STIIME

CATHRINE Paul
DELAISSE William
DELORT Tristan
PUERTAS Antoine

Projet Web 2020

Présentation

StIImE est un “réseau social” pour les jeux vidéos. Il permet de créer son profil, ajouter ses jeux et son pseudo, poster des messages sur les profils, les jeux, en public, noter les jeux, écrire des reviews. Les inspirations principales sont Steam, Twitter et MyAnimeList.

Il a ainsi pour but de comporter un annuaire des membres, jeux, serveurs (TrackMania, Minecraft par ex), noté par l’ensemble des utilisateurs (MyAnimeList/Steam) et de partager ses réactions, ses images, chercher différents partenaires de jeux sur le réseau social.

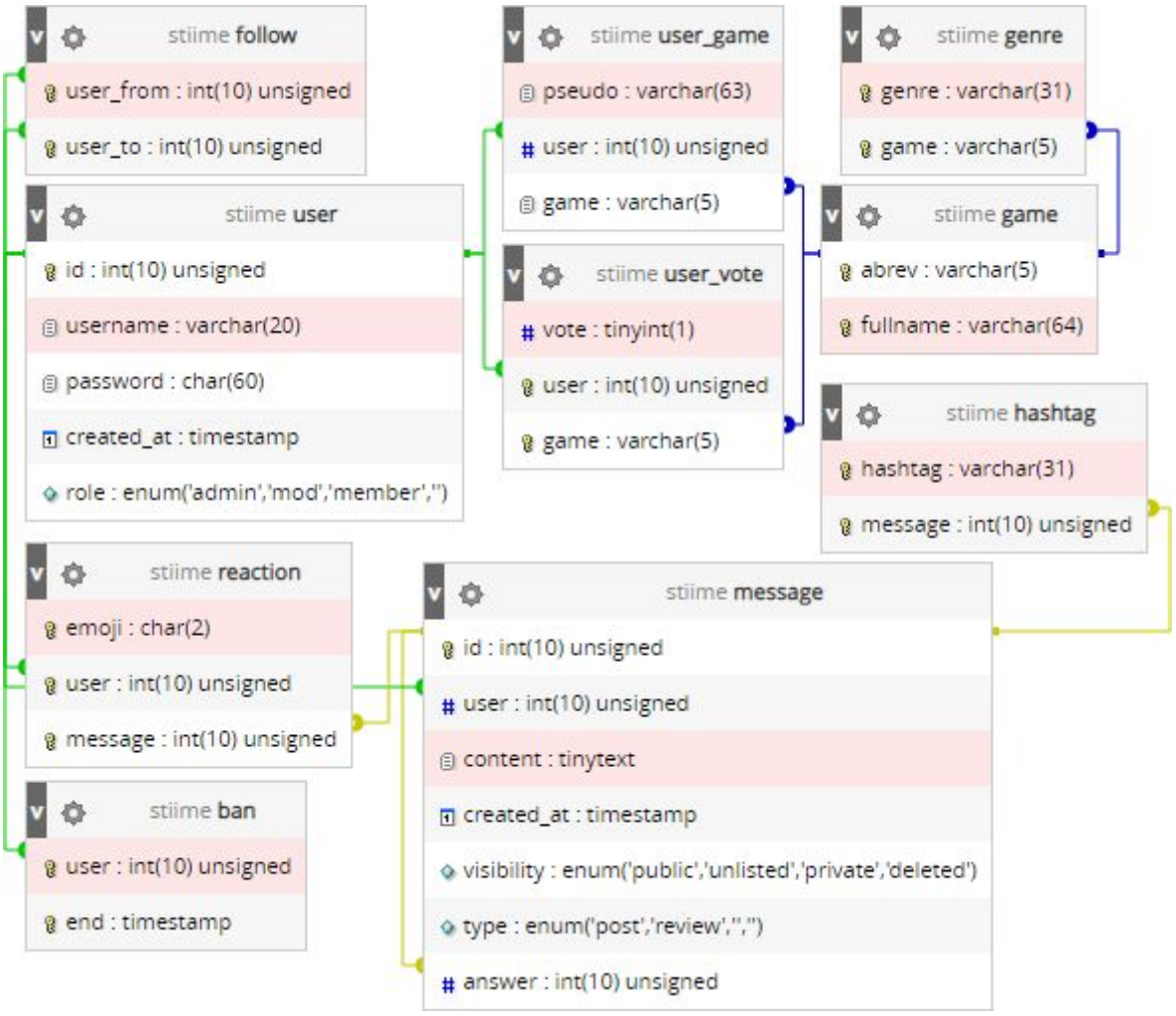
Les fonctions principales retenues dans le cahier des charges sont les suivantes :

