Jules HIRTZ
Yann MIJATOVIC
Théo PINCHON
Alexandre PERROT

Mimir
Tuteur : Yann BONIFACE
Année 2023 - 2024

MIMIR

Récapitulatif 5ème itération

Sommaire

Planning prévisionnel

- **Itération 5 (28h du 07/01 au 21/02) attendu**

Ajouter les fonctionnalités optionnelles si possible (Jules)
- Question par image
- live-code

Finaliser fonctionnalités optionnelles (Yann / Jules)
Continuer à tester l'application et régler les bugs (Alexandre)
Peaufiner l'application (Théo)
Préparer soutenance fin de semestre (Toute l'équipe)

- **Itération 5 réel**

Refactoring de tout le layout front-end (Alexandre)
Réorganisation de l'affichage des decks (Yann)
Changement de l'enregistrement des statistiques (Yann)
Mise en place d'un système de mail (Théo)
Intégration d'un éditeur complexe : LaTeX, images, … (Jules)
Système de partage de decks entre contacts (Alexandre)
Intégration des providers pour l'authentification (Alexandre)

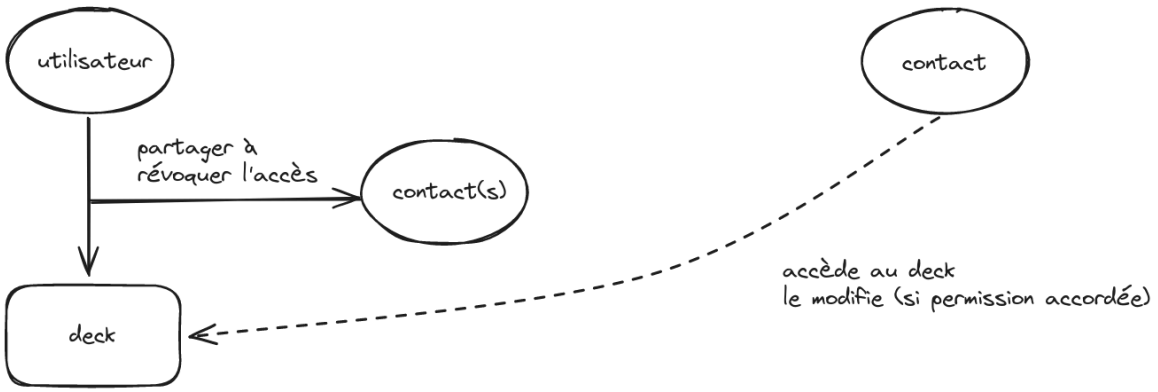
Gestion de partage de decks

Après avoir créé son deck, le site n'avait qu'un seul moyen de le partager pour l'instant : via le marketplace (et seulement si le deck était mis en public par l'utilisateur). Il a fallu donc mettre en place un système qui puisse partager ses decks (public ou privé) avec ses contacts au choix.

Il y a aussi la possibilité d'accorder à son contact la permission de modifier le deck en question (pour le corriger ou ajouter des cartes)

