

27/06/2025

projet étudiant

**CESI**  
ÉCOLE D'INGÉNIEURS

# **Rapport**

# **Breezy**

Adrien MOLINA  
Théo CAVAGNE

Ilyass TRAN  
Jules BESSON  
Guillaume FINOTTO

## Table des matières

1. Introduction .....	2
a. Objectifs et attentes du projet.....	2
2. Plan de départ.....	3
a. Maquette.....	3
b. Architecture logicielle .....	9
3. Organisation .....	11
a. Méthodes Agiles .....	11
b. GitHub .....	11
c. Hiérarchisation des tâches et fonctionnalités .....	12
d. Répartition du travail .....	13
4. Étapes et ressources mobilisées.....	13
5. Rendu.....	15
a. Features validées.....	15
b. Présentation générale .....	18
6. Axes d'améliorations .....	22
7. Conclusion .....	23

## 1. Introduction

Ce rapport présente le travail réalisé par notre groupe dans le cadre du développement de l'application Breezy, un réseau social conçu pour permettre aux utilisateurs de partager des messages courts sous forme de postes. À travers ce document, nous retracerons les différentes phases du projet, de la conception à la mise en œuvre, en détaillant à la fois notre réflexion initiale, notre organisation en équipe, et les aspects techniques de la solution développée.

Nous commencerons par une description de la phase de préprojet, au cours de laquelle nous avons défini notre vision de l'application à travers une maquette fonctionnelle et un schéma d'architecture logicielle. Dans un second temps, nous expliquerons notre méthodologie de travail, inspirée des principes agiles, ainsi que l'usage de GitHub pour la gestion collaborative du projet. Enfin, nous conclurons ce rapport par une présentation du résultat final, en soulignant les fonctionnalités implémentées, ainsi que des pistes d'amélioration envisagées pour de futures évolutions de l'application.

### a. Objectifs et attentes du projet

L'objectif principal de Bredy est de créer un réseau social léger et réactif, inspiré de Twitter/X, permettant aux utilisateurs de partager des messages courts, de liker, commenter, suivre d'autres membres, envoyer des messages privés, et recevoir des notifications en temps réel.

Ce projet vise aussi à expérimenter une architecture micro-services conteneurisée, évolutive et sécurisée. Il s'agit pour nous de maîtriser les outils modernes comme Docker, Node.js, React, MongoDB et l'hébergement sur AWS EC2.

## 2. Plan de départ

### a. Maquette

La maquette du projet a été créée à l'aide de l'application Figma, un outil de design d'interface (UI/UX) qui permet de concevoir des maquettes, des prototypes interactifs et de collaborer en temps réel. C'est l'application de maquettage web la plus utilisée dans le milieu du design.

Nous avons choisi de partir d'une maquette Figma open source inspirée de Twitter. Cela nous a permis de disposer rapidement d'une architecture de base pour les pages et d'accélérer le processus de design de l'application. La maquette : est accessible [ici](#).

À cette étape, nous avons décidé d'utiliser une bibliothèque de composants React basée sur Tailwind CSS, afin d'avoir un rendu visuel soigné. Après quelques recherches, l'équipe a opté pour FlyonUI. Cependant, cette bibliothèque ne disposait pas de version Figma gratuite, ce qui a considérablement limité la ressemblance entre la maquette et le produit final.

Le choix de cette bibliothèque nous a toutefois permis de bénéficier d'une gestion des thèmes très efficace, avec 8 thèmes disponibles par défaut. Pour représenter ces thèmes dans Figma, chaque couleur utilisée dans les pages est définie comme une variable. Ainsi, il est possible de changer dynamiquement le thème en modifiant simplement ces variables.

Dans une démarche de cohérence artistique, nous avons également choisi de renommer certaines fonctionnalités principales avec des termes en lien avec le nom de l'application. Au lieu de parler d'un « post utilisateur », nous parlons d'un Breeze, et au lieu de dire « publier », nous utilisons le terme souffler (traduction de breath).

Enfin, nous avons désigné deux composants de navigation ainsi que sept pages, dont certaines comportent plusieurs états.

### **Navbar pour les téléphones**

Afin de faciliter la navigation dans l'application pour les utilisateurs mobile, nous avons ajouté, comme dans beaucoup d'application, une barre de navigation en bas de l'écran, facilement accessible pour les utilisateurs smartphone car proche du pouce.

Elle comportera 5 boutons différents pour un accès facile aux 5 pages principales, donc : Accueil (le fil), Recherche, Breeze (publication), notifications et messages.

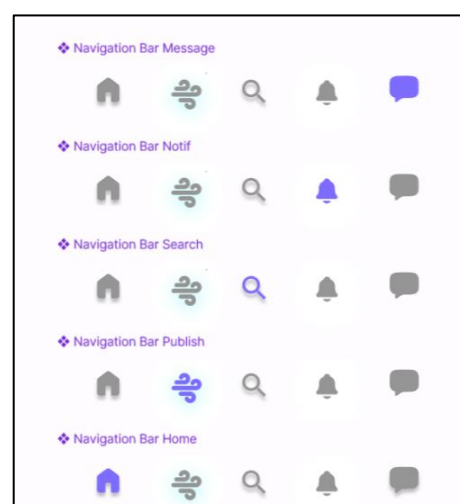


Figure 1 : Design de la base de navigation

## SideBar, 2 versions

La Sidebar, présente sur PC et smartphone est un autre outil de navigation. La Sidebar des mobiles permet essentiellement d'aller sur le profil de l'utilisateur, de changer les thèmes et langues et aussi de se déconnecter. La Sidebar des PC, quant à elle doit posséder les fonctionnalités de la Navbar des smartphones en plus du reste.

