Jules HIRTZ
Yann MIJATOVIC
Théo PINCHON
Alexandre PERROT

Mimir
Tuteur : Yann BONIFACE
Année 2023 - 2024

MIMIR

Récapitulatif 3ème itération

IUT nancy Charlemagne

Sommaire

Planning prévisionnel.....3
- Itération 3 (28h du 22/01 au 27/01).....3

Description du fonctionnement de l'IA.....3

BDD et Back-end :.....4

Decks :.....4

Users :.....4

Composants React et visuels :.....5

Prévision prochaine itération.....6

Récapitulatif.....6

Planning prévisionnel

- **Itération 3 (28h du 22/01 au 27/01)**

Réaliser l'application fonctionnelle / Lien entre le front, le back et les données (Toute l'équipe)
Composants React et visuels (Yann / Alexandre)
BDD et Back-end (Théo / Jules)

Description du fonctionnement de l'IA

Le modèle d'IA que nous utilisons est Vigogne, un sous-modèle de Vicuna, mais fine-tuné en français, ce qui va grandement nous être utile pour créer des questions et des réponses en français.
Pour faire simple, le modèle est récupéré sur HuggingFace avec un fichier .gguf (https://huggingface.co/TheBloke/Vigogne-2-7B-Instruct-GGUF Q4_K_M) qui le contient. On installe le fichier dans un répertoire (sur Linux), puis on crée un fichier "Modelfile" qui permettra de définir un template pour le modèle, c'est à dire la façon dont il va répondre au prompt. Dans notre cas, on lui indique un format de JSON à respecter en retour à chaque questions/réponses ainsi que juste lui indiquer qu'on lui donne un contexte à chaque fois où il devra générer des questions/réponses.
Une fois ce Modelfile crée, on initialise le modèle avec son template dans l'application **Ollama** installé sur la machine. Cette application permet une fois installé de faire tourner des modèles d'IA sur sa machine à l'aide du CPU et du GPU. Plus le GPU est puissant, plus l'IA aura un temps de traitement court (assez logique).

On fait tourner le modèle en permanence sur un VPS avec une très bonne carte graphique + CPU, ce qui permet d'envoyer des requêtes POST de part un site web vers l'API de Ollama permettant de transmettre un prompt à un modèle d'IA ainsi que différents autres paramètres. Dans notre cas, on indique aussi dans la requête POST que le retour doit être un JSON. Dans le prompt transmis au modèle, on lui ré indique le format désiré de JSON pour maximiser nos chances que l'IA ne décide pas de faire un autre format que celui désiré. On indique aussi à la fin du prompt le nombre de questions et de réponses voulues.

De cette façon, le modèle nous génère des questions et des réponses sur le contexte donné dans le prompt. Pas trop de soucis avec le format retourné par l'IA, seulement des questions/réponses bateaux et le nombre de questions/réponses transmises n'est pas toujours celui attendu.

BDD et Back-end :

Récupérer les données de la BDD pour les afficher dans l'interface utilisateur est un cap indispensable au projet. Dans notre environnement NextJs, chaque appel à la BDD correspond à une requête HTTP sur des Endpoints.

Un Endpoint est une page du serveur ne contenant aucune interface graphique, nous avons donc mis en place une architecture de type CRUD (Create Read Update Delete). Dans ce cas, chaque page doit définir, pour son bon fonctionnement, des méthodes correspondant aux requêtes http telles que : GET, POST, PUT ou DELETE. Chaque endpoint est donc limité à 4 méthodes maximum et chacune d'entre elles manipule des objets de type Model provenant de la librairie mongoose. Ce type d'Objet permet l'utilisation de méthodes triviales dans le domaine de la bdd comme par exemple : find, update ou save.

Decks :

Étant le format principal des données manipulées, le model Deck est utilisé actuellement dans 3 endpoint soit 6 méthodes. Ce nombre devrait légèrement fluctuer selon les différents besoins de l'application.

On remarquera qu'aucune requête ne concerne les cartes seules. Nous avons choisi de manipuler les collections de cartes uniquement dans le côté utilisateur de l' application. Chaque page récupère le deck qui la concerne et en utilise les propriétés pour lire, modifier ou créer des cartes.