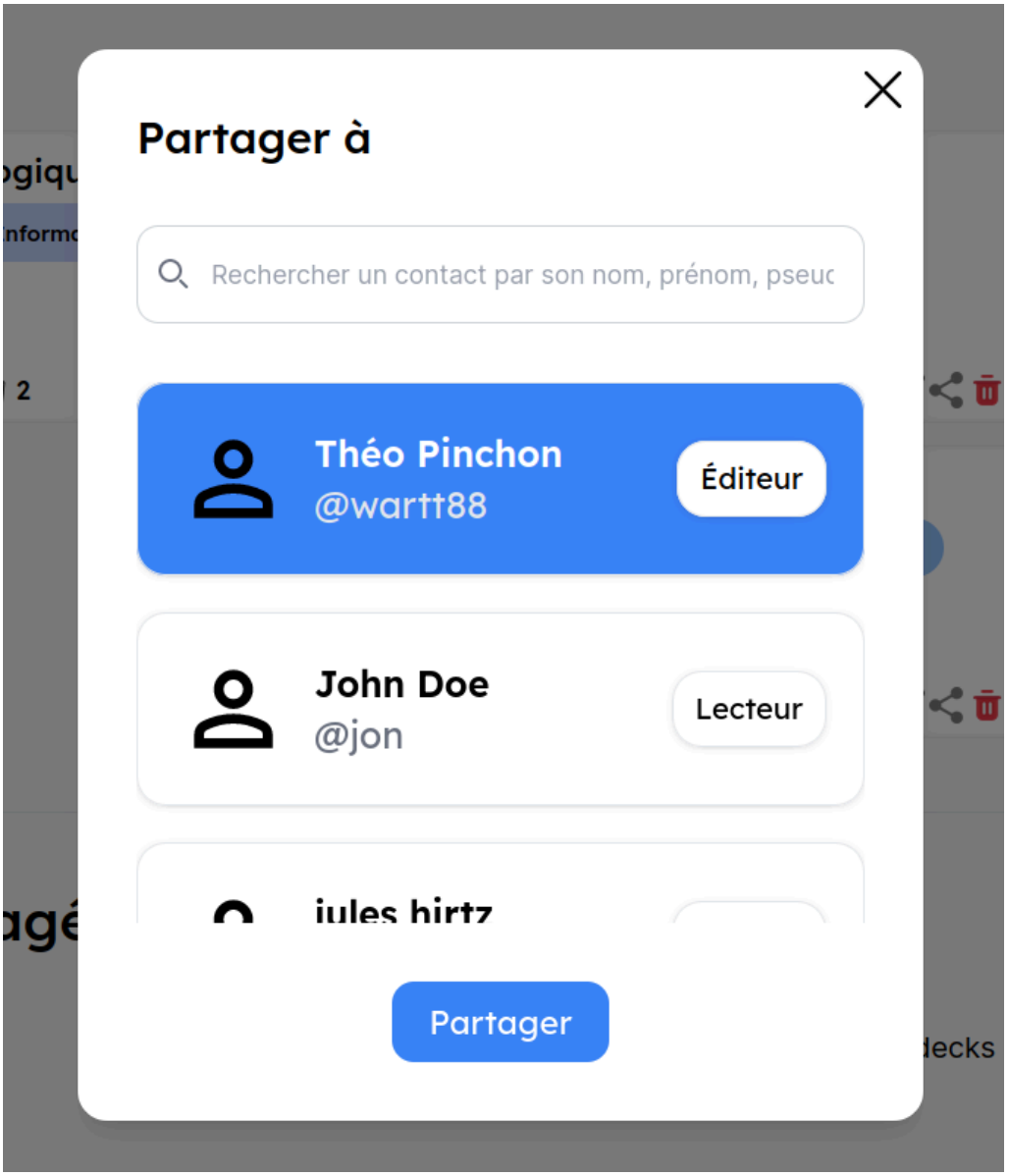
Il y a une subtilité de ce système par rapport à celle de l'importation depuis le marketplace, c'est que au lieu de "copier" le deck du créateur vers le compte de l'utilisateur qui a copié, l'utilisateur aura juste accès au deck (et peut le modifier) mais, le deck ne sera pas copié. Ce qui veut dire que lorsqu'un utilisateur joue ce deck (pour s'entraîner), les statistiques de ce deck seront directement mise chez le créateur, et non chez l'utilisateur qui à joué ce deck, comme le ferait l'importation d'un deck. Cela permettra donc d'englober les statistiques et de faire des tendances global pour un deck au lieu d'avoir des statistiques par utilisateur.

Voici un schéma qui est censé refléter l'idée de la fonctionnalité



Pour implémenter cette fonctionnalité dans notre application, il a fallu faire ces étapes :

- Ajout d'une nouvelle information dans la base de données MongoDB pour savoir qui a partagé quoi et à qui comme deck.
- Mise en place d'un popup front-end pour afficher la liste des contacts et à qui partager (cf. screenshot)



- Mise en place de toute la logique métier (API) qui ajoute l'information que le deck a été partagé à tel utilisateur avec tel autorisation (lecteur ou éditeur)
- Mise en place d'un système d'affichage des decks partagés (récupération de tous les decks partagés avec moi, cf. screenshot)

Réorganisation de l’affichage des decks

Pendant cette itération, nous avons retouché la façon dont les decks sont affichés dans le dashboard et dans la page “Mes decks”.

