UCLouvain

Université Catholique de Louvain

LINFO1212 : Projet d'approfondissement en sciences informatiques

Groupe F

Julien Devos
Béranger Deketelaere
Honoré Nizeyimana Niwentwari

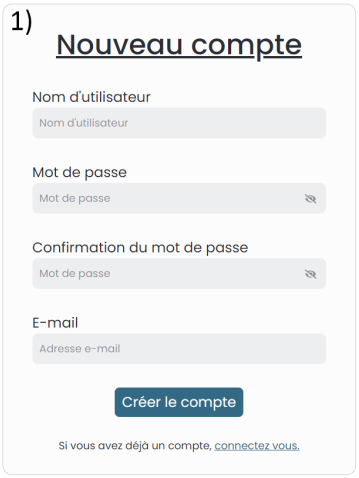
December 20, 2021

Contents

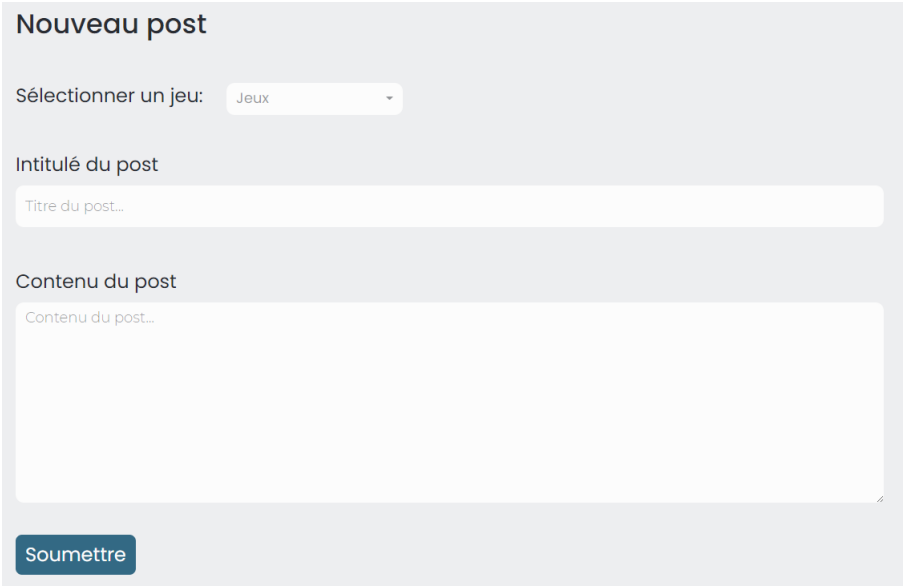
1 Les besoins de l'utilisateur et mise en place de ceux-ci

L'utilisateur doit être capable de:

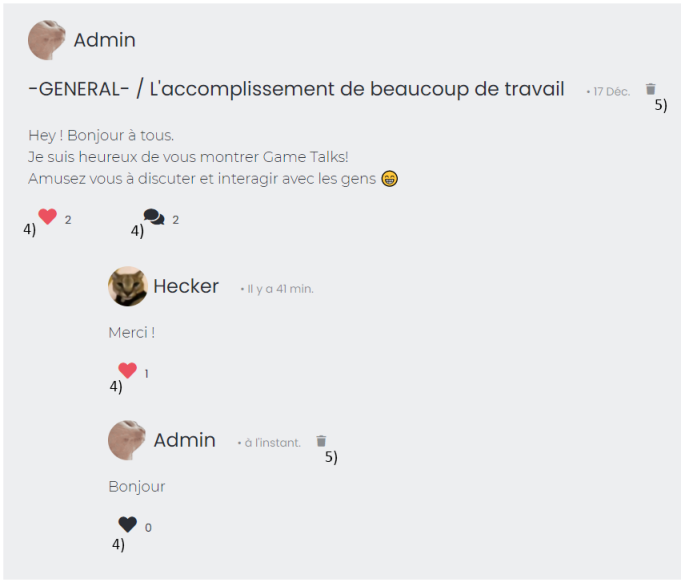
1. Se créer un compte.
2. Se connecter à son compte à l'aide de son nom d'utilisateur et mot de passe.

