Lina MEDANI
Groupe: Géryon

Audit des projets Lucarne et Destina

1- Projet Lucarne du groupe SudoPing:

Le projet **Lucarne** est conçu pour offrir aux utilisateurs une plateforme qui permet de rechercher des joueurs de football en fonction de plusieurs critères comme le nom, le club, le poste ou encore le pays. Il est disponible en deux versions: une version web et une version Desktop.
D'un point de vue technique, le projet repose sur un **frontend** développé avec TypeScript, Bootstrap, ElectronJS et HTML/CSS. Le **backend** fonctionne avec NodeJS(Express) et Electron IPC. Pour la **base de données**, l'équipe avait initialement prévu d'utiliser SQLite, mais après plusieurs tests, elle a finalement opté pour un fichier JSON, jugé plus adapté au volume de données traité .
L'application propose plusieurs fonctionnalités essentielles, notamment une recherche de joueurs basée sur l'algorithme de Levenshtein pour gérer les fautes de frappe, un tri des résultats selon leur popularité ou par ordre alphabétique, ainsi qu'une présentation claire des joueurs sous forme de liste avec leurs principales informations. Grâce à ElectronJS, elle est accessible aussi bien sur le web qu'en version bureau.
Cette application s'adresse avant tout aux passionnés de football qui souhaitent obtenir rapidement des informations sur des joueurs sans passer par des bases de données complexes ou des sites surchargés de publicités.

1-A- Analyse technique:

- **Qualité du code:**

Le code est plutôt bien organisé dans son ensemble, avec une séparation claire entre le frontend et le backend, ce qui facilite la navigation dans le projet. L'utilisation de TypeScript est un bon choix, car elle permet d'ajouter du typage statique et de réduire les erreurs courantes. Cependant, il manque des commentaires dans certaines sections critiques du code. Par exemple, la gestion de l'IPC (Inter-Process-Communication) ou la logique de traitement des données JSON ne sont pas suffisamment documentées. Ajouter des commentaires détaillés et expliquer l'objectif des fonctions faciliterait grandement la compréhension, surtout pour un autre développeur qui pourrait être amené à reprendre le projet.

La logique de l'application est relativement bien découpée, mais il serait possible d'aller encore plus loin. Par exemple, la base de données et la logique de recherche pourraient être extraites dans des modules distincts. Cela permettrait de rendre le code plus modulaire, réutilisable. Une telle séparation facilite l'ajout de nouvelles fonctionnalités à l'avenir et la maintenance de l'application, tout en rendant l'ensemble du code plus lisible et compréhensible.

Lina MEDANI
Groupe: Géryon

- **Performance:**

En l'état actuel, l'application fonctionne bien tant que le nombre de joueurs reste limité. Cependant, lorsque le volume de données augmente, l'utilisation d'un fichier JSON pour stocker les informations devient un point de friction. Si le projet venait à se développer et à accueillir un grand nombre de joueurs, cette solution pourrait rapidement devenir un goulot d'étranglement. Une migration vers une base de données plus adaptée, comme SQLite ou MongoDB, permettrait non seulement d'améliorer la gestion de données, mais aussi de rendre les recherches plus rapides et évolutives.

L'algorithme de Levenshtein utilisé pour la recherche approximative est bien choisi, mais il est recalculé à chaque fois qu'une recherche est effectuée. Cela peut devenir inefficace avec une grande base de données, car chaque recherche impliquerait de recalculer la distance de Levenshtein pour chaque joueur à chaque fois. Une solution serait d'optimiser cette partie du code en prétraitant les données, ou en utilisant des techniques d'indexation pour limiter les calculs à des sous-ensembles spécifiques de données, ce qui permettrait de réduire le coût de chaque recherche.

- **Sécurité:**

L'application Lucarne ne valide pas les entrées utilisateur, ce qui peut exposer l'application à des risques d'injection de scripts (XSS). Ajouter une validation pour nettoyer les chaînes de caractères saisies par l'utilisateur permettrait de renforcer la sécurité de l'application.

1-B- Evaluation fonctionnelle:

Le projet atteint son objectif en fournissant une interface intuitive pour rechercher les joueurs de football. Les critères de recherche, comme le nom, le club ou le poste, permettent aux utilisateurs de trouver rapidement les informations souhaitées. L'algorithme de Levenshtein est bien intégré pour corriger les fautes de frappe, et le tri des résultats par la popularité ou ordre alphabétique offre une bonne flexibilité.