Dans la page “Mes decks”, les mauvais decks étaient affichés (deck public), il fallait donc seulement modifier quelques lignes afin de pouvoir afficher les decks correspondant à cette section du site, qui sont les decks ayant pour owner_id l’identifiant de l’utilisateur.

Il reste ensuite les trois sections dans le dashboard, la page par défaut d’un utilisateur une fois connecté, ou l’accueil tout simplement. Il y‘a 3 sections qui affichent par défaut tous les decks dans la base de données distante Atlas MongoDB.
La première section à modifier fut celle de l’historique récent, soit les derniers decks que l’utilisateur a FINI de parcourir (pas forcément entièrement, valide si il a quitté le deck en plein milieu tant qu’il est arrivé sur la page de fin de deck). Cette section répertorie donc les 10 derniers decks que l’utilisateur a visité au long de son utilisation du site. Pour récupérer les bons identifiants de chaque deck, nous passons donc par une liste de decks attribuées à chaque utilisateur du site. Cette liste est donc incrémentée à chaque deck complété. Cette liste se mettra à jour pour chaque deck fini par l’utilisateur, et remettra en tête de liste le dernier deck.

La deuxième section représente les recommandations proposées à l’utilisateur. Ces recommandations se basent sur l’historique récent, et plus précisément sur les tags des decks de l’historique récent de l’utilisateur. Cette section sera donc remplie par les decks avec des tags correspondant à ceux dans les decks contenus dans l’historique MAIS ne prendra pas en compte les decks déjà présents dans l’historique (dans les recommandations de decks). Les tags les plus présents dans l’historique de l’utilisateur seront plus mis en avant dans les propositions de decks (pour être plus en cohérence avec les préférences utilisateurs).

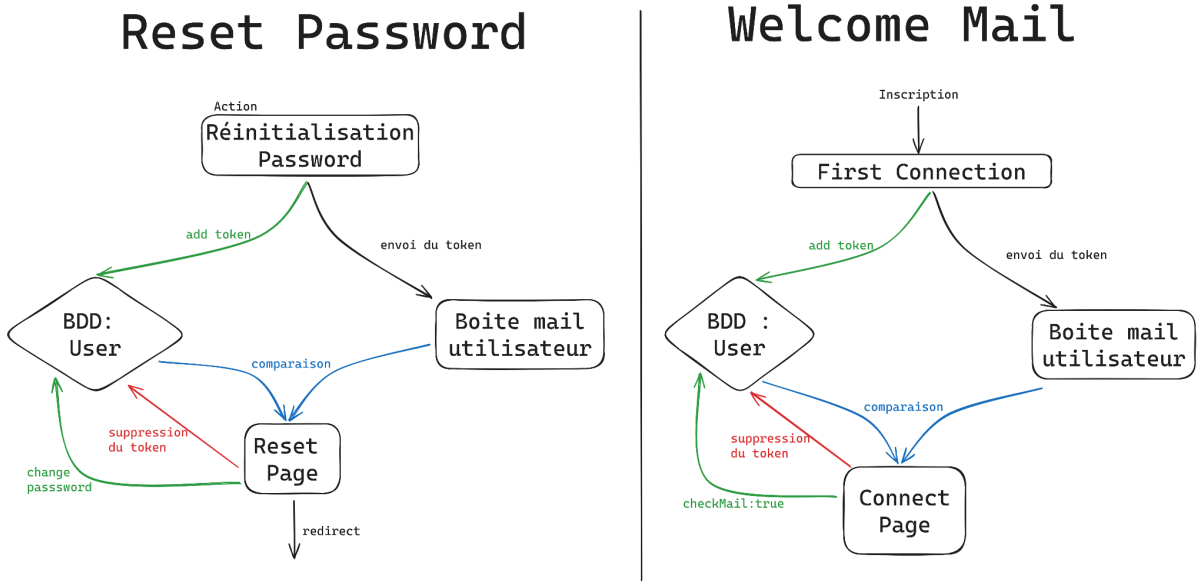
La dernière et troisième section du dashboard sont les decks partagés. Celle-ci fut la plus simple à rajouter, sachant qu’il fallait juste avoir besoin d’implémenter la fonctionnalité de pouvoir partager des decks, modifier la structure de l’utilisateur dans la BDD pour ajouter les decks partagés. Il ne reste ensuite plus qu’à récupérer les ID des decks contenus dans cette nouvelle partie du document et d’afficher les decks récupérés.

