Rapport de projet service web

- Intitulé : Système de feedback sur des produits numériques
- Élaboré par : Ghassen abdelaziz - Chaima Limayem - Chaima Gadhgadhi
- Classe 4ème GLSI-A

Analyse du problème

1. Contexte

Une entreprise de développement SaaS met à disposition plusieurs produits numériques à ses utilisateurs. Cependant, elle ne dispose actuellement **d'aucun système structuré** pour collecter, centraliser et analyser les **retours utilisateurs** concernant ces produits. Les feedbacks, bien que cruciaux pour l'amélioration continue des services, sont donc **sous-exploités**.

2. Entités identifiées & relations

2-1 En tités :

🔹 Users

Représente un utilisateur authentifié de la plateforme

Champ	Type	Description
`id`	UUID / int	Identifiant unique de l'utilisateur
`username`	string	Nom d'utilisateur
`password`	string	Mot de passe (hashé)

🔹 Products

Représente les produits vendus par la plateforme.

Champ	Type	Description
`id`	UUID / int	Identifiant unique du produit
`name`	string	Nom du produit
`category`	string (optionnel)	Type logiciels

🔹 Feedbacks

Représente un retour utilisateur sur un produit.

Champ	Type	Description
`id`	UUID / int	Identifiant unique
`user_id`	UUID / int	Référence vers l'utilisateur
`product_id`	UUID / int	Référence vers le produit
`rating`	int	Note de 1 à 5
`comment`	string	Commentaire de l'utilisateur
`date`	datetime	Date du feedback

2-2 Relations :

Entité	Relation	Multiplicité
User	donne plusieurs feedbacks	1 → *
Product	reçoit plusieurs feedbacks	1 → *
Feedback	appartient à un User et un Product	* → 1

3- Liste des fonctionnalités :

Authentification Admin :

Fonctionnalité	Type	Endpoint/Query	Description
Connexion admin	`POST` GraphQL	`login(username, password): token`	Authentifie l'admin et retourne un JWT.
Accès sécurisé aux routes	`Middleware`	`Authorization: Bearer <token>`	Protéger les pages `/admin` et accès à `/graphql` pour les opérations admin.

Feedbacks :

Fonctionnalité	Type	Endpoint/Query	Description
Ajouter un feedback	`Mutation` GraphQL	`addFeedback(user, product, rating, comment)`	Envoie un nouveau feedback.
Lister tous les feedbacks	`Query` GraphQL	`allFeedbacks { id, user, product, rating, comment, date }`	Récupère tous les feedbacks enregistrés.
Supprimer un feedback	`Mutation` GraphQL	`deleteFeedback(id: ID!): Boolean`	Supprime un feedback par ID.

Statistiques Produits :

Fonctionnalité	**Type**	**Côté Client**	**Description**
Moyenne des notes par produit	Calcul JS	Local (client-side)	Calcule la moyenne des notes de chaque produit.
Classement des meilleurs produits	Calcul JS	Local (client-side)	Trie les produits par note moyenne pour l'affichage.

Recherche et filtres

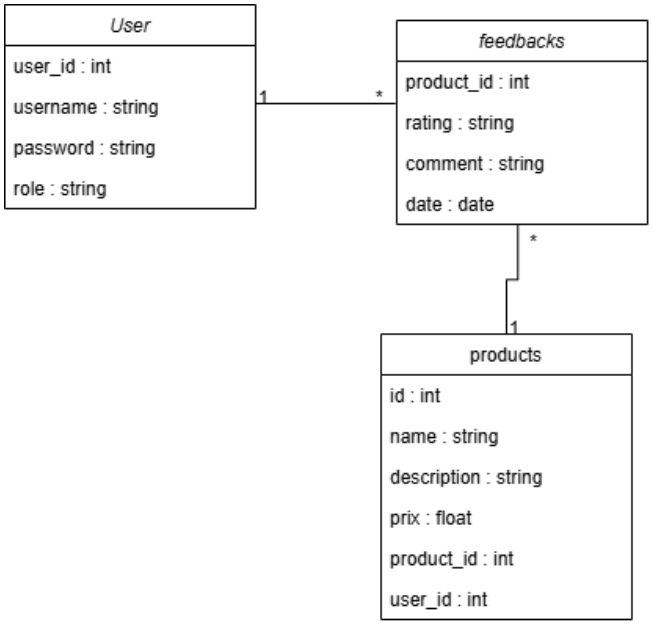
Fonctionnalité	Type	Côté Client	Description
Filtrer par nom d’utilisateur	JS DOM	Barre de recherche	Filtre les résultats selon l’utilisateur.
Filtrer par nom de produit	JS DOM	Barre de recherche	Filtre les résultats selon le produit.