- Base de données de jeux communautaire
- Base de données de stats perso de jeux (jeux joués/temps/etc) → Steam Web API
- Fil d’actualité pour chaque jeu
- Noter et donner son avis sur un jeu
- Annuaire avec filtres de recherches (jouant aux mêmes jeux, etc)
- Leaderboard

Réseau social, microblogging

- Blog personnel, fil d’actualité “Twitter”
- Réactions et reblog

Conception back-end



Cette conception permet de bien prendre en compte les fonctionnalités voulues. La base de données a été optimisée avec les contraintes d'unicité, d'indexation, clés étrangères pour améliorer les performances des requêtes.

Pour la sécurité des mots de passe, on a choisi le stockage des mots de passe en utilisant bcrypt : sel généré automatiquement, forte sécurité. Le champ `password` a donc le type CHAR(60)

```
password_hash ( string $password , int $algo [, array $options ] ) : string
$algo = PASSWORD_BCRYPT
password_verify ( string $password , string $hash ) : bool
```

Caution
Using the **`PASSWORD_BCRYPT`** as the algorithm, will result in the **`password`** parameter being truncated to a maximum length of 72 characters.

BCrypt generates an implementation-dependent 448-bit hash value. You might need CHAR(56), CHAR(60), CHAR(76), BINARY(56) or BINARY(60)

Conception front-end

Pour faciliter la conception front-end, on a utilisé Bootstrap et FontAwesome. Une partie du CSS est donc déléguée aux frameworks, ce qui permet d'éviter une partie du travail de mise en page (layout). Cependant, un style particulier avec un code couleur et des formes récurrentes est mis en place pour donner une identité au site.

L'architecture du site est un pseudo framework fait en php, avec un routeur, des contrôleurs, des composants, des services. Cette architecture sans framework est le produit de divers choix pour faciliter la conception du site, mais peut apporter des problèmes de sécurité et de maintenance par l'absence de règles codifiées, de documentation etc. L'idée générale de l'architecture est de s'en rapprocher. Ainsi les composants possèdent leur description HTML, leur style CSS, et éventuellement le code JS correspondant.

Arborescence (les documents, readme… ne sont pas apparents)

```
└──── stiime
    ├──── api          Les contrôleurs utilisés par l'API uniquement (fetch
    |                  JS, réponse JSON)
    ├──── controller   Les contrôleurs utilisés par le routeur
    ├──── css          La mise en forme globale du site
    ├──── data         Les données par défaut
    ├──── img          Les différents assets utilisés
    ├──── js           Les scripts globaux
    ├──── lib          Les librairies téléchargées localement
    ├──── Model        La connexion à la base de donnée
        ├──── Entity     Représentation des données en objet
        ├──── Hydrator       Fonctions pour transformer les données en objet
        └──── Repository     Les requêtes pour les différentes données
    ├──── utils        Les fonctions utilitaires (autoloader, parsing)
    ├──── view         Les vues et composants
    ├──── .htaccess    Paramètres pour réécriture des URL par le routeur
    ├──── config.php   Paramètres de connexion à la DB
    └──── index.php    Routeur
```