Pour faciliter l'appel à ces méthodes utilisables uniquement par des "fetch", nous les avons abstraites dans un fichier de requêtes "deck-requests". L'utilisation ne se fait alors qu'avec des fonctions javascript classiques.

Users :

Dans l'optique de pouvoir afficher le profil d'un utilisateur, nous avons créé de quoi modifier l'utilisateur en cours de navigation dans l'app (le current user de la session). Nous avons simplifier la requête à l'api en rajoutant une couche d'abstraction grâce à l'implémentation d'une méthode fetchCurrentUser et updateCurrentUser.

Pour pouvoir faire le système de contacts,nous avons dû effectuer quelques modifications notamment sur le modèle de la base de données puisqu'il n'était pas parfaitement adapté.

Ensuite nous avons mis en place deux nouveaux endpoint :

- Pour l'ajout d'un nouveau contact
- Pour la suppression d'un contact

Et une surcouche d'abstraction pour pouvoir récupérer facilement dans le front-end ses contacts, les ajouter ou les supprimer.

Composants React et visuels :

Composants :

Deck :

On a tout d'abord commencé à créer 3 composants différents pour les decks :

- Un deck personnel, qui est affiché avec des boutons de modifications, partage et suppression
- Un deck public qui est affiché avec son auteur, son nombre de pages et ses upvotes/downvotes
- Un deck statistiques avec le nombre de questions validée/échouée par l'utilisateur dans la page statistiques

On a ensuite changé puisque ce type de visuels ne nécessite qu'un composant, juste un footer qui change selon le type.

Profil :
Nouvelle interface basée sur la maquette figma reconstruite récemment. Création de la page d'édition avec un formulaire pour changer sa biographie et d'autres informations liées à l'utilisateur.

Contact :

Après avoir mis en place le système de contact au niveau du back-end, on a donc réutilisé les composants front-end qui avait été fait à l'itération précédente, et nous avons lié le back-end avec le front-end. Notamment en visualisant les contacts qu'on a ajoutés, en recherchant un nouveau contact et l'ajouter et supprimer un contact.

Pages et autres visuels :

La majorité des visuels ont été retravaillés avec l'aide de TailwindCSS et liés au back suite au merge réalisé au début de l'itération, notamment :

- L'accueil
- Connexion + inscription
- Homepage de l'user
- Mes decks
- Marketplace (explorer)
- Contacts
- Profil