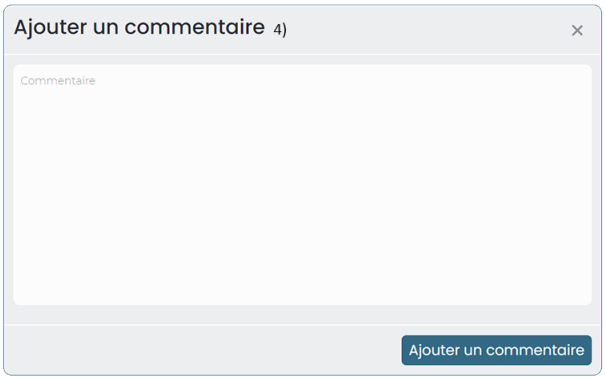


3. Créer un post sur un jeu au choix.

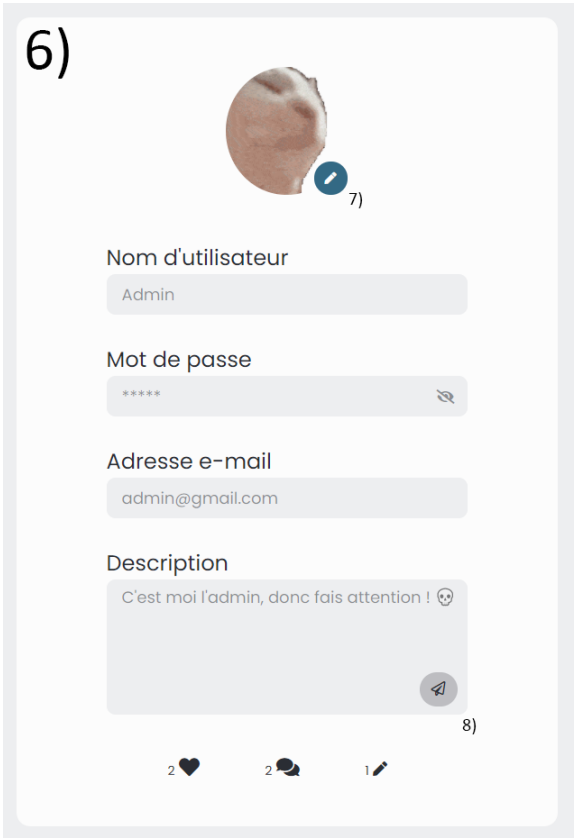


4. Commenter et liker des posts ou des commentaires.
5. Supprimer ses posts et commentaires.

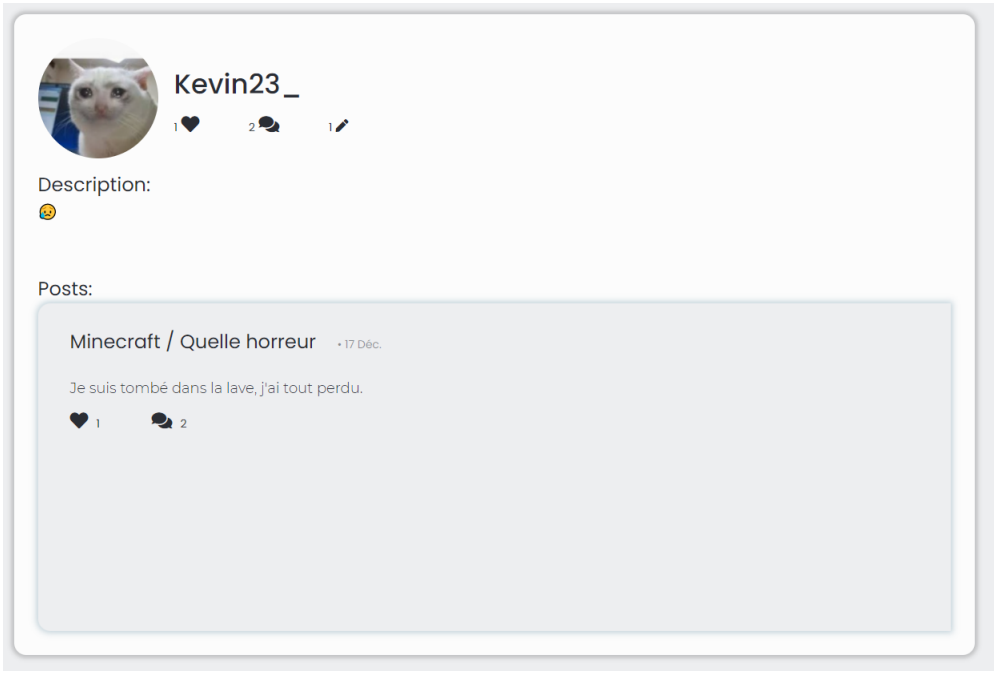




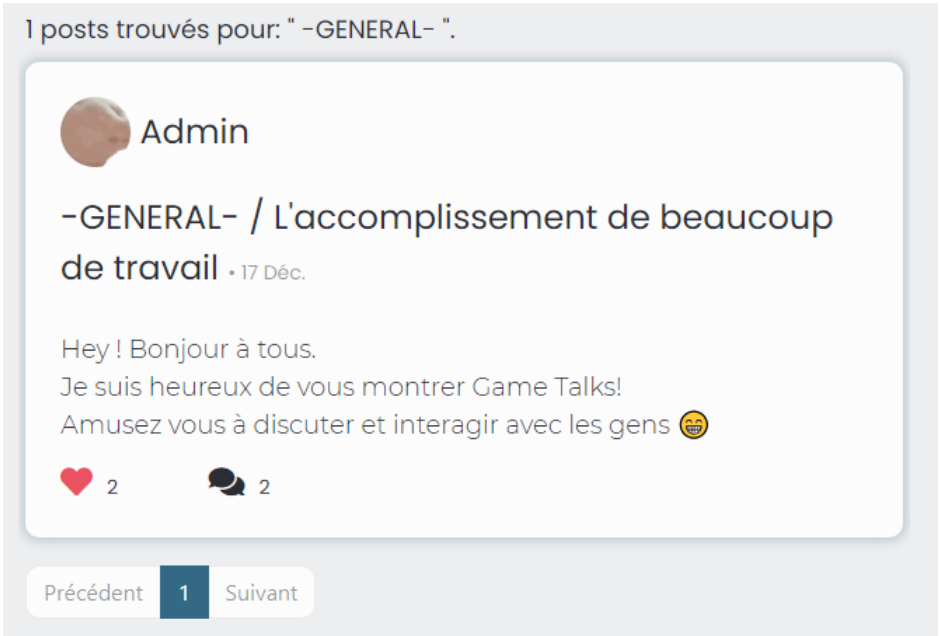
6. Voir son profil avec ses données d'utilisateur.
7. Modifier sa photo de profile.
8. Modifier sa description.



9. Voir le profil d'un autre utilisateur.



10. Rechercher parmi les posts et les commentaires.



2 Prise en compte des commentaires de la spécification

2.1 Commentaires sur la spécification

Il nous avait été conseillé de rajouter:

- un croquis d'un poste type qui manquait à notre rapport initial.
- l'option d'accéder à un profil d'utilisateur.
- la possibilité de rechercher un post en fonction de ses commentaires.

2.2 Changements effectués

Nous avons bien pris en compte les commentaires et nous avons implémenté les fonctionnalités ci-dessus.
Nous avons également rajouté un exemple de post type au rapport final.

3 Architecture du système

3.1 Description des collections de la base de donnée

- Games
  - id, name
- Users
  - id, Username, password, mail, desc
- Comments
  - id, post_id, author_id, content, date, likes, like_id
- Posts
  - id, game_id, author_id, title, content, date, likes, comments, like_id, subject