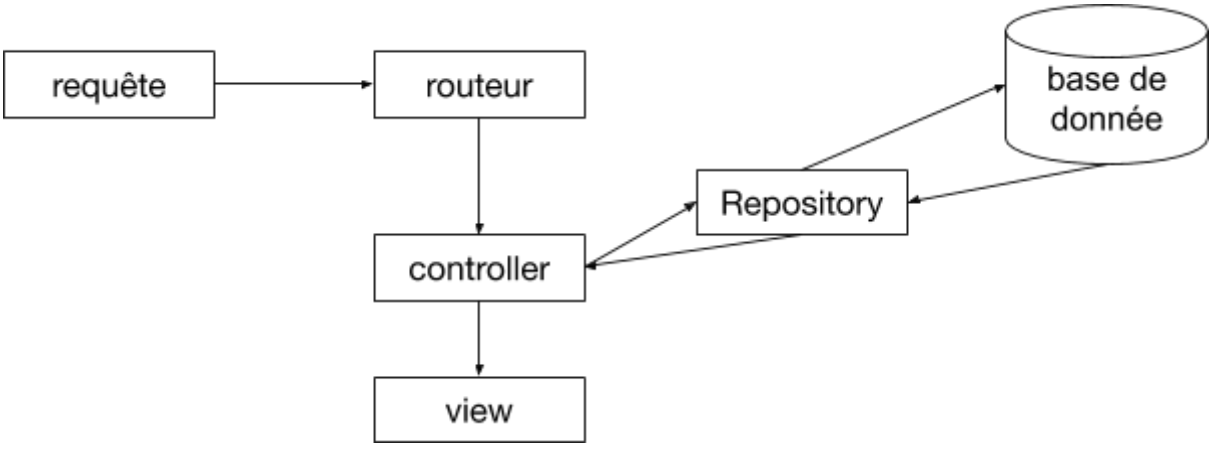
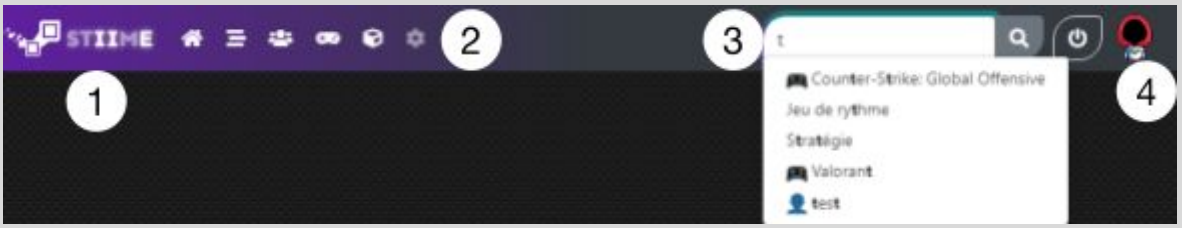


Schéma du chemin d'une requête dans l'architecture

Fonctionnalités

Navbar



1 Logo du site

2 Barre de navigation vers les différentes parties

La dernière icône ⚙ n'est visible que par les administrateurs et comporte le panneau d'administration (pas réalisé)

3 Barre de recherche utilisant Typeahead.js, une librairie qui gère dynamiquement les suggestions.

Une première version utilisait notre propre version via une `<datalist>` qui à chaque input de l'utilisateur mettait les options récupérées via la route `/api/search?q=%query%`, mais pour des raisons esthétiques, elle a été remplacée par une librairie.

Après l'entrée ou non de caractères, elle affiche donc dynamiquement en JS, les utilisateurs, jeux, hashtags, genres qui contiennent ces lettres.

4 Bouton de déconnexion et profil de l'utilisateur

Register

Enregistre un compte avec l'username et le mot de passe envoyé en POST

Mot de passe hashé dans le back-end avec bcrypt (fonction php `password_hash`), le sel est donc généré automatiquement, on ne s'en occupe pas.