🌐 **Pages Web Services :**

Page	Route HTTP	Description
Formulaire feedback	`GET /`	Page principale pour envoyer un avis.
Page de connexion admin	`GET /login`	Page de login pour les administrateurs.
Tableau de bord admin	`GET /admin`	Page de gestion et visualisation des feedbacks.

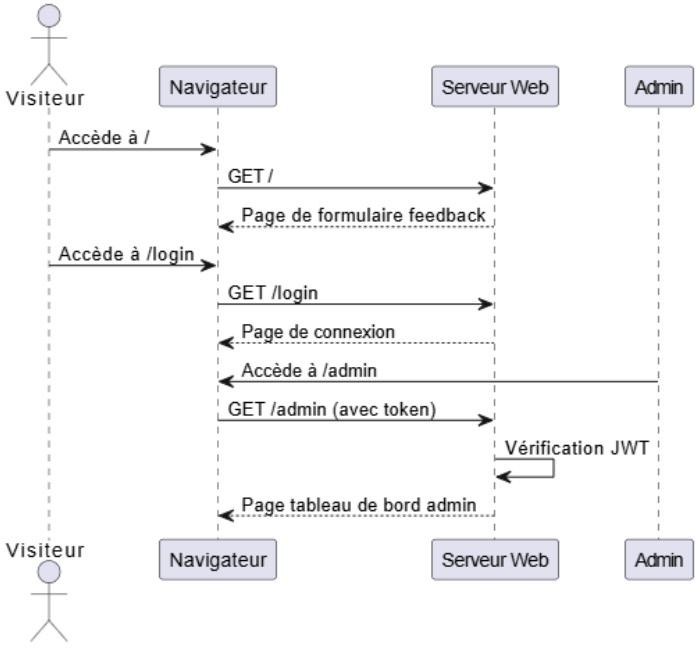
3. Diagrammes :

Diagramme de classes :