Elle permet aussi d'avoir des espaces vacants pour permettre à l'avenir d'avoir de nouvelles pages et fonctionnalités.

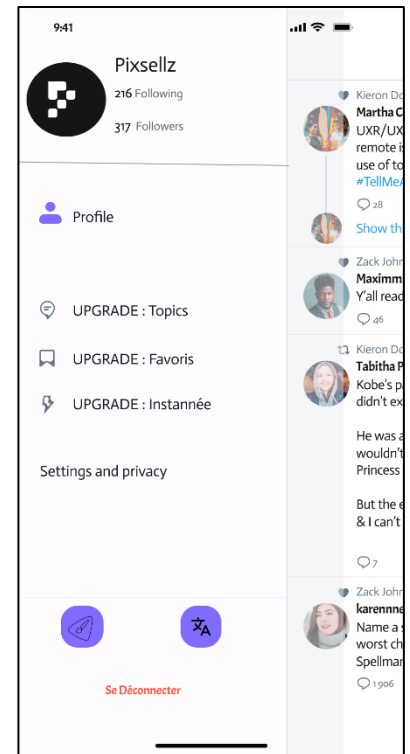


Figure 2 : Sidebar

## Page de Profil

La page Profil sera accessible quand on clique sur le profil. Si l'utilisateur visite son propre profil, le bouton indiquera « Edit Profile » pour modifier son profil, si c'est le profil d'un autre utilisateur, alors le bouton « Edit profil » deviendra « Subscribe » (s'abonner en français) ou « Subscribed » (abonné en français).

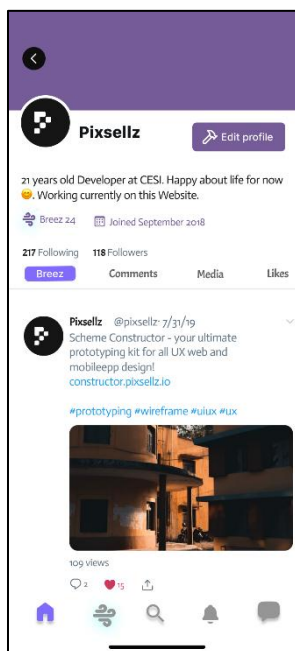


Figure 3 : Page de profil | Breez

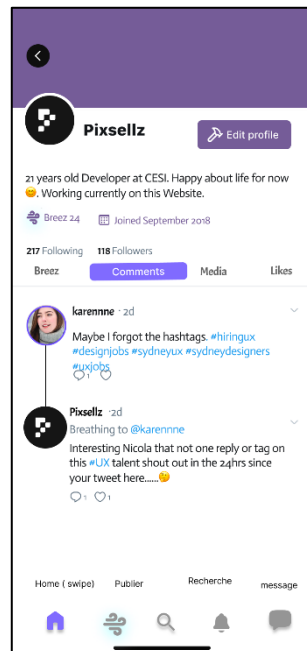


Figure 4 : Page de profil | Commentaires

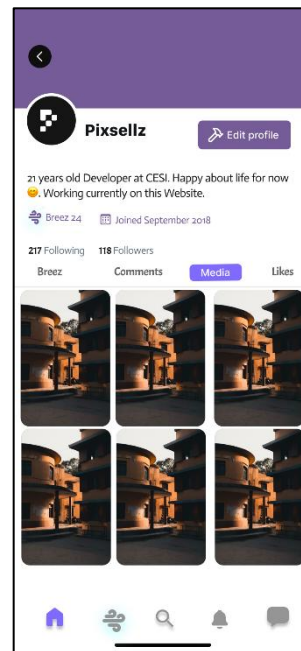


Figure 3 : Page de profil | Media

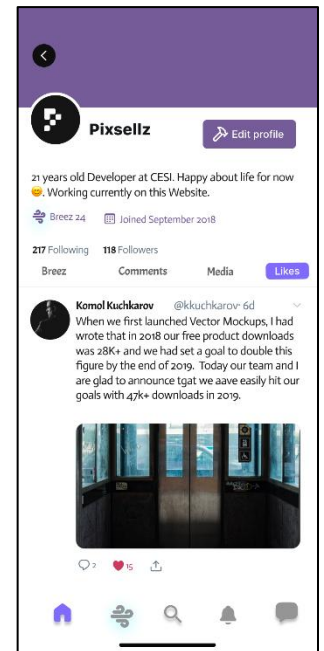


Figure 6 : Page de profil | Like

Les pages de profils affichent les informations importantes des utilisateurs comme, le nombre d'abonnés, de followers, de publications (Breezes), la biographie, la photo de profil, et la date d'inscription.