Serveur de mail

Afin de renforcer la sécurité et le bon fonctionnement de l'application, certaines fonctionnalités telles que la vérification de l'adresse mail à la première connexion ou le lien de " mot de passe oublié " se sont imposées comme nécessaires.

Pour se faire, nous avons décidé d'utiliser un service externe de distribution de mail appelé resend. Ce service propose divers abonnements dont un gratuit proposant 3000/mois soit 100 mails à la journée donc suffisant pour notre application, toujours à petite échelle. La seule condition d'utilisation du service est de fournir un nom de domaine servant de point de passage à la distribution des mails. Nous avons donc utilisé le nom de domaine possédé par Alexandre.

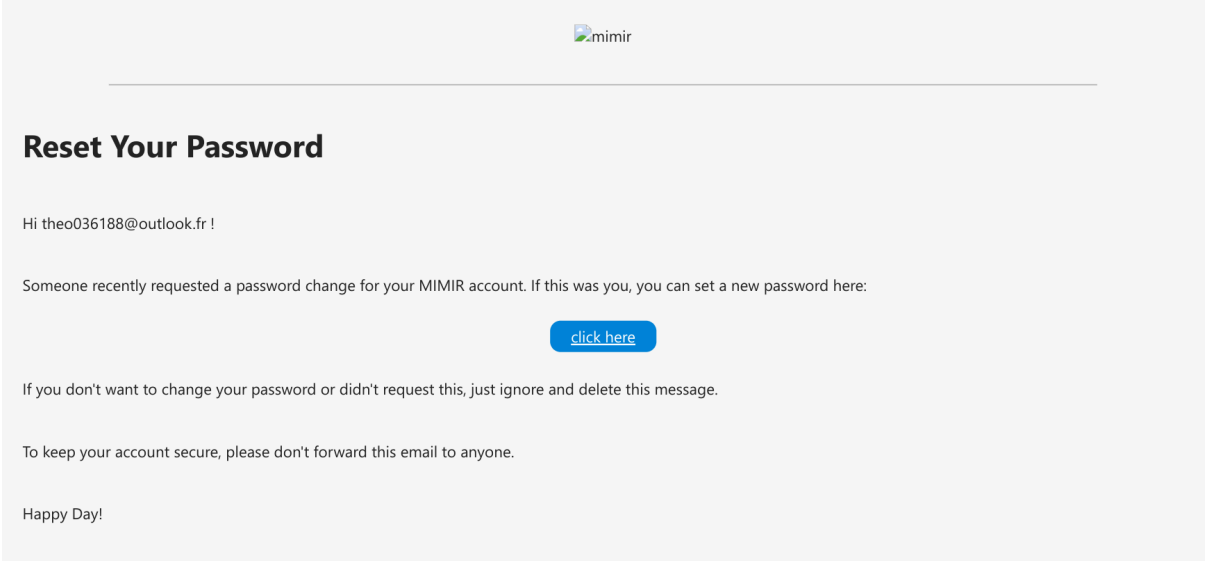
Comme précisé en amont, les mails sont utilisés comme transport d'un token soit de vérification de l'adresse mail soit de réinitialisation de mot de passe. Situations représentées par le schéma suivant :