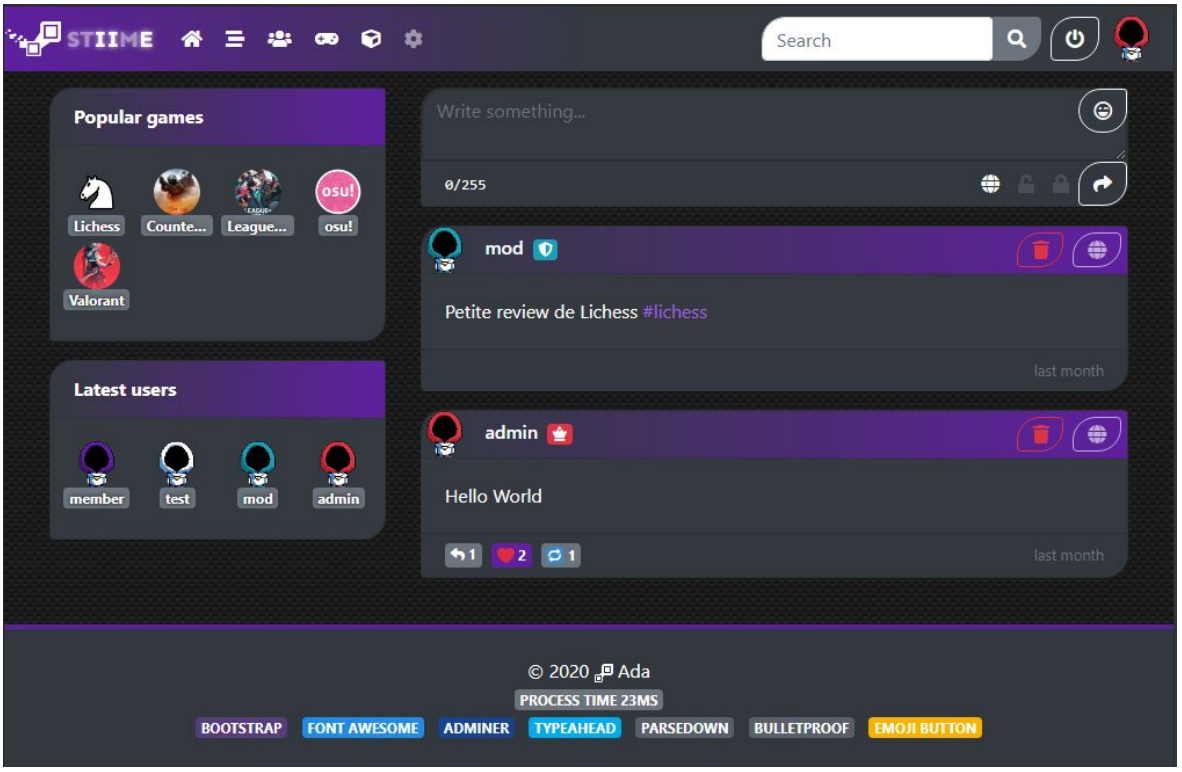
ID est le numéro de l'utilisateur, puisqu'il peut changer de pseudo, qui s'incrémente à chaque nouvel utilisateur. La suppression d'un utilisateur laisse un "trou".

Login

Connection avec le username et mot de passe, vérification en back-end avec `password_verify`

Le contrôleur créer une session et stocke les informations de l'utilisateur (id, username et role), puis route vers la page principale.

Page principale et timeline



Affichage des derniers messages avec une visibilité publique, scroller vers le bas affiche dynamiquement 20 autres messages plus anciens (JS).

Sur la gauche, on retrouve les jeux populaires : la requête trie les jeux qui ont le plus de messages référençant le jeu avec le hashtag, ainsi #lichess mentionne le jeu et le place en première position.

Les derniers utilisateurs, par date d'inscription sont affichés en dessous.