Additionnellement, on ajoutera des informations supplémentaires : les Breezes publiés, les Breath (commentaires), les images/vidéos, les publications likées.

## Page recherche

Permet de rechercher des utilisateurs et des publications en utilisant des tags et d'afficher les résultats. Il y a une sauvegarde de la dernière recherche qui sera aussi affichée.

Quand la recherche est lancée, un icône de chargement apparaîtra pour faire patienter l'utilisateur.

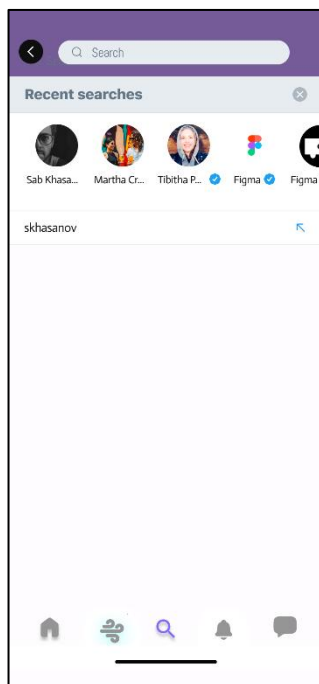


Figure 7 : Page de recherche

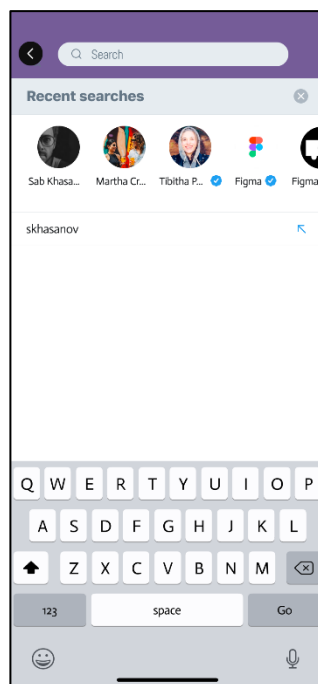


Figure 8 : Page de recherche avec clavier

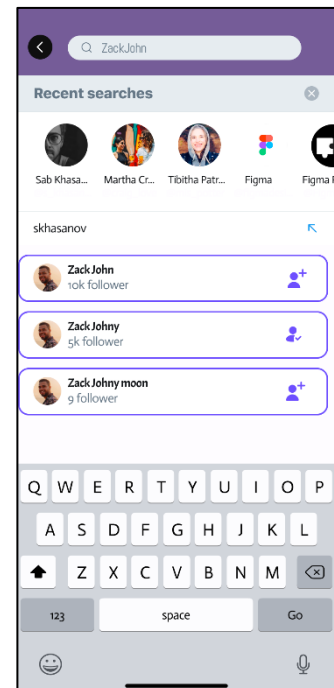


Figure 9 : Page de recherche avec clavier et résultats

## Page d'accueil et composant Breeze (publication)

Affiche un fil d'actualités qui contient les Breezes (publications) des utilisateurs à qui il est abonné, l'utilisateur peut défiler vers le bas et voit ainsi tout son fil d'actualités, donc les publications des personnes à qui il est abonné.

Les Breeze peuvent être cliqués et ouverts dans une page de Breeze qui affichera les commentaires et permettra à l'utilisateur de commenter.

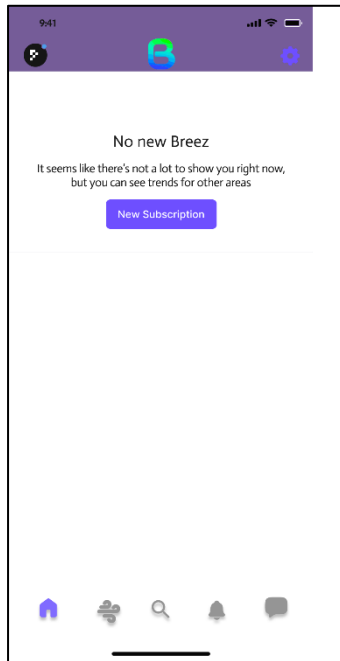


Figure 10 : Accueil vide



Figure 115 : Accueil avec un fil d'actualités

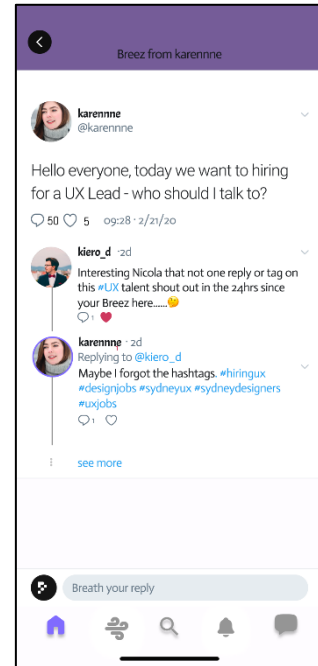


Figure 12 : Détail d'un Breeze

## Page de Publication

Quand un utilisateur souhaite publier un Breeze, il lui suffit d'appuyer sur l'icône de vent dans la Navbar, ensuite, il tombera sur la page pour publier, ici il pourra ajouter des images avec le bouton en bas à gauche, et pour publier il appuiera sur le bouton de vent en bas à droite ou sur le bouton « Breath » (souffler) en haut.