L'id de chaque élément dans chaque collection, est un identifiant généré par mongoDB.
Users: desc, la description qui s'affiche sur le profil utilisateur.
Posts: likes,comments, nombre de likes et commentaires du post.
Posts: like_id, Array qui contient tous les identifiants des utilisateurs qui on liké le post en question.
Posts: subject, un String qui comprend le contenu du post qui pourra être recherché.
Comments: likes, le nombre de likes du commentaire.
Comments: like_id, Array qui contient tous les identifiants des utilisateurs qui on liké le commentaire en question.

3.2 Le module Mongoose

Nous avons décidé d'utiliser le module Mongoose pour faciliter l'insertion et la recherche dans la base de donnée. Le module permet de créer des schémas de chaque collection de la base de donnée,

```
const mongoose = require('mongoose');


const GameSchema = mongoose.Schema({
    name: {
        type: String,
        required: true
    }
})

module.exports = mongoose.model('Games', GameSchema);
```

Figure 1: Schéma de la collection Games.

ce qui permet de créer un nouvel objet de la collection au moment de rajouter un nouvel élément.

```
const post = new Post({
    game_id: req.body.game_id,
    author_id: req.session.user_id,
    title: req.body.title,
    content: req.body.content,
    subject: subject
});

const savedPost = await post.save();
```

Figure 2: Exemple d'ajout d'un post dans la base de donnée.