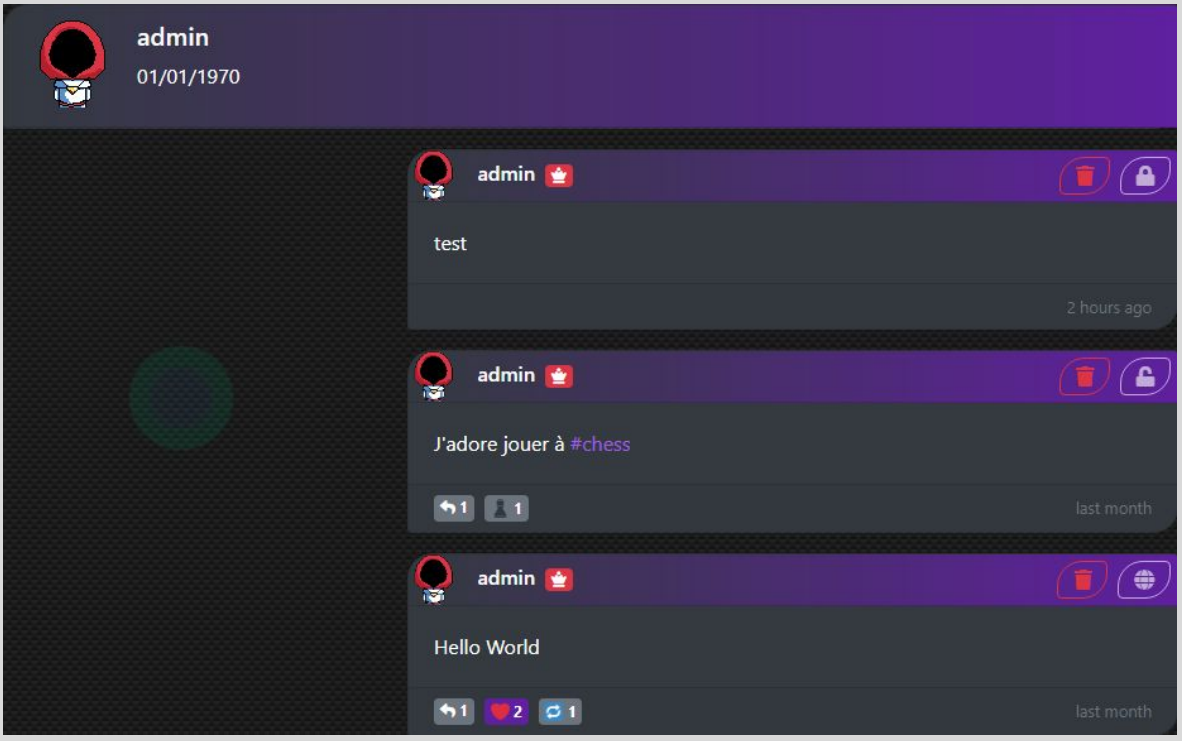
On peut entrer et poster un nouveau message, en choisissant la visibilité, un panel pour choisir les emojis est aussi intégré. La limite de caractères est 255 (type `TINYTEXT`).

Seuls les admins peuvent supprimés les messages. Pour l'administration, la suppression d'un message change sa visibilité en "deleted", ils ne sont pas purgés de la base de donnée, uniquement avec le panel d'administration (pas réalisé).

Les réactions ne sont pas gérées, elles sont néanmoins visibles avec les réponses.