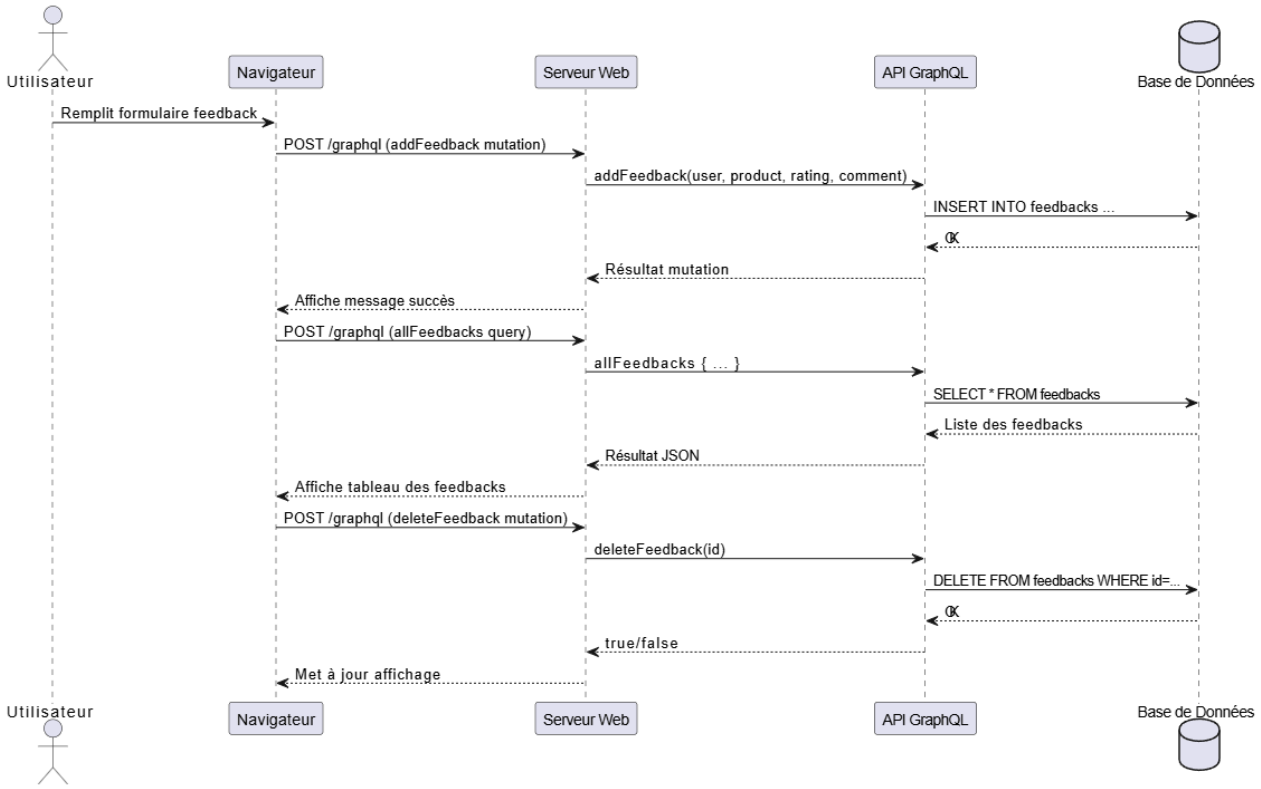


4. Schema des requetes & reponses du service

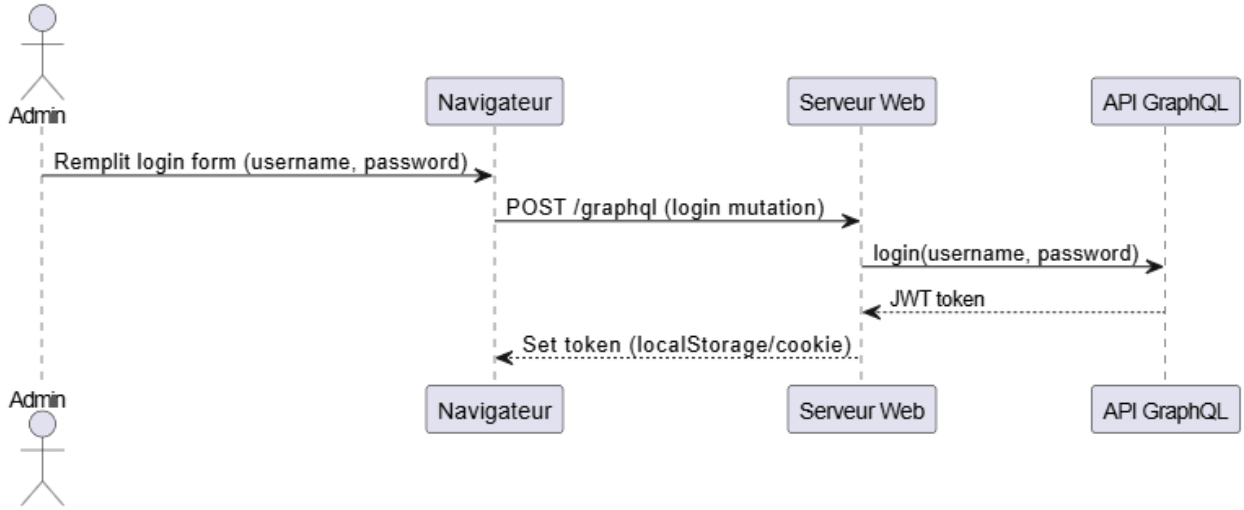
Navigation :