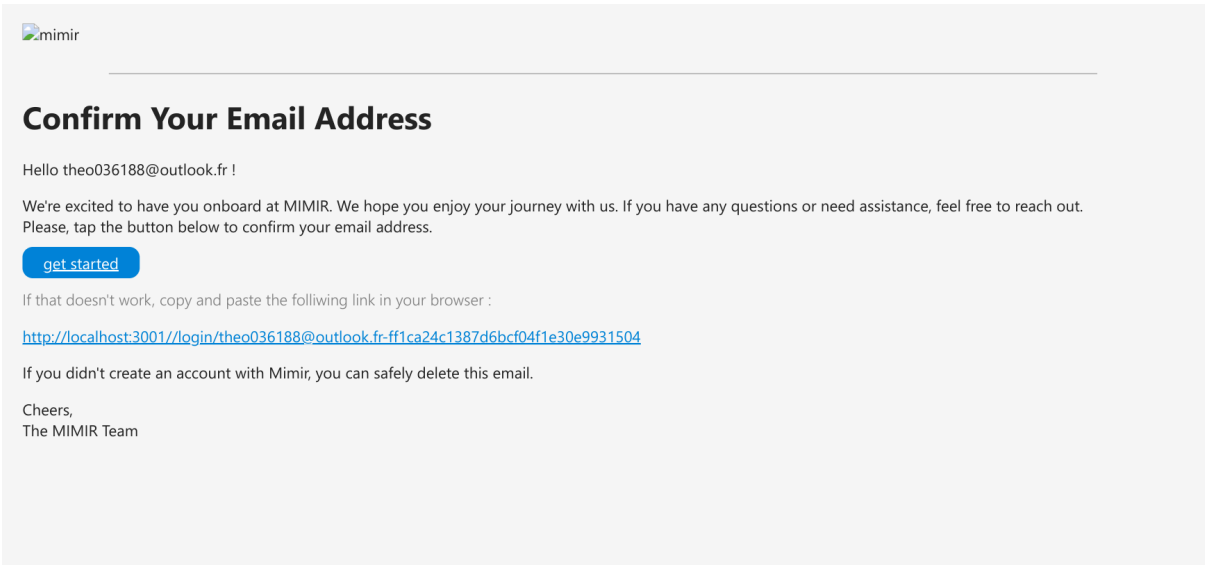
En complément, il faut garder en tête que tant que l'utilisateur n'a pas validé son adresse mail, il ne peut ni se connecter à l'application, ni réinitialiser son mot de passe.

En ce qui concerne les difficultés rencontrées, elles concernent toutes la structure d'un mail. Cette dernière requiert une arborescence stricte et du css in html. De plus, les images doivent être hébergées en ligne ou intégrées aux pièces jointes du mail. Néanmoins, les mails sont dorénavant implémentés et obligatoire à la création d'un compte. Voici donc le format des mails :

- Reset Password :

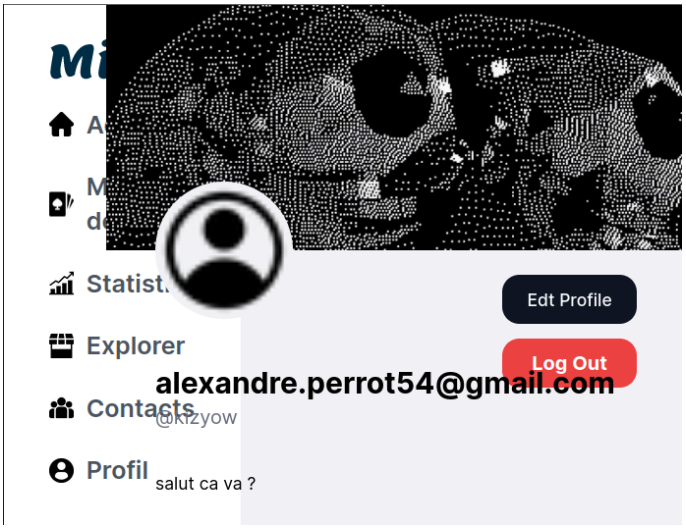


- Welcome mail :