Le module permet également de définir une valeur par défaut pour un attribut de la collection, par exemple pour définir la date.

```
date: {
    type: Date,
    default: () => Date.now()
},
```

Figure 3: Valeur par défaut

Il est également plus simple de se connecter à la base de donnée. Il n'est pas nécessaire d'écrire les appels au même endroit que la connexion à celle-ci.

```
// Connect mongoose to db
mongoose.connect('mongodb://localhost:27017/GameTalks', () => {
    console.log('Connected to db')
});
```

Figure 4: Connexion à la base de donnée.

```
let post = await Post.find({"_id":req.query.id});
```

Figure 5: Appel à la base de donnée.

3.3 Fonctionnalité de recherche d'informations

L'utilisateur a la possibilité de rechercher des posts en fonction des mots clés qu'il rentre dans la barre de recherche. Les posts sont trouvés si un ou plusieurs mot clés se trouvent dans l'attribut subject de ceux-ci. Un post peut également être trouvé s'il ne contient pas le mot clé mais qu'il possède un commentaire qui le contient. Pour permettre d'être plus précis sur la recherche, l'utilisateur a la possibilité de sélectionner un ou plusieurs filtre qui permet de filtrer les résultats en fonction du ou des jeux choisis.

Pour améliorer la reconnaissance des mots clés, nous avons créer une fonction qui permet de "lemmatiser" un String. Autrement dit, lui donner une forme "racine", une manière d'être identifiée peut importe s'il est au pluriel, au singulier, en fin de phrase, avant une virgule ou encore s'il commence par "s', qu', l', c'". Ce qui permet de transformer chaque mot recherché dans sa forme racine, les comparer avec le "subject" des posts qui lui aussi est lemmatisé.

Pour permettre un classement par pertinence des posts trouvés lors de la recherche. L'indice TF-IDF est calculé pour chaque post. Cet indice permet de mettre en premier les posts pour les quels les mots recherchés ont le plus d'importance.

Pour conclure, la recherche implémentée est performante et arrive à trouver un post dans la plupart des cas.

3.4 L'utilisation des routes Express

Pour permettre de rendre le code plus lisible et de ne pas mettre l'entièreté des routes dans le fichier "app.js". Nous avons utilisé les Router express pour permettre de mettre chaque itinéraire dans son fichier respectif.

```
// Import all routes
const homeRoute = require('./routes/home');
const postRoute = require('./routes/post');
const postsRoute = require('./routes/posts');
const userRoute = require("./routes/user");
const gameRoute = require("./routes/game");
const loginRoute = require("./routes/login");
const registerRoute = require("./routes/register");

app.use('/', homeRoute);
app.use('/post', postRoute);
app.use('/posts', postsRoute);
app.use('/user', userRoute);
app.use('/game', gameRoute);
app.use('/login', loginRoute);
app.use('/register', registerRoute);
app.get('*', async (req, res) => {
    res.render("404.html")
});
```

Figure 6: Importation des routes du site.

Donc, si l'utilisateur veux accéder à l'url suivant:

- https://ip/, ça ira dans le fichier ./routes/home
- https://ip/post, ça ira dans le fichier ./routes/post
- https://ip/post/add, ça ira dans le fichier ./routes/post pour l'itinéraire /add

3.5 Création d'un module

Nous avons créé un module "utils" qui contient l'entièreté des fonctions créées pour le site. Ce qui permet de l'importer dans n'importe quel fichier du projet et d'utiliser les fonctions qu'il contient.

```
// complete the posts with date, username and game
await utils.completePost(post, req.session.user_id, function (result){
    post = result;
});
```

Figure 7: Exemple d'utilisation d'une fonction du module utils

3.6 Organisation des fichiers statiques

Nous avons organisé le dossier "static" en plusieurs sous dossiers. Un dossier "fonts" qui contient les deux polices utilisées sur le site ainsi que la police d'icônes crée par nos soins. Ensuite, un autre dossier "img" qui contient toutes les images utilisées. Dans ce dossier image, un dossier "logo", qui contient les logos du site et enfin, un dossier "users-pfp" qui contient les photo de profil des utilisateurs.

