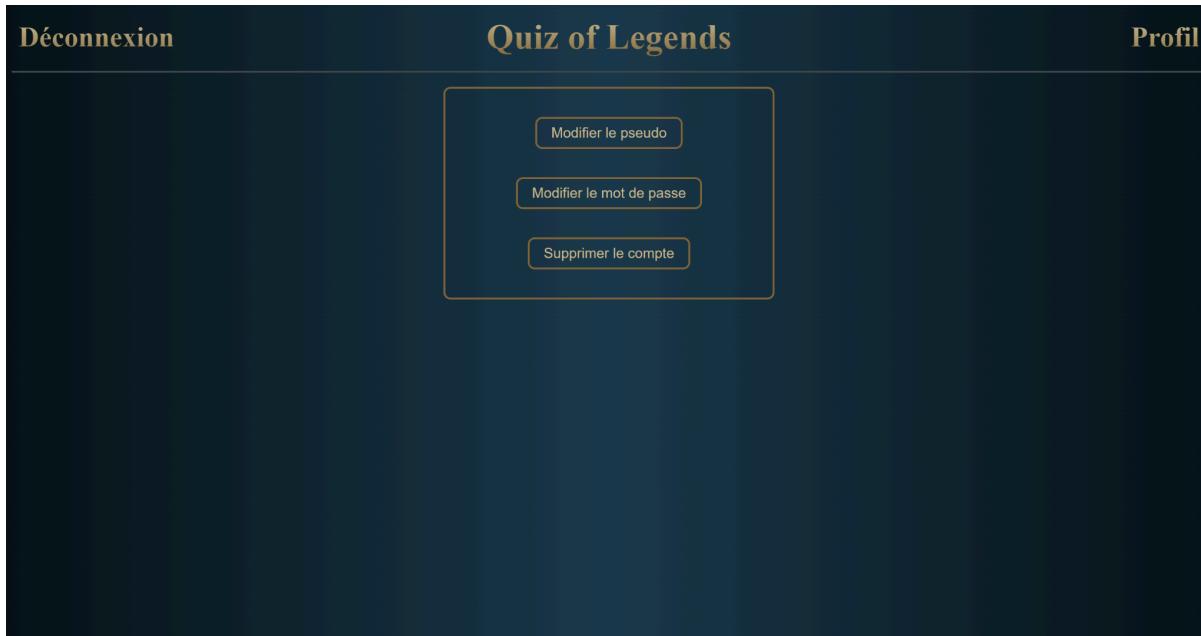
- **Composant Paramètres**

Le composant Paramètres offre à l'utilisateur la possibilité de modifier son pseudo, son mot de passe ou de supprimer son compte. Pour cela, l'utilisateur doit entrer le mot de passe de son compte pour pouvoir effectuer chaque modification. Des vérifications sont effectuées pour s'assurer que les champs sont remplis et corrects.

Appels AJAX effectués au serveur :

- `/api/user/changePseudo/{userid}` pour mettre à jour le pseudo de l'utilisateur.

- **/api/user/changePassword/{userid}** pour mettre à jour le mot de passe de l'utilisateur.
- **/api/user/deleteUser/{userid}** pour supprimer le compte de manière permanente de la base de données.



- **Composant QuestionPage**

Le composant QuestionPage est la page de quiz de l'application, où les utilisateurs répondent à des questions sur les matchs de League of Legends. Le composant utilise des données dynamiques pour afficher les détails du match, tels que la file de jeu, la carte, la durée du match, les objectifs et les statistiques des participants. Il propose deux types de questions : la première question concerne l'équipe gagnante du match, tandis que la deuxième question porte sur le rang moyen des joueurs dans la partie. Une fois avoir répondu à la question, l'utilisateur peut choisir de passer à la question suivante ou de revenir à l'écran d'accueil.

Appels AJAX effectués au serveur :

- **/api/match/getRandomMatch** pour récupérer les données d'un match aléatoire qu'on utilisera pour produire la question.
- **/api/quiz/reponse1** pour répondre à une question sur le déroulement du match et met à jour les statistiques de l'utilisateur en fonction de la justesse de la réponse.
- **/api/quiz/reponse2** pour répondre à une question sur le rang moyen de la partie et met à jour les statistiques de l'utilisateur en fonction de la précision de la réponse.

Appels AJAX effectués vers des services tiers :

- [https://ddragon.leagueoflegends.com/cdn/14.7.1/data/en\\_US/summoner.json](https://ddragon.leagueoflegends.com/cdn/14.7.1/data/en_US/summoner.json) pour récupérer les données des sorts d'invocateur et ainsi identifier les sorts contenues dans les détails du match récupéré.
- <https://static.developer.riotgames.com/docs/lol/queues.json> pour récupérer les données des files de jeu et identifier le type de file et la carte jouée du match récupéré.

Déconnexion
Quiz of Legends
Profil

### Question

Quel est le rang moyen de cette partie ?