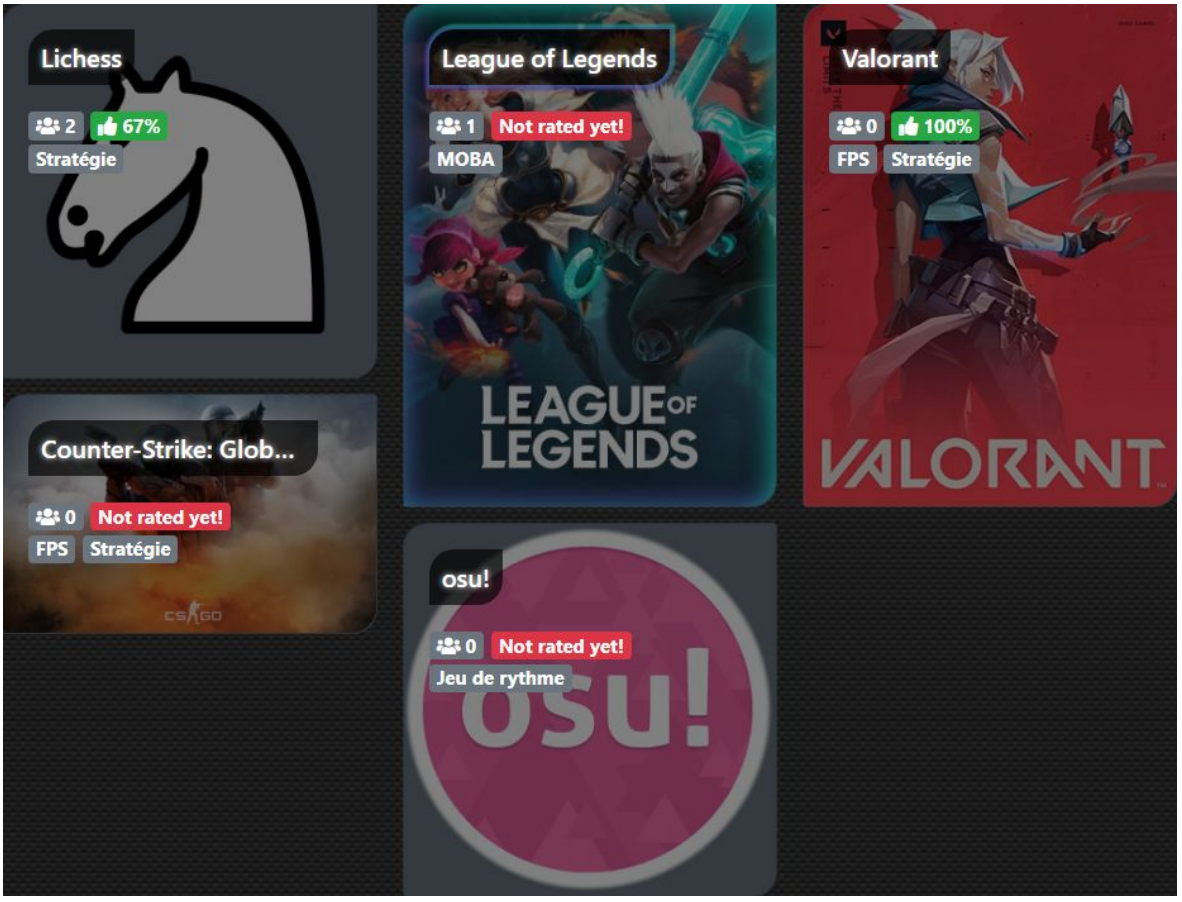
De manière similaire, l'onglet "timeline" affiche les jeux que vous possèdez (dont vous avez renseigné un pseudo), les gens qui vous suivez et leurs messages sur la droite (publics, "unlisted" et privés si vous vous suivez mutuellement).