3.6.1 Séparation du css

Vu que nous utilisons un certain nombre de typographie différente, nous avons créé un fichier css qui ne contient que les différentes règles pour les polices (font-style.css)

Il y a donc différentes classes comme: titre1, titre2, headline1, headline2, etc. Toutes ces classes correspondent à un certain style d'écriture (police, taille de police, poids de la police, etc) Ce qui nous permet de ne pas répéter ces informations et de simplement utiliser cette classe sur les texte en fonction du style qu'ils doivent avoir.

Ce fichier contient également les différentes classes utilisées par les icônes. Ex (icon-Quit, icon-profile, etc) Ce qui permet de simplement utiliser une classe quand on veut mettre un icône.

4 Approches utilisés pour assurer la qualité du projet

4.1 Ajouts supplémentaires

Nous avons tenu à rajouter divers fonctionnalités supplémentaire qui n'était cité dans la spécification comme un filtre de recherche, un mode sombre, un profile public d'utilisateur et plus encore.

Nous avons aussi tenu à ajouter un aspect esthétique important à notre site en prenant soin à réaliser un logo et à sélectionner des couleurs avec attention pour avoir un look moderne.

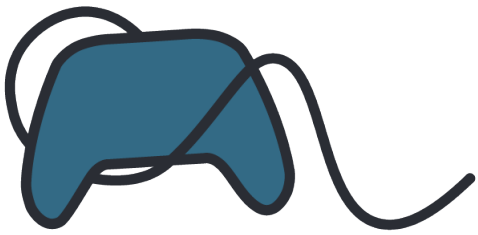


Figure 8: Logo du site

4.2 La réalisation de tests

Le site a été testé en créant des comptes, ajoutant des posts, des commentaires, des likes, enlevant les likes. En bref, nous avons testé chaque aspect du site.

4.3 L'utilisation d'un outil de prototypage: Figma

Nous voulions avoir un site d'apparence moderne. Pour se faire nous avons utilisé Figma qui est un outil en ligne de design d'application. Nous avons donc défini chaque page que devait comporter le site, ainsi que chaque couleur et chaque typographie utilisé sur celui-ci.