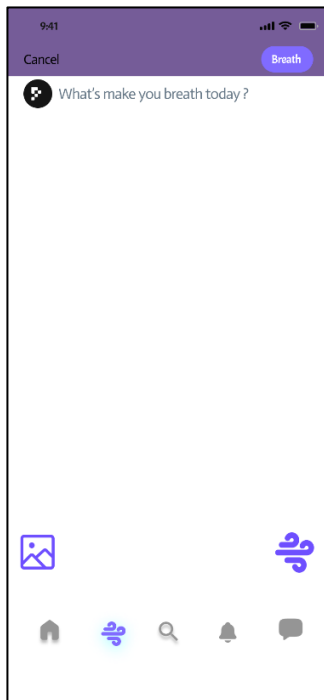


Figure 13 : Page pour poster un Breeze

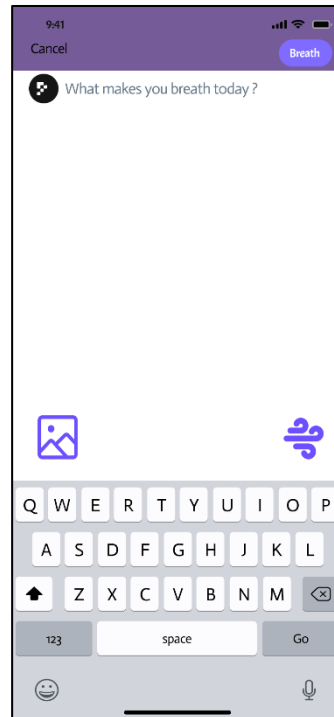


Figure 14 : Page pour ajouter un Breeze avec clavier



## Pages pour les messages

Pour accéder à la liste des messages, l'utilisateur peut appuyer sur le dernier bouton de la navbar, ou dans la sidebar pour accéder à la page qui affichera toutes ses discussions, les nouveaux messages non lus seront accentués et mis en 1<sup>er</sup> dans la liste.

Quand on rentre dans une conversation, on aura accès aux messages envoyés par les 2 utilisateurs, avec la possibilité de supprimer et d'éditer les messages, on peut bien évidemment en envoyer.

En appuyant sur le bouton avec un logo d'enveloppe depuis la page de la liste des messages, on peut créer une conversation avec un ami (réciproquement abonné).

