Rapport du projet de programmation Web

WADRA Pierre

RUEFF Nathan

3A Cyberlog

Rappel des objectifs :

Le projet consiste en la création d'un site web dédié à la gestion des tâches en ligne, permettant aux utilisateurs de stocker et de gérer leurs tâches sur un serveur distant. Le site offre la possibilité de s'authentifier pour gérer ses propres tâches ou de s'inscrire pour les nouveaux utilisateurs. Le système est développé avec une architecture divisée en deux principales composantes :

- **Back-end (serveur)** : Implémenté avec Node.js et le mini-framework ExpressJS, le back-end utilise une base de données (au choix entre relationnelle comme MySQL ou NoSQL comme MongoDB) pour gérer les données. L'API REST développée permet la gestion des utilisateurs et des tâches. Les fonctionnalités utilisateur incluent l'inscription, la mise à jour des informations personnelles, et la désinscription. Une gestion spécifique est prévue pour un administrateur qui peut gérer tous les utilisateurs. Pour les tâches, les utilisateurs peuvent ajouter, marquer comme complétées, ou supprimer leurs tâches après authentification. La méthode d'authentification utilisée peut être basée sur des sessions ou sur des tokens JWT.
- **Vue (client)** : Le front-end produit du HTML pour créer une interface utilisateur web conviviale. Les pages peuvent être écrites en HTML pur ou générées côté serveur via des moteurs de template tels que EJS, facilitant l'interaction utilisateur avec l'application.

Degré d’aboutissement

Le projet a été réalisé conformément aux spécifications initiales avec la mise en place de l'ensemble des fonctionnalités requises, à l'exception de la partie permettant à l'administrateur de modifier les informations des utilisateurs. Cette fonctionnalité ne fonctionne pas correctement.

Structure Générale du Projet

Frontend (Dossier `public`)

- **Contenu** : Ce dossier contient les fichiers statiques qui sont directement servis au client. Il inclut:
  - **HTML** : Les fichiers HTML qui forment la structure de notre site web.
  - **CSS** : Les feuilles de style pour le design et la mise en page de notre site.
  - **JavaScript** : Scripts qui gèrent la logique côté client, interactions avec l'API pour envoyer et recevoir des données.
- **Fonction** : Le frontend est la partie visible par l'utilisateur. Les fichiers ici gèrent l'affichage, l'interactivité, et la communication avec le backend via les appels API.

Backend (Dossier `src`)

1. **Dossier** `api` :
   - **Contenu** : Contient les configurations initiales du serveur, telles que la mise en place du serveur Express, la configuration des middlewares globaux comme CORS et JSON parser, et le démarrage du serveur.
   - **Fonction** : Sert de point d'entrée pour l'application backend. Initialise les configurations nécessaires pour que l'application serve les requêtes.

2. **Dossier** `config` :
    - **Contenu** : Contient des fichiers de configuration, typiquement pour des connexions à la base de données.
    - **Fonction** : Centralise les configurations liées à la base de données, facilitant la gestion et la réutilisation des connexions dans toute l'application.
3. **Dossier** `routes` :
    - `users.js` : Gère les routes liées aux opérations sur les utilisateurs comme l'inscription, la connexion, la mise à jour des profils, et la suppression d'utilisateurs.
    - `tasks.js` : Gère les routes pour les opérations sur les tâches, y compris l'ajout, la modification, la récupération et la suppression des tâches.
    - **Fonction** : Chaque fichier dans ce dossier définit des routes spécifiques pour différentes parties de l'application, traitant les requêtes et interagissant avec la base de données pour réaliser les opérations demandées.

Interaction entre les Composants

- **Frontend et Backend** : Le frontend envoie des requêtes au backend via les API définies dans les fichiers de routes (`users.js` et `tasks.js`). Le backend traite ces requêtes, interagit avec la base de données et retourne les réponses appropriées.
- **API et Base de Données** : Les fichiers dans `routes` utilisent les configurations de base de données établies dans `config` pour interroger ou modifier les données dans la base de données selon les actions de l'utilisateur.

Répartition des tâches :

Les tâches ont été réparties de la manière suivante :

Initialement, nous n'étions pas informés que le projet pouvait être réalisé en binôme. Pierre WADRA avait déjà entamé le développement, nous avons donc

décidé de poursuivre sur la base de son travail préliminaire. À ce stade, la base de données était déjà établie, et une page de connexion existait sans être fonctionnellement reliée au reste de l'application.

L'infrastructure initiale du serveur avait également été mise en place dans le fichier `src/api/index.js`.

Nathan RUEFF a pris en charge le développement des routes dans `tasks.js` et `users.js`.

Par la suite, nous avons collaboré étroitement pour finaliser le projet. Cette collaboration incluait la complétion des routes, l'intégration avec la base de données, ainsi que la mise en place et le stylisme des pages HTML, CSS, et JavaScript situées dans le dossier `public`.

Axe d'amélioration

- **Implémentation de la modification des utilisateurs par l'administrateur** : Compléter la fonctionnalité permettant à l'administrateur de modifier les données des utilisateurs pour une gestion plus flexible et complète.
- **Amélioration de l'interface utilisateur** : Rendre l'interface plus intuitive et réactive. Ici nous nous sommes concentrés sur l'essentiel
- **Sécurité renforcée** : Renforcer les mesures de sécurité, en intégrant des pratiques plus sécurisée comme le RSA que nous avons vu dans certain groupe.

Conclusion

En conclusion, ce projet de gestion de tâches en ligne a permis de mettre en pratique des compétences variées en développement web, en alliant à la fois des

aspects back-end et front-end. À travers la réalisation de ce site, nous avons approfondi notre compréhension des API REST avec Node.js et ExpressJS, et nous avons également renforcé notre maîtrise des interactions entre le serveur et les bases de données, ainsi que des mécanismes d'authentification comme JWT.