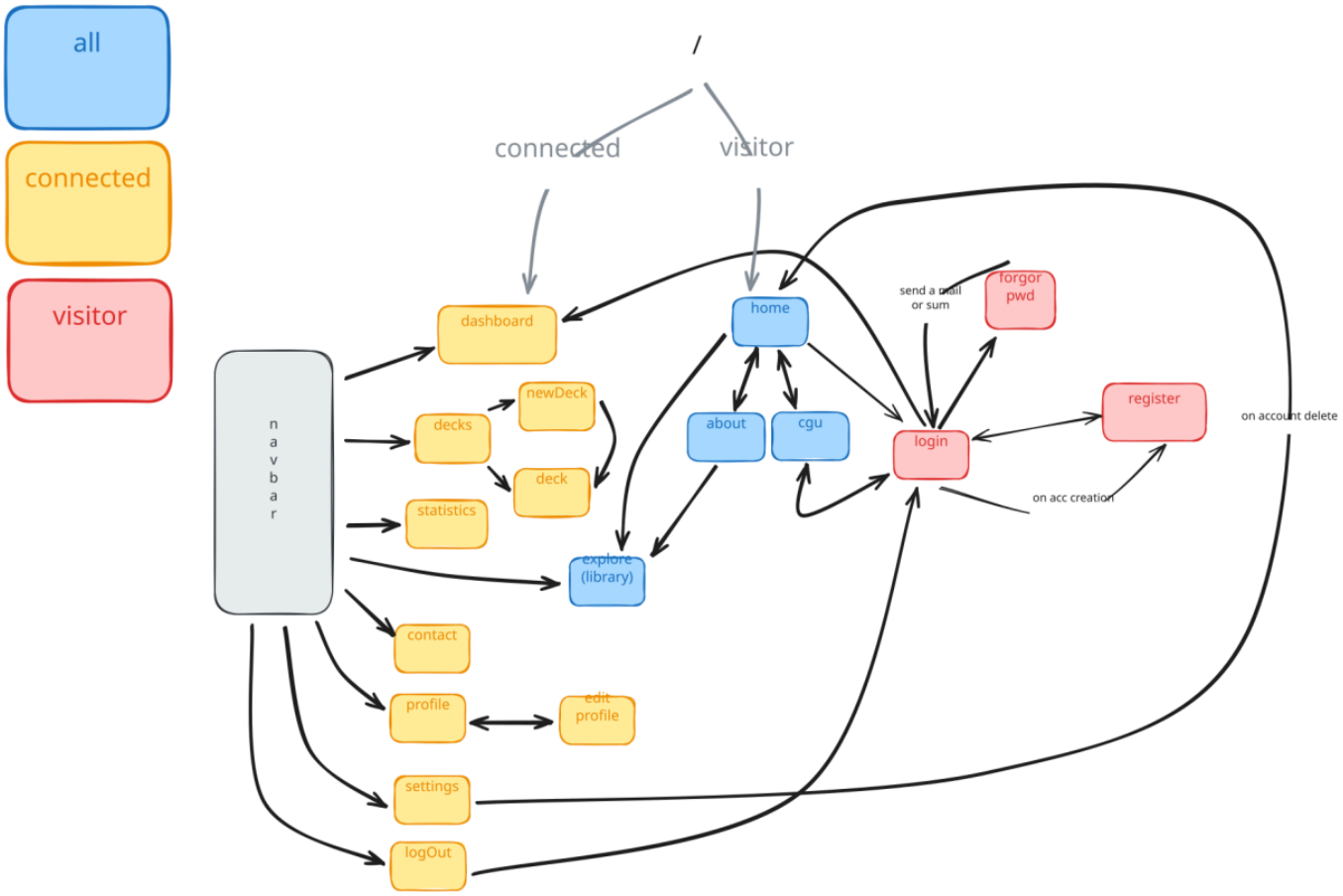
Certaines pages ont vu leur visuel changer, mais le back n'a pas forcément eu le temps d'être fait lors de cette itération comme :

- Paramètres

- Statistiques (WIP)
- Création/Edition/Visualisation des decks en détails

Expérience utilisateur :

Redesign de l'user flow en fonction de la session :



Prévision prochaine itération

L'un des objectifs majeurs pour la prochaine itération est de pouvoir répondre à un deck depuis la nouvelle interface et également d'intégrer l'intelligence artificielle à la création des cartes.Par ailleurs, il est prévu d'améliorer la création des cartes en intégrant un éditeur de texte plus modulable. Nous prévoyons également de mettre en place les améliorations de l'user flow (modification des liens entre les pages pour plus de fluidité). Pour terminer, nous comptons renforcer la sécurité des comptes utilisateur en créant un module de vérification de mail et la possibilité de changer son mot de passe, également grâce à un mail de vérification.

Récapitulatif

Durant cette itération, nous avons donc pu reproduire entièrement la maquette que nous avions réalisé à l'itération 2 afin d'avoir un site Internet plus esthétique et ergonomique, de plus qu'on a pu réaliser des composants qui nous permettent d'intégrer les données de la base de donnée sans faire de modification majeure aux fichiers ce qui accélère grandement le processus de liaison.

Mercredi après-midi, nous avons fait une avancée majeure, car nous avons fait un merge réunissant le travail de toute l'équipe (Backend / Frontend) et qui nous permet d'avoir enfin un projet fonctionnel.

Nous avons pu terminer complètement certaines fonctionnalités, mais nous allons continuer à terminer le reste des fonctionnalités durant l'itération 4 notamment le fait de pouvoir réviser ses decks.

Nous avons donc actuellement une application qui possède certaines fonctionnalités mais qui lui manque ses fonctionnalités principales.