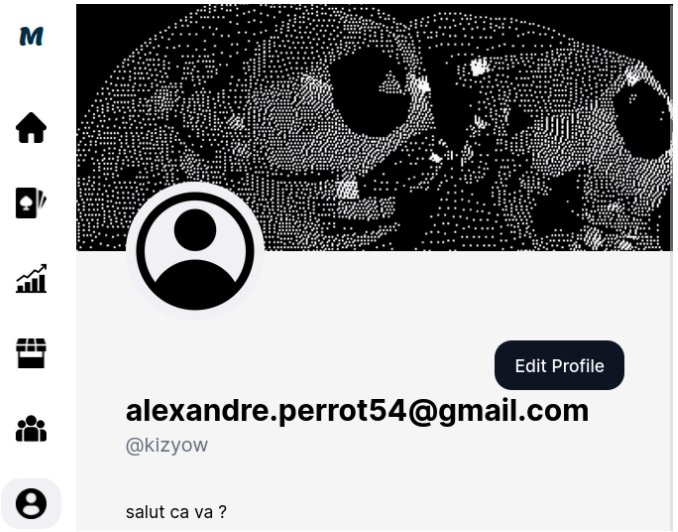


Refactoring du layout front-end

Notre application avait des problèmes d'affichage front-end, car nous n'avions pas implémenté chaque page de la même façon et de plus, nous n'avions surtout pas fait de code responsive (qui ne s'adapte pas à tout taille d'écran), ce qui causait des problèmes d'affichages.



Affichage avant le refactoring



Affichage après le refactoring

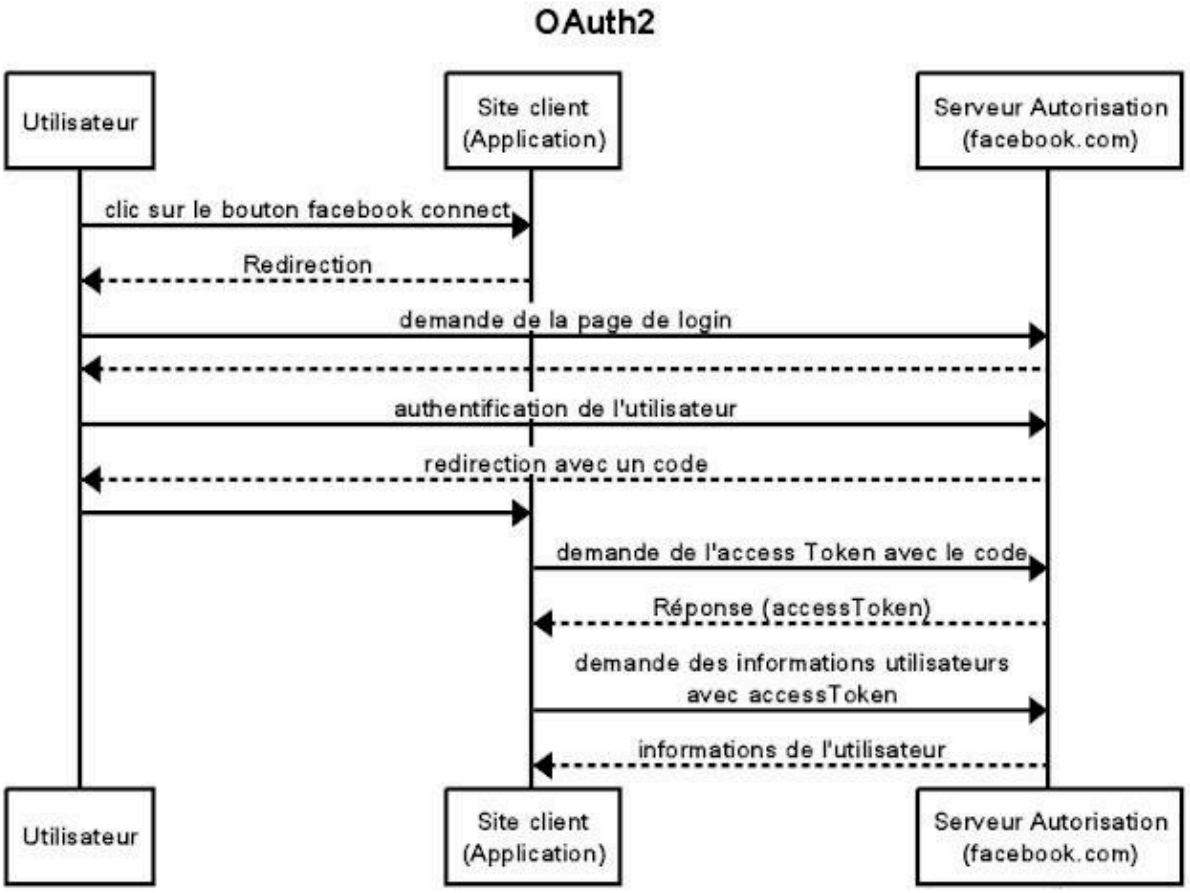
On a dû modifier toutes les pages web afin de s'adapter correctement au nouveau layout, qui est cette fois-ci responsive. Et cela a causé un grand changement au niveau de la structuration des fichiers dans le projet.