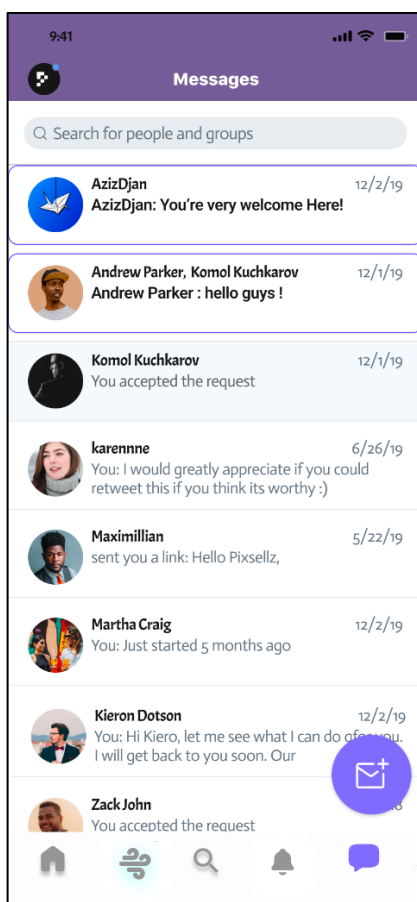


Figure 15 : Liste des messages

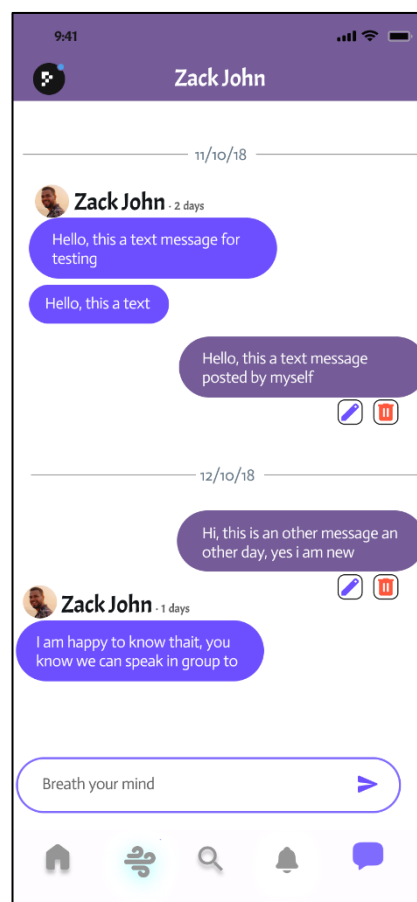


Figure 16 : Conversation entre deux amis

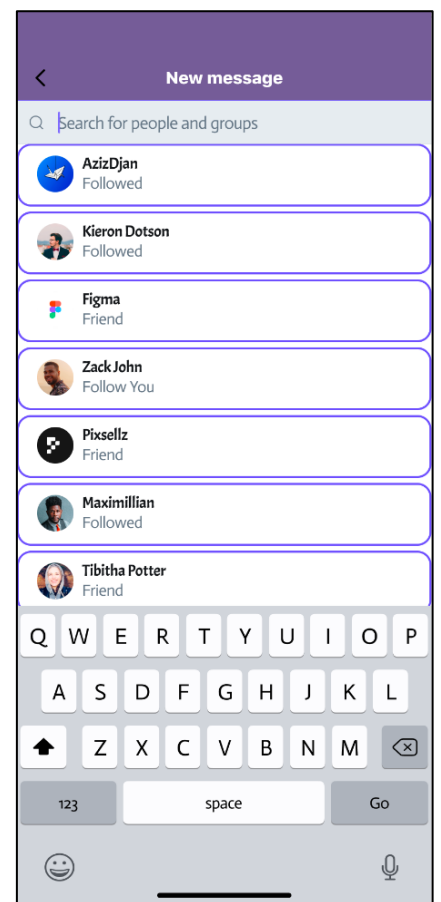


Figure 6 : Page de création d'une nouvelle conversation

## Pages manquantes

Vous remarquerez qu'il manque des pages assez importantes, comme les pages de connexion et de création de compte. Elles ont été créées directement dans le front car la library de composants nous offre des templates déjà prêtes pour ces pages là, donc pas besoin de design.

### b. Architecture logicielle

L'architecture sur laquelle nous sommes partie repose sur une approche micro-services conteneurisée et déployée sur un serveur distant AWS EC2. Chaque service est isolé dans un conteneur Docker distinct et interagit avec les autres services via une API Gateway (Nginx), en suivant une logique de séparation des responsabilités (Single Responsibility Principle). Tous les services fonctionnent dans un réseau Docker privé, permettant une communication inter-conteneurs sécurisée et efficace.

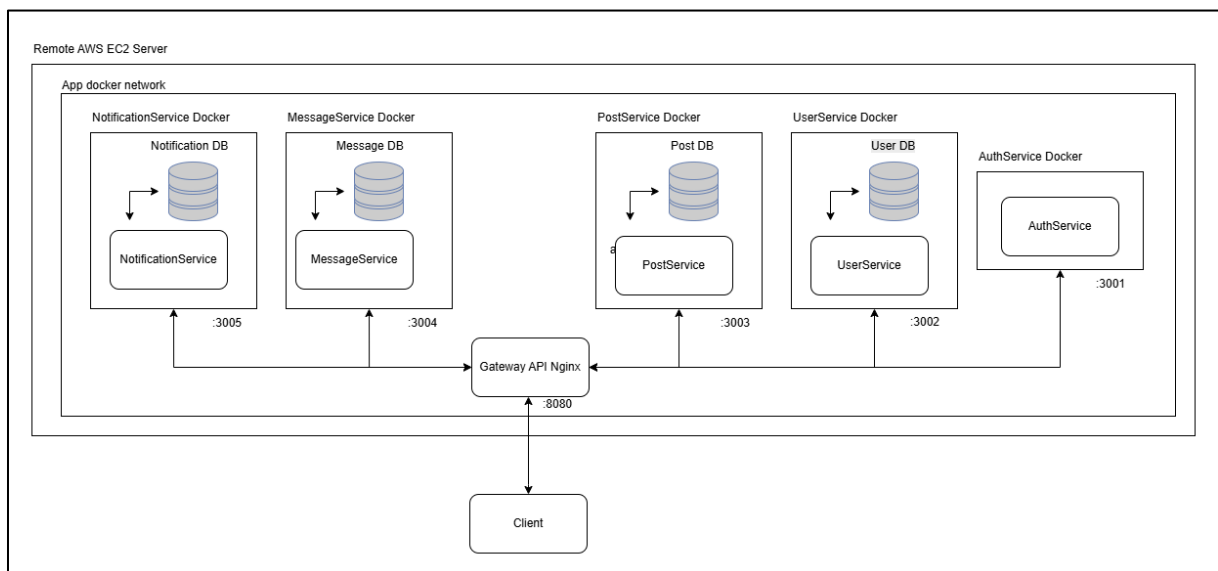


Figure 18 : Architecture logiciel

Explications des différents services :

- Auth service :
  - Port : 3001
  - Gère l'authentification et l'autorisation des utilisateurs.
  - Service central pour sécuriser l'accès aux autres micro-services.
- User service :
  - Port : 3002
  - Gère les informations des utilisateurs, leurs profils et données personnelles.
- Post service :
  - Port : 3003

- Responsable de la gestion des publications créées par les utilisateurs.
- Message service :
  - Port : 3004
  - Prend en charge l'envoi et la réception de messages privés entre utilisateurs.
- Notification service :
  - Port : 3005
  - Gère les notifications en temps réel (ex. : nouveaux messages, likes, etc.).
- Gateway API (Nginx)
  - Port exposé : 8080
  - Sert de point d'entrée unique pour le client.
  - Redirige les requêtes vers les micro-services appropriés en fonction des routes.