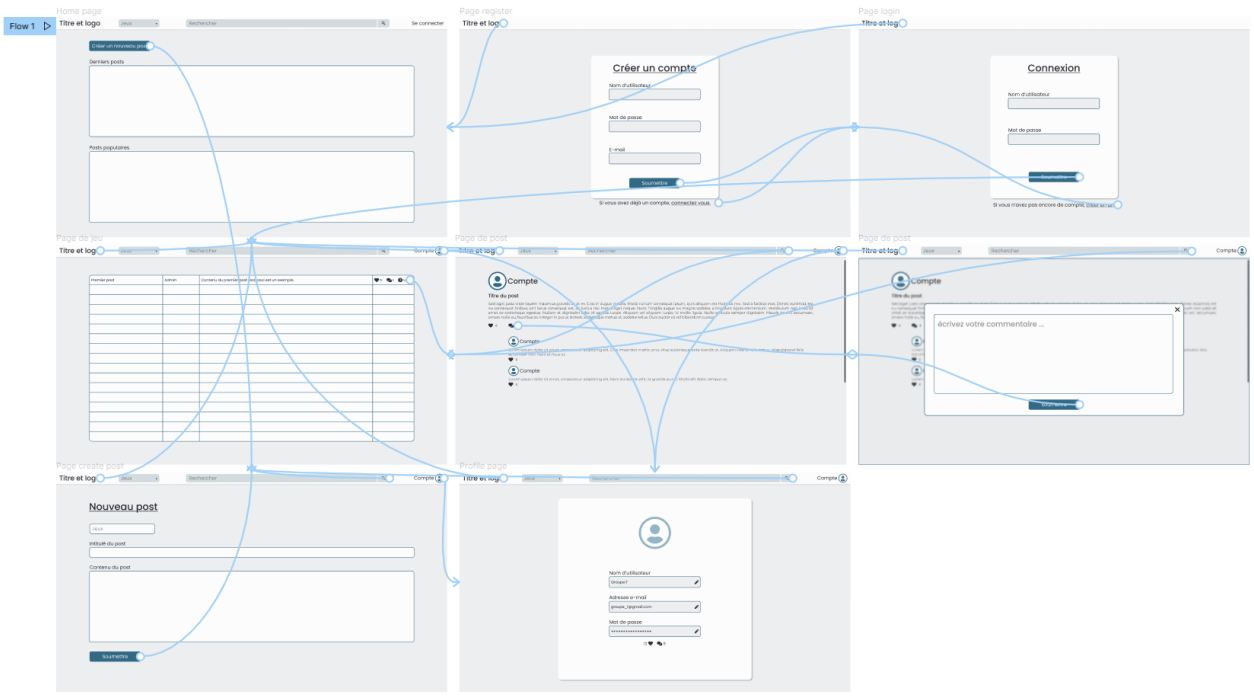


Figure 9: Prototype du mode clair sur le site

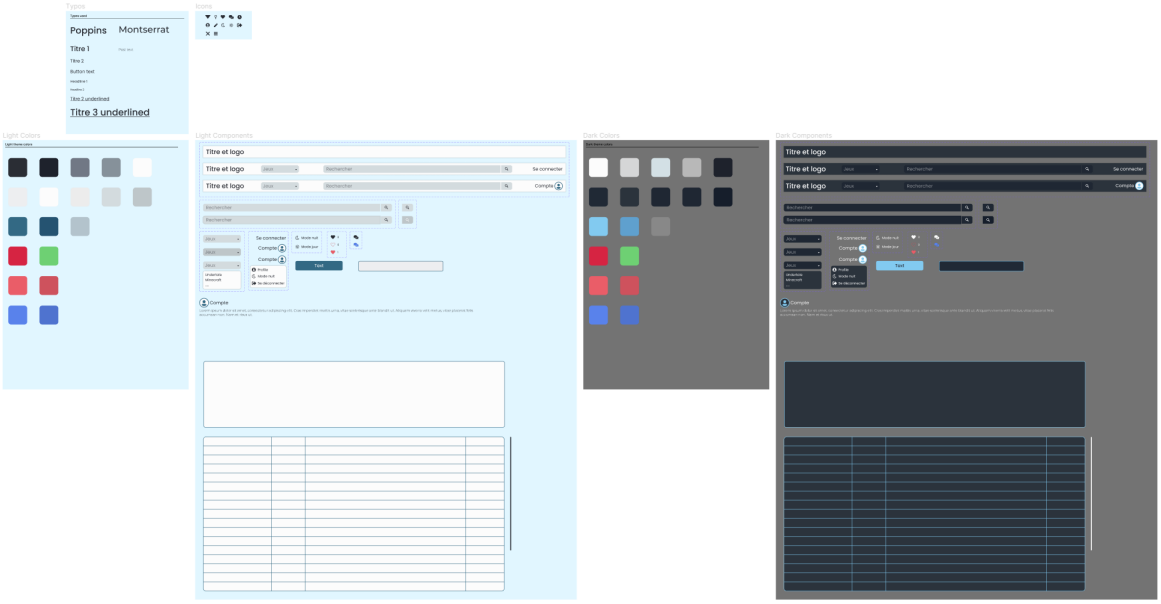


Figure 10: Composants du site (couleurs, typographie, icônes)

5 Répartition du travail

Nous avions réparti les pages à faire entre chaque membre du groupe. Et nous avons également organisé des réunions pour expliquer ce qui a été fait. Des réunions pour faire certaines parties importante ensemble on été faite comme par exemple, le prototype, la fonctionnalité de recherche et finalement le rapport.

Nous avons également testé le site chacun de notre côté, pour l'installation et le fonctionnement.

6 Conclusion

Nous avons répondu aux spécifications du projet en implémentant un site élégant comprenant une base de donnée regroupant l'ensemble des données du site, un algorithme de recherche efficace, la prise en charge de comptes utilisateurs ainsi qu'en ajoutant des fonctionnalités supplémentaires tels qu'un filtre de recherche, un profile public pour chaque utilisateur, la possibilité de supprimer des éléments dans la base donnée, etc.

Pour finir, nous pensons avoir correctement répondu aux problématiques initiales en offrant une solution performante.

Merci d'avoir tout lu ♡