**5v5 Ranked Flex games**  
Summoner's Rift  
41:47

**Equipe bleue**

Bjergkerking	5 / 3 / 5 3.333	216140	292	7.0/min	Objets
Grünkohl Hans	20 / 12 / 9 2.417	414356	45	11.1/min	Objets
Swagner X	4 / 8 / 10 1.750	231424	252	6.0/min	Objets
Dominator	15 / 3 / 8 7.667	287958	232	5.6/min	Objets
Jonearpitta	7 / 3 / 22 9.667	56679	42	1.0/min	Objets

**Equipe rouge**

DrOpYoUdOwN	5 / 6 / 3 1.333	286612	334	8.0/min	Objets
FalscherFuffi	19 / 11 / 2 1.909	269106	32	0.8/min	Objets
she dubbed me	4 / 11 / 7 1.000	321629	275	6.6/min	Objets
Asian Prince	0 / 11 / 4 0.364	254645	281	6.7/min	Objets
ViuviuLove	1 / 12 / 6 0.583	61280	56	1.3/min	Objets



- Composant ProfilPage

Le composant ProfilPage affiche le profil d'un utilisateur qui comprend sa photo de profil, son pseudo, son niveau, ses statistiques sur les réponses aux questions, et sa précision dans les réponses. L'utilisateur peut également modifier sa photo de profil en cliquant sur son image, ce qui ouvre une liste d'images disponibles parmi lesquelles il peut choisir. Une fois une nouvelle image sélectionnée, elle est mise à jour et affichée instantanément.

Appels AJAX effectués au serveur :

→ `/api/user/setPicture` pour mettre à jour la photo de profil de l'utilisateur.



### 4.3. Intégration CSS grâce à des sources tierces

Pour enrichir notre interface utilisateur, nous avons adapté des composants CSS provenant de BBBBootstrap [2] et CodePen [3] pour certains éléments comme les formulaires de connexion, d'inscription et la barre de recherche. Ces plateformes offrent du code CSS prêts à l'emploi, que nous avons ajusté pour correspondre à notre site.

Nous nous sommes également appuyés sur une solution trouvée sur Stack Overflow [4] pour créer une flèche stylisée pour le bouton d'accès au quiz afin de reproduire le style caractéristique des boutons du jeu League of Legends.

Nous avons également opté pour la police de caractères Friz Quadrata [5] qui est utilisée dans le client du jeu League of Legends afin d'ajouter une touche authentique à notre site et rappeler l'univers visuel du jeu. Nous l'avons téléchargée à partir de la source officielle disponible sur font.download.

## 5. Hébergement et Scalabilité

Pour l'hébergement de notre application "Quiz of Legends", nous utilisons une instance EC2 d'AWS [6]. Nous avons configuré cette instance avec un groupe de sécurité permettant le trafic HTTP et HTTPS, garantissant ainsi une communication sécurisée entre les utilisateurs et le serveur.

L'application est divisée en deux parties : le serveur, écrit en Go, et le client, développé avec React. Ces composants sont construits et exécutés de manière à fonctionner ensemble, le serveur Go s'occupant également de servir l'application React.

Pour garantir que notre application redémarre automatiquement en cas de panne ou après un redémarrage de l'instance, nous avons configuré un service systemd sur notre instance. Ce service systemd assure la gestion du serveur Go et s'occupe de lancer l'application automatiquement.

L'application est accessible via l'adresse IP publique de l'instance (<http://13.60.55.141:8080/>), permettant aux utilisateurs de se connecter et d'utiliser notre service en ligne sans interruption.

Pour scaler notre site et gérer une augmentation du trafic utilisateur, plusieurs approches peuvent être envisagées :

- En configurant un **Elastic Load Balancer**, nous pouvons distribuer le trafic entrant entre plusieurs instances. Cela permet de réduire la charge sur une seule instance et d'améliorer la performance et la disponibilité de l'application.
- L'**Auto scaling** permettrait d'ajuster automatiquement le nombre d'instances en fonction de la demande. En définissant des règles de scaling basées sur des métriques telles que l'utilisation du CPU ou la latence des requêtes, nous pouvons assurer que l'application maintienne une performance optimale, même en période de forte affluence.

En adoptant ces pratiques, "Quiz of Legends" peut répondre à une demande croissante, assurant ainsi une expérience utilisateur fluide et fiable.

## 6. Conclusion

En conclusion, le développement de notre site web "Quiz of Legends" a été une expérience enrichissante et amusante à effectuer. Notre intérêt pour le jeu League of Legends nous a motivés à créer ce quiz et a rendu le projet beaucoup plus intéressant pour nous. Nous avons eu l'occasion d'appliquer les notions que nous avons apprises en cours et en travaux dirigés, et voir le résultat final fonctionner correctement a été très satisfaisant.

Cette expérience nous a permis de mieux comprendre le processus de développement d'une application web et nous a donné un aperçu de ce à quoi pourrait ressembler le travail d'un développeur web dans le monde réel. Nous sommes fiers du résultat final et nous sommes reconnaissants pour les connaissances que ce projet nous a appris.

## 7. Bibliographie

1. Go Driver Quick Start - Go Driver v1.15.  
<https://www.mongodb.com/docs/drivers/go/current/quick-start/>.
2. Bootstrap 5 Glowing Login Form Example.  
<https://bbbootstrap.com/snippets/bootstrap-glowing-login-form-61831104>.
3. Ahmad Bassam Emran. (s.d.). Glowing Login Form. CodePen.  
<https://codepen.io/ahmadbassamemran/pen/rNjMXqg>
4. Kolisnyk, Yevhenii. « How to create arrow with border with CSS ». Stack Overflow, 21 janvier 2022, <https://stackoverflow.com/q/70797108>.
5. Font Download. Friz Quadrata Font Download, <https://font.download/font/friz-quadrata>.
6. « AWS | Amazon EC2 – Service d'hébergement cloud évolutif ». Amazon Web Services, Inc., <https://aws.amazon.com/fr/ec2/>.
7. Riot Developer Portal. <https://developer.riotgames.com/apis>.
8. React. <https://fr.react.dev/>.
9. « MongoDB: La Developer Data Platform ». MongoDB, <https://www.mongodb.com/fr-fr>.