L'architecture présentée offre plusieurs avantages significatifs liés à sa conception en micro-services. Tout d'abord, elle permet **une scalabilité indépendante** : chaque service fonctionnant dans son propre conteneur, il est possible de le mettre à l'échelle de manière autonome selon les besoins en ressources ou la charge de trafic spécifique qu'il doit gérer, sans impacter les autres services.

Par ailleurs, **la maintenance** s'en trouve grandement simplifiée. Étant donné que chaque micro-service est découplé des autres, les équipes de développement peuvent modifier, tester et déployer un service individuellement, sans risque de provoquer des effets de bord sur l'ensemble du système. Cela favorise également l'organisation en équipes spécialisées par domaine fonctionnel.

Cette architecture renforce aussi **la résilience** du système global. Une panne ou un dysfonctionnement dans un service (par exemple, le service de notifications) n'empêche pas le bon fonctionnement des autres, comme l'authentification ou la gestion des utilisateurs. Cela contribue à une meilleure tolérance aux pannes et à une disponibilité accrue.

Enfin, **la sécurité** est améliorée grâce à l'intégration d'un service d'authentification centralisé (AuthService), garantissant une gestion cohérente des identités et des autorisations. De plus, l'ensemble des communications client passe par une API Gateway (Nginx), qui centralise les points d'entrée, simplifie la gestion des accès et

permet de mettre en œuvre des politiques de sécurité, de journalisation ou de limitation de débit de manière uniforme sur tous les services.

### 3. Organisation

Dans cette section, nous détaillons l'approche organisationnelle adoptée pour la réalisation efficace de notre projet. Cela inclut les outils utilisés, les méthodes de coordination interne ainsi que la structuration des tâches afin d'assurer une progression fluide et cohérente du travail au sein de l'équipe.

#### a. Méthodes Agiles

Nous nous sommes inspirés des méthodes agiles pour définir notre stratégie d'organisation. Cette approche se traduit notamment par l'utilisation d'un backlog permettant de stocker et de gérer les tâches à effectuer tout au long du projet. Nous avons également mis en place des réunions quotidiennes ou régulières (daily meetings), afin de suivre l'avancement, d'identifier rapidement les éventuels blocages et d'adapter constamment notre travail à l'évolution des besoins.

#### b. GitHub

Pour l'organisation et le suivi de notre projet, nous utilisons GitHub, en particulier la fonctionnalité "Projets" accompagnée d'un système de tickets. Dès le début, nous avons décomposé le sujet initial en plusieurs tâches afin de constituer une première version de notre backlog. Ce backlog a été enrichi progressivement au fil de l'avancée du projet, à mesure que de nouvelles tâches ou des ajustements apparaissaient nécessaires.

Cette méthode nous permet d'attribuer rapidement des tâches aux différents membres de l'équipe, de suivre clairement ce qui reste à réaliser, et de comprendre l'état d'avancement global du projet. Chaque tâche est clairement définie et assignée, facilitant ainsi une répartition efficace du travail.

En parallèle, nous organisons régulièrement, toutes les 24 à 48 heures, des échanges au sein de l'équipe. Ces discussions visent à suivre l'avancement individuel de chaque membre, à identifier rapidement les éventuelles difficultés rencontrées par chacun, et à apporter un soutien si nécessaire. Ces réunions nous servent également à valider des décisions communes, par exemple concernant l'aspect esthétique du frontend, afin d'assurer la cohérence et la qualité globale du projet.

### c. Hiérarchisation des tâches et fonctionnalités

Lors des premières réunions, nous avons commencé par prioriser les fonctionnalités essentielles, nécessaires au bon fonctionnement de l'application :

- Authentification et création de compte (Fx1, Fx2)
- Création et affichage de publications (Fx3, Fx4, Fx5)
- Système de likes et commentaires (Fx6, Fx7, Fx8)

Pour cela nous avons segmenté le projet en de nombreuses sous-tâches et sélectionné les plus importantes pour l'implémentation d'une base de travail. A chaque nouvelle réunion, nous ajoutons les tâches suivantes. Une fois le corps principal implémenté, nous avons ajouté les fonctionnalités secondaires comme le système de notifications (Fx14, Fx15), les messages privés (Fx17), etc.