Facilité d'utilisation et ergonomie:

L'interface est simple est bien structurée grâce à Bootstrap, ce qui assure une navigation fluide sur différentes tailles d'écrans. La barre de recherche, placée de la manière centrale, rend l'application facile à utiliser. Le design minimaliste aide à concentrer l'attention sur les informations essentielles, et la version bureau, via ElectronJS, fonctionne de manière fluide.

L'application fonctionne globalement bien, mais certains points peuvent être améliorées:

- Il serait utile d'afficher un message clair en cas de recherche sans résultat(comme, "aucun joueur trouvé")
- L'application manque de gestion d'erreurs pour les saisies invalides dans la barre de recherche. Un mécanisme de validation des entrées serait utile.
- Bien que la version Bureau fonctionne bien, des tests supplémentaires sur différentes plateformes (Windows, macOS) seraient nécessaires pour assurer une comptabilité complète .

Lina MEDANI
Groupe: Géryon

1-C- Documentation et Maintenabilité:

Le code du projet est relativement bien structuré, mais certains commentaires sont manquants, notamment dans des sections critiques comme la gestion d'IPC ou le traitement de données JSON. Ajouter des commentaires détaillés serait mieux.

En ce qui concerne la documentation, elle est présente dans une certaine mesure, mais elle pourrait être plus complète. Par exemple, bien que le readme fournisse des informations de base sur le projet et son installation, une documentation détaillée des fonctionnalités et de l'architecture serait utile pour mieux comprendre le fonctionnement global et faciliter la prise en main du projet.

Le processus d'installation est suffisamment documenté dans le readme. Les étapes nécessaires pour installer Node.js, les dépendances via npm install, et démarrer le serveur avec node app.js sont clairement indiquées .

Concernant l'évaluation du projet, il est relativement facile à faire évoluer, mais quelques améliorations peuvent être apportées. Par exemple, la logique de recherche et la gestion des données peuvent être séparées en modules distincts, ce qui améliorerait la modularité du code.

1-D- Recommandations et conclusions:

Points forts:

L'interface du projet est claire et bien structurée, ce qui facilite son utilisation. L'utilisation de TypeScript pour le frontend permet une meilleure gestion des erreurs grâce au typage statique. De plus, l'algorithme de Levenshtein est efficace pour corriger les fautes de frappe et améliore l'expérience utilisateur.

Point faibles:

Cependant, la documentation du code est insuffisante, et il manque des commentaires dans des sections critiques, ce qui rend la compréhension de code un peu plus difficile. L'utilisation d'un fichier JSON pour stocker les données devient un goulot d'étranglement dès que le volume de données augmente, ce qui peut nuire à la performance du projet. De plus, l'algorithme de Levenshtein est recalculé à chaque recherche, ce qui peut ralentir l'application lorsqu'il y a un grand nombre de joueurs. Enfin, l'absence de validation des utilisateurs expose l'application à des risques de sécurité.

Propositions d'amélioration:

Pour améliorer la maintenabilité du projet, il est recommandé d'ajouter des commentaires détaillés dans le code. Une migration vers une base de données plus robuste, comme SQLite ou MongoDb, permettrait d'optimiser la gestion des données et de rendre les recherches plus rapides. L'algorithme de Levenshtein pourrait également être optimisé pour éviter de recalculer la distance de chaque recherche.

Lina MEDANI
Groupe: Géryon

2- Projet Destina:

Le projet Destina est une application web permettant aux utilisateurs de rechercher des lieux d'entrée en fonction de critères variés tels que les thèmes, les pays et les villes. Son objectif est d'offrir une expérience fluide et intuitive pour la découverte de destinations. Le backend repose sur TypeScript avec Deno, tandis que la base de données utilisée est Neo4j. Le frontend est développé en React.

Parmi les fonctionnalités principales, on retrouve la possibilité d'effectuer des recherches de lieux par nom, thème ou relation entre lieux. L'application offre également une exploration dynamique grâce à l'utilisation d'une base de données graphe, garantissant une navigation fluide et intuitive. L'optimisation des requêtes et la gestion des performances sont également prises en compte avec des mécanismes tels que la pagination, le debouncing et la cache. Destina s'adresse principalement aux voyageurs en quête de nouvelles destinations, aux curieux intéressés par des lieux atypiques et aux personnes souhaitant organiser un voyage efficace.