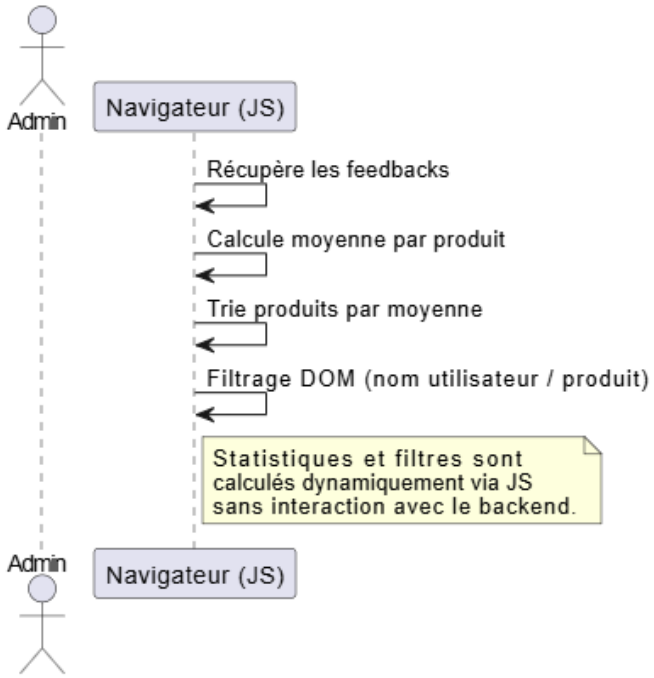
Feedbacks (ajout, listing, suppression) :



Authentification :



Statistiques :



4. Documentation:

4.1 Description des endpoints GraphQL:

Authentification Admin:

Mutation : login(username: String!, password: String!): String
Permet d'authentifier un administrateur et de récupérer un jeton JWT nécessaire pour les opérations protégées.

Feedbacks:

Mutation : `addFeedback(user: ID!, product: ID!, rating: Int!, comment: String!): Feedback`
-Permet à un utilisateur d'ajouter un retour sur un produit.
Query : `allFeedbacks: [Feedback]`
-Permet de récupérer l'ensemble des retours enregistrés dans le système.
Mutation : `deleteFeedback(id: ID!): Boolean`
Permet de supprimer un feedback spécifique en se basant sur son identifiant.

4.2 Structure des types GraphQL:

```
type User {
  id: ID!
  username: String!
}

type Product {
  id: ID!
  name: String!
  description: String
  prix: Float!
}

type Feedback {
  id: ID!
  user: User!
  product: Product!
  rating: Int!
  comment: String!
  date: String!
}
```

4.3 Exemples d’utilisation:

Ajouter un feedback:

```
// Mutation : ajouter un feedback
  addFeedback: async ({ user, product, rating, comment }) => {
    console.log("addFeedback appelé avec :", {
      user,
      product,
      rating,
      comment,
    });
```

Voir tous les feedbacks:

```
query {
  allFeedbacks {
    id
    user {
      username
    }
    product {
      name
    }
    rating
    comment
    date
  }
}
```

Supprimer un feedback:

```
// supprimer un feedback
  deleteFeedback: async ({ id }) => {
    try {
      const res = await Feedback.deleteOne({ _id: id });
      console.log("🗑️ Résultat suppression :", res);
      return res.deletedCount === 1;
    } catch (err) {
      console.error("❌ Erreur deleteFeedback :", err);
      return false;
    }
  },
};
```