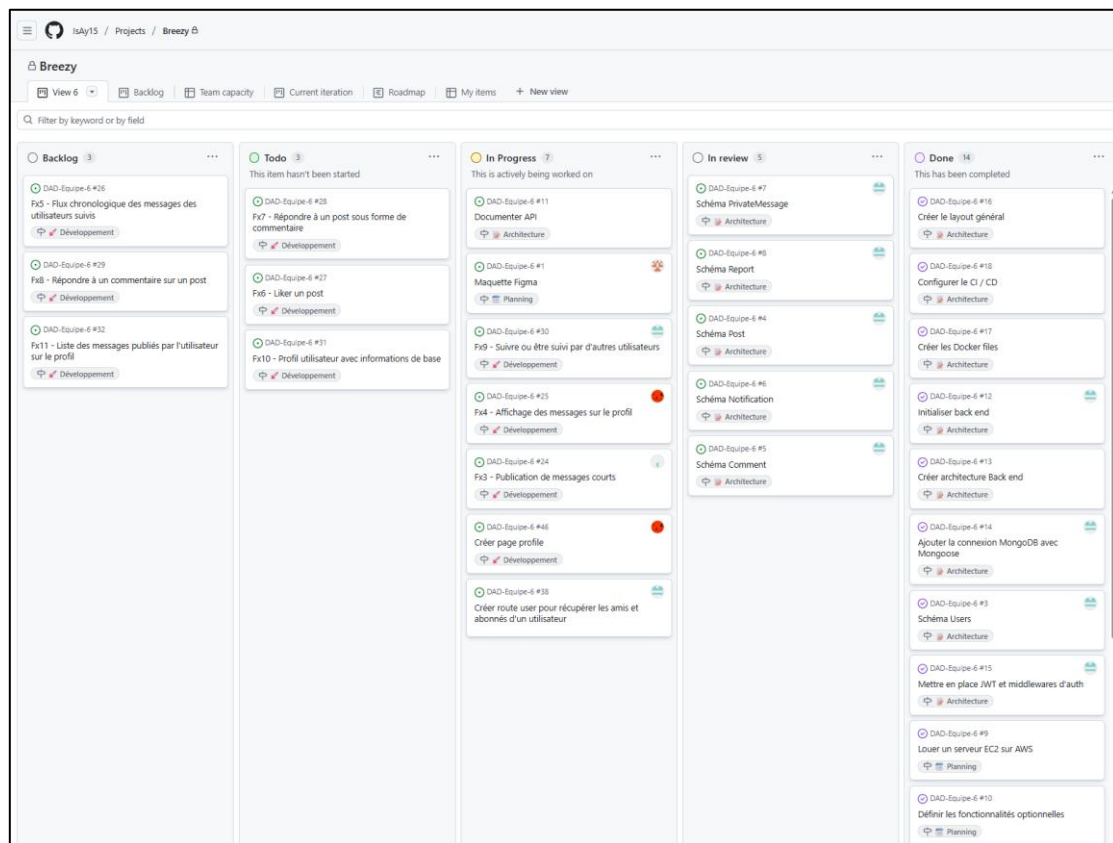


Figure 19 : Backlog du projet

#### d. Répartition du travail

Nous avons choisi de répartir le travail au sein de notre équipe en créant des groupes spécifiques pour le backend et le frontend. Ce choix a été fait selon les préférences des membres de notre groupe. En effet, grâce aux différents workshops réalisés en cours, chacun d'entre nous a pu expérimenter diverses positions possibles (frontend, backend, sécurité, etc.), auxquelles s'ajoutent les expériences personnelles de chacun. Cependant, afin de maintenir une compréhension globale du projet, nous partageons régulièrement nos tâches et méthodologies lors des réunions quotidiennes. Il est arrivé occasionnellement que certains membres changent de domaine pour diversifier leurs compétences. Actuellement, notre organisation comprend une personne en charge en charge des maquettes, une personne dédiée au développement frontend, une autre responsable du raccordement frontend-backend, ainsi que deux personnes travaillant sur les micro-services et la conteneurisation côté backend.

#### 4. Étapes et ressources mobilisées

Le développement de Breezy s'est déroulé en plusieurs phases successives, chacune ayant mobilisé des ressources et outils spécifiques. Cette organisation a permis une montée en complexité progressive du projet.

- Phase 1 – Préparation et idéation

Cette phase a été consacrée à la définition des besoins fonctionnels et techniques de l'application.

- Rédaction du backlog initial à partir des user stories fournies.
- Élaboration d'une maquette Figma représentant les écrans principaux (page d'accueil, profil utilisateur, formulaire d'inscription, etc.).
- Discussions en groupe sur les technologies à employer (React, Node.js, MongoDB, Docker, etc.).

Outils utilisés : Notion, Figma, Discord, GitHub Projects.

- Phase 2 – Mise en place du backend

L'objectif principal était de créer une base fonctionnelle pour l'authentification et les services fondamentaux.

- Rédaction du backlog initial à partir des user stories fournies.

- Élaboration d'une maquette Figma représentant les écrans principaux (page d'accueil, profil utilisateur, formulaire d'inscription, etc.).
- Discussions en groupe sur les technologies à employer (React, Node.js, MongoDB, Docker, etc.).

Outils utilisés : VS Code, MongoDB Compass, Docker, Postman, Git.

- Phase 3 – Développement du frontend

Parallèlement à l'API, le frontend a été développé avec une approche mobile-first.

- Construction de l'interface avec React + Next.js et stylisation avec Tailwind CSS.
- Gestion de l'état local, des appels API, et des formulaires (login, inscription, création de poste).
- Intégration des jetons JWT et redirections post-authentification.

Outils utilisés : React, Tailwind, Axios, VS Code, Notyf.

- Phase 4 – Intégration, tests et déploiement

C'est la dernière phase avant rendu :

- Tests des interactions entre services (Auth, Post, User, Notification...).
- Corrections de bugs liés aux appels API et à la sécurité (ex : CORS, vérification JWT). (Page d'accueil, profil utilisateur, formulaire d'inscription, etc.).
- Déploiement sur un serveur AWS EC2, configuration du docker-compose, et mise en production.
- Ajustements UI/UX en fonction des retours.