Intégration des providers pour l'authentification

Nous voulons intégrer un moyen de s'authentifier à notre application sans mot de passe et rapidement, mais tout autant sécurisé, en utilisant des providers reconnus et de confiance (tel que Apple, Google, Facebook, …)

Dans notre cas, nous avons choisi d'utiliser Google et GitHub en tant que providers afin d'authentifier l'utilisateur.

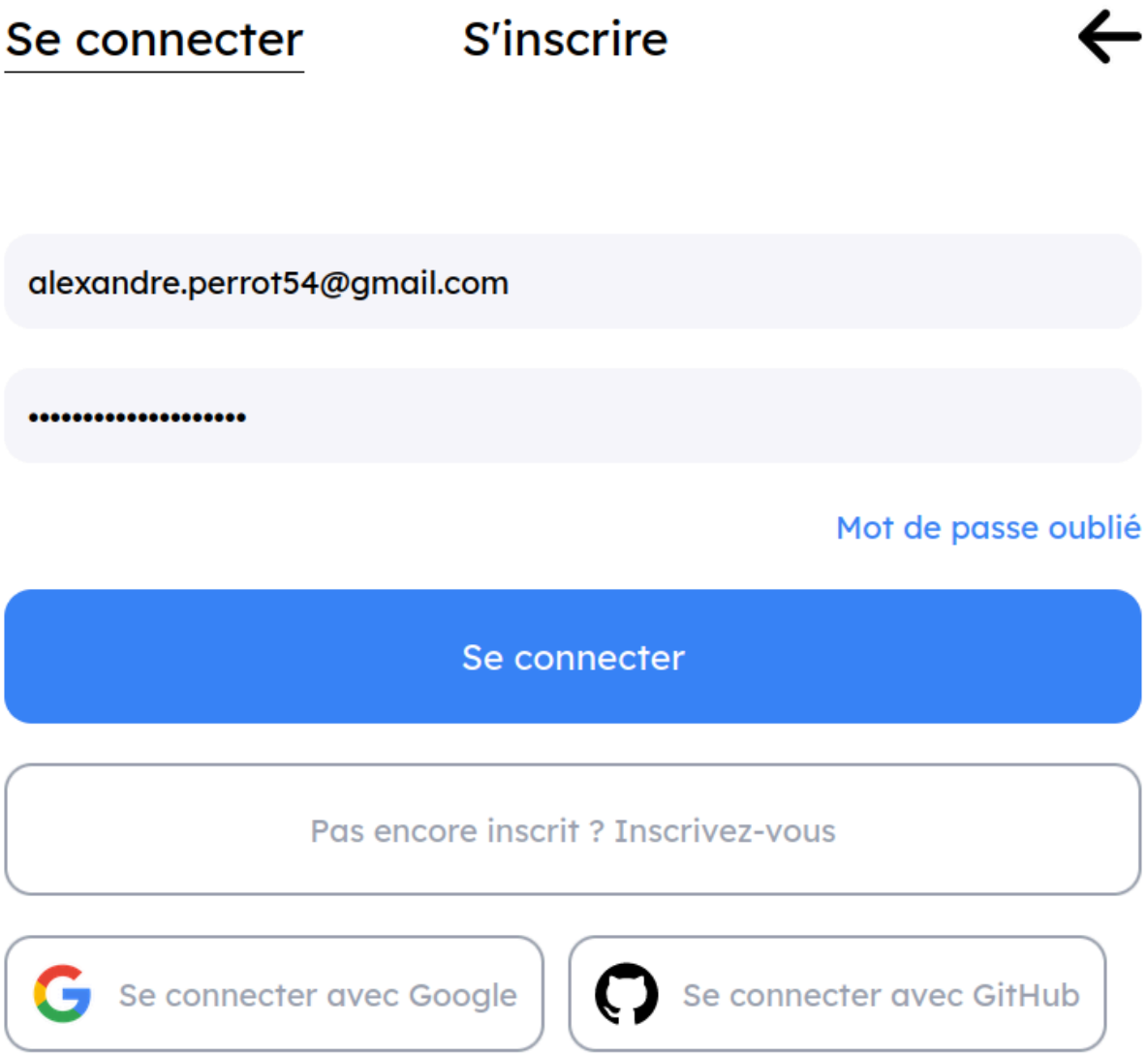
L'idée de ces systèmes est qu'il utilisent un protocole reconnu qui ne somme OAuth2, ci-dessous un schéma détaillé qui explique son fonctionnement