Page de profil `GET {ID}`



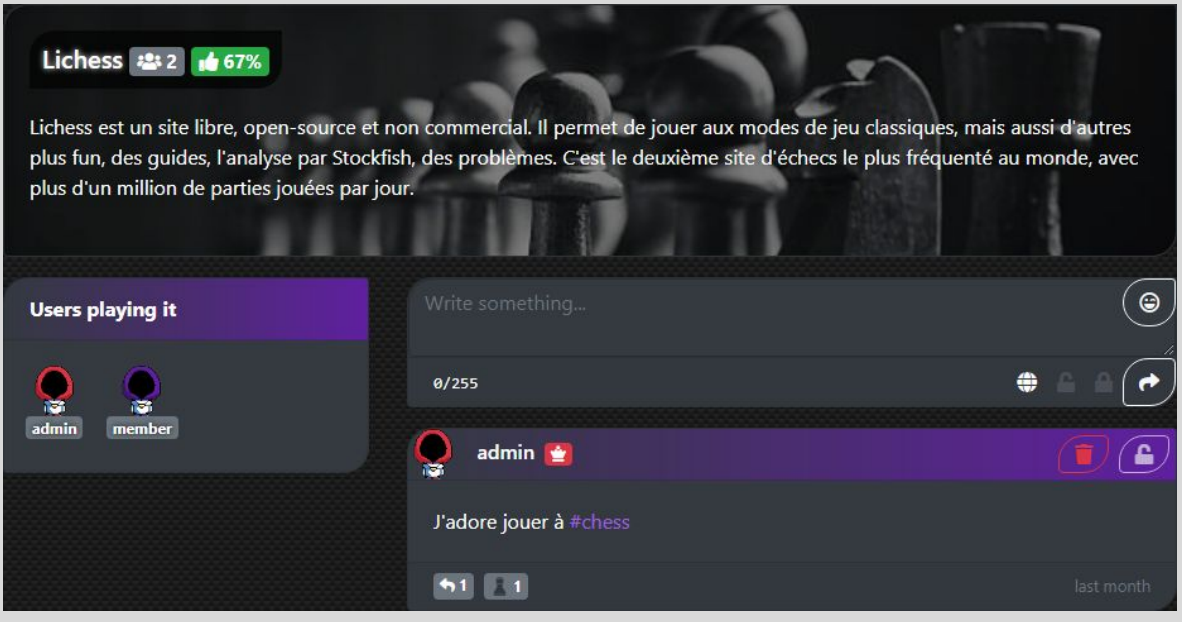
Affichage de son pseudo, sa date de création, ses messages.
Pas réalisé : les jeux et pseudos, les gens qui suivent le compte, les gens que le compte suit.

Annuaire des jeux



Les jeux sont affichés avec leur nom, leur image, le nombre de joueurs, leur note et leurs genres.

Page de jeu game `GET {ID}`



On affiche les informations de la page de l'annuaire ainsi qu'une description, une image de fond. On peut visualiser les joueurs qui ont un pseudo dans le jeu (affiché avec un tooltip au passage de la souris), ainsi que les messages mentionnant le jeu.

Les tâches non réalisées mais prévues

Page de recherche

+ Par termes dans un statut
+ Par hashtag (par jeu, recherche de groupe, ou non "spéciaux)
+ D'utilisateurs cf. **Navbar** autocomplétion
+ De jeux cf. **Navbar** autocomplétion

Panel administration

+ Affichage des utilisateurs et sanctions
+ Suppression utilisateur
+ Affichage des messages supprimés (ils ne sont pas supprimés définitivement mais leur visibility est "deleted") et suppression **définitive** de la DB
+ Création d'un jeu
+ Ajout d'un genre
+ Modification de l'username d'un utilisateur (si inapproprié)

Page de communauté

Tests

- Vérification du respect des normes HTML avec **w3c** (ne pas oublier l'attribut alt des images)
- Injection SQL
- XSS

Pour aller plus loin

API RESTful JSON en PHP et à documenter totalement

Système de notifications (push api notification js) → Bootstrap Toast

ActivityPub

Conclusion

Le projet n'a pas pu être réalisé à terme à cause du manque de temps et la difficulté d'implémentation des fonctionnalités de base, notamment celles des messages : visibilité, parsing des hashtags, markdown, les différents type, l'ajout, la suppression, les réactions, la gestion des emojis. Cela dit la conception du projet, de son architecture, des choix techniques a été enrichissante.