2-A- Analyse technique:

Le code est bien structuré et suit les principes fondamentaux du développement moderne. L'utilisation de TypeScript assure un typage strict, facilitant la maintenance et réduisant les erreurs potentielles. L'architecture est organisée de manière claire avec une séparation nette entre le serveur, les routes et les contrôleurs. Oak est employé pour la gestion des routes et des middlewares, ce qui contribue à la lisibilité et à la modularité du projet. Cependant la gestion des erreurs est absente, l'implémentation d'un système de gestion des erreurs avec try/catch améliorerait la robustesse de l'application.

Le frontend repose sur du React, couplé à react-router-dom pour assurer une navigation fluide et efficace entre les différentes pages d'application. Des feuilles de style CSS externes sont utilisées pour structurer l'interface utilisateur.

D'un point de vue des performances, plusieurs optimisations ont été mises en place. La pagination réduit la charge serveur en limitant la quantité de données chargées en une seule requête. De plus, la mise en cache des images et le chargement différé améliorent l'expérience utilisateur en réduisant les temps de chargement. L'implémentation de debouncing limite le nombre de requêtes envoyées inutilement au serveur. Toutefois, des axes d'amélioration subsistent, notamment en veillant à une indexation efficace des données Neo4j et en intégrant un mécanisme de pré chargement intelligent des ressources.

En matière de sécurité, Deno apporte un certain niveau de protection grâce à sa gestion stricte des permissions, limitant les accès aux ressources sensibles. Cependant, il serait judicieux d'ajouter des vérifications plus poussées sur les entrées utilisateur afin de prévenir les potentielles injections Cypher. Par ailleurs, une sécurisation renforcée des accès APU et du stockage des données serait un plus pour garantir une robustesse accrue du système.

Lina MEDANI
Groupe: Géryon

2-B- Evaluation fonctionnelle:

L'application Destina remplit bien son objectif principal en fournissant des résultats pertinents et en facilitant l'exploration des lieux. L'interface est bien pensée, avec une navigation fluide et des filtres efficaces. La présentation des résultats est claire permettant aux utilisateurs de trouver facilement des informations.

Cependant, quelques problèmes subsistent:

- Certains lieux manquent d'informations détaillées ce qui nuit à l'expérience utilisateur.
- Un visuel par défaut pourrait améliorer la présentation. L'ajout d'images issues de sources libres, ou la possibilité pour les utilisateurs d'ajouter leurs propres images, pourrait renforcer l'attrait de l'application.

- Une optimisation supplémentaire des requêtes est recommandée. L'implémentation d'index pour les recherches les plus fréquentes et l'optimisation de jointure des requêtes Cypher pourraient considérablement améliorer la réactivité du système.

2-C- Documentation et maintenabilité:

Le code de Destina est généralement bien structuré, mais il manque certains commentaires explicatifs. Par exemple, les fonctions et méthodes principales, celles liées à l'interaction avec la base de données Neo4j ou à la gestion des requêtes complexes, bénéficieraient grandement de commentaires détaillant leur logique. L'ajout de commentaires clarifiant le rôle des principales parties du code améliorerait la compréhension du projet.

L'installation est partiellement documentée, mais des informations supplémentaires sur les versions requises de Deno, les dépendances et la gestion des erreurs lors de l'installation seraient utiles.

Le projet est modulaire et relativement facile à faire évoluer. Cependant, sans documentation technique complète et des bonnes pratiques claires, l'ajout de nouvelles fonctionnalités pourrait devenir plus compliqué.

2-D- Recommandations et conclusions:

Points forts:

- Utilisation de technologies modernes et performantes (Deno, TypeScript, Neo4j, React)..

- Interface fluide et intuitive, optimisée pour l'exploration dynamique des lieux.

- Mise en place de techniques d'optimisation comme la pagination, le caching et le debouncing.

- Sécurisation partielle grâce aux permissions strictes de Deno.

Lina MEDANI
Groupe: Géryon

Points faibles:

- Manque de commentaires dans le code, compliquant la compréhension des fonctionnalités clés.
- Absence de gestion avancée des erreurs et des validations d'entrées utilisateur.
- Certaines requêtes Cypher pourraient être optimisées pour de meilleures performances.
- Informations sur certains lieux limitées, réduisant l'expérience utilisateur.

Propositions d'amélioration:.

- **Optimiser les performances :** Mettre en place des index Neo4j et améliorer la gestion des requêtes pour réduire les temps de réponse.
- **Renforcer la sécurité :** Ajouter une validation stricte des entrées utilisateur pour éviter les injections Cypher.
- **Améliorer l'expérience utilisateur** : Ajouter des images par défaut pour les lieux manquant d'informations et permettre aux utilisateurs d'en ajouter.