Implémenter cette fonctionnalité a été simple puisque la librairie NextAuth propose d'utiliser des providers externes (il suffit simplement d'avoir une clé API de chaque provider pour que cela fonctionne).

Nous n'avons pas changé la logique de l'application, puisque NextAuth se charge d'initier la connexion avec les providers et de retourner un compte utilisateur associé au mail du provider (e-mail Google : alexandre.perrot54@gmail.com retournera un compte utilisateur avec l'email alexandre.perrot54@gmail.com sur Mimir)

Et pour finaliser la fonctionnalité, on a rajouté deux boutons sur la page de connexion proposant la possibilité de se connecter soit par mail/mot de passe, soit par Google ou soit par GitHub.



Création d’un clone de notion pour apprendre à réaliser un éditeur de texte complexe en typescript

La première étape de ce projet a été l'initialisation d'un nouveau projet TypeScript. J'ai défini un `package.json` pour gérer les dépendances et j'ai installé TypeScript ainsi que les types Node.js pour assurer la compatibilité avec l'environnement Node.js .

Ensuite, j'ai structuré le projet en créant des dossiers pour les composants, les services et les ressources, comme les images. Cela a permis d'organiser le code de manière logique et de faciliter la maintenance et l'évolutivité du projet.

Pour le cœur de l'éditeur, j'ai dû développer un composant principal qui intègre un champ de texte pour l'écriture des utilisateurs. Ce composant a été conçu pour être réactif et pour gérer l'état du texte saisi par l'utilisateur. Chaque frappe de clavier a été capturée et l'état du texte a été mis à jour en conséquence, garantissant ainsi une expérience utilisateur en temps réel .

L'un des défis majeurs était l'ajout de la fonctionnalité d'insertion d'images. J'ai conçu un composant d'image qui permettait aux utilisateurs de sélectionner une image depuis leur ordinateur et de l'insérer dans l'éditeur. Ce composant a été intégré à l'éditeur de texte, offrant ainsi une fonctionnalité de collage d'images .

Pour améliorer l'interface utilisateur, j'ai utilisé du CSS et une bibliothèque de composants comme Tailwind CSS surchargée par shadcn, pour styliser l'éditeur. J'ai veillé à ce que l'interface soit attrayante et facile à utiliser, avec une attention particulière portée à l'affichage correct des images .

Récapitulatif

Sachant que depuis l'itération 4, nous avons une application fonctionnelle avec les features principales auxquelles nous tenions à coeur et les fonctionnalités *sine qua non* pour que l'application fonctionne correctement, nous avons voulu concentrer cette itération sur des choses moins principales, mais de tout de même intéressante pour une meilleure expérience utilisateur.

Nous sommes tout de même assez contents car le travail que nous avons fourni lors de cette itération a permis de renforcer la QoL déjà existante de notre application web ainsi que la sécurité de connexion à l'application Mimir.

Nous avons donc en fin de compte un site fonctionnel plus performant avec encore plus de fonctionnalités.

Objectif pour l'itération 6

- Page Statistiques (Yann)
- Build le projet (Théo)
- Implémenter l'éditeur de texte (Jules)
- Implémenter du LateX
- Ajouter des images aux cartes (Jules)
- Vérification pour la création d'un deck (Alexandre)
- Images dans les profils (Alexandre)