Outils utilisés : AWS EC2, Docker Compose, GitHub Actions (si utilisé).

## 5. Rendu

### a. Features validées

- Création de compte (Fx1) : formulaire avec validation, appel API vers auth-service

L'inscription se fait via un formulaire React avec des contrôles immédiats côté frontend pour garantir une bonne expérience utilisateur :

Le nom d'utilisateur doit contenir au moins 3 caractères. La saisie est surveillée dynamiquement et le champ est invalidé si la contrainte n'est pas respectée.

Le mot de passe doit avoir au minimum 6 caractères et contenir au moins une lettre minuscule (vérifié par une regex côté client).

En parallèle, une requête debounced est envoyée automatiquement pour vérifier l'unicité du nom d'utilisateur via le auth-service, qui renvoie un message d'erreur si le pseudo est déjà pris.

Une fois tous les champs validés, le formulaire envoie une requête POST au auth-service.

Le backend refait les contrôles pour des raisons de sécurité (vérifications serveur obligatoires) :

- Unicité du pseudo et du courriel en base MongoDB,
- Validation du mot de passe via une regex (taille et minuscule),
- Hachage du mot de passe avec bcrypt avant stockage.

Après création du compte, le backend génère un access token et un refresh token, tous deux envoyés dans des cookies sécurisés.

Cette architecture permet à l'utilisateur d'être connecté automatiquement après inscription, tout en garantissant sécurité et contrôle serveur.



Breezy

Inscription

Créez un compte pour commencer

Nom d'utilisateur\*

Entrez votre nom d'utilisateur

Email\*

Entrez votre email

Mot de passe

Niveau de sécurité du mot de passe : Vide

Votre mot de passe doit contenir :

- Au moins 6 caractères
- Au moins une lettre minuscule

Recommandations pour un mot de passe fort :

- Au moins une lettre majuscule
- Au moins un chiffre
- Au moins un caractère spécial

Confirmez le mot de passe\*

☐ J'accepte la [politique de confidentialité](#)

S'inscrire

Vous avez déjà un compte ?

Connexion

ou


 S'inscrire avec Google

Figure 20 : Page d'inscription

Breezy

Connexion

Connectez-vous pour accéder à votre compte

Nom d'utilisateur ou email

Entrez votre nom d'utilisateur ou email

Mot de passe

☐ Se souvenir de moi

[Mot de passe oublié ?](#)

Se connecter

Vous n'avez pas de compte ?

Créer un compte

ou


 Se connecter avec Google

Figure 21 : Page de connexion

- Fx2 – Authentification sécurisée

L'authentification repose sur une stratégie JWT avec refresh token, assurant à la fois sécurité et confort utilisateur.

Lors de la connexion, le auth-service génère deux tokens :

Un access token (durée de vie de 10 minutes)

Un refresh token (durée de vie de 7 jours, usage unique)

Les deux sont envoyés dans des HttpOnly cookies, ce qui les protège des accès JavaScript côté client (contre les attaques XSS). Le refresh token est également stocké dans la base de données associée à l'utilisateur.

Lorsqu'un access token expire, le frontend tente automatiquement de le régénérer via le refresh token. Si ce dernier est valide et présent en base, un nouveau couple de tokens est renvoyé. Le refresh token utilisé est alors retiré de la base et remplacé par un nouveau (usage unique).

Lors d'une déconnexion, le refresh token est supprimé de la base, ce qui empêche toute réutilisation ultérieure, même s'il était encore valide. Cette approche garantit à la fois sécurité, durée de session longue et révocation possible côté serveur.

- Fx22 – Interface multi-langues

L'interface multilingue a été implémentée à l'aide du module next-intl, une solution dédiée à la gestion des langues dans les applications Next.js. Ce module permet de centraliser les traductions dans des fichiers JSON, placés dans le dossier /messages à la racine du projet. Pour déterminer la langue à afficher, une valeur est stockée dans un cookie, généralement défini lors de la sélection de la langue par l'utilisateur. Ensuite, le Provider de next-intl récupère cette valeur pour charger dynamiquement les contenus traduits à partir du fichier approprié (par exemple fr.json pour le français), assurant ainsi une internationalisation cohérente et contextuelle de l'interface.

- Fx23 – Thème personnalisé

Le changement de thème permet aux utilisateurs de personnaliser l'apparence de l'application (clair, sombre, etc.). Cette fonctionnalité est rendue possible grâce à un système basé sur des attributs HTML et un cookie. Lorsqu'un utilisateur choisit un thème, son choix est enregistré dans un cookie (nommé BREEZY\_THEME). Au prochain chargement de la page, l'application lit ce cookie et applique automatiquement le bon thème à l'interface, sans que l'utilisateur ait à le reconfigurer. L'effet visuel est immédiat : les couleurs, arrière-plans et éléments graphiques s'adaptent selon le thème choisi. L'interface de sélection

des thèmes propose plusieurs styles visuels et un mode “système” qui s’aligne sur les préférences du système d’exploitation. Ce système rend l’expérience plus agréable, personnalisable et cohérente à travers toutes les pages.

## b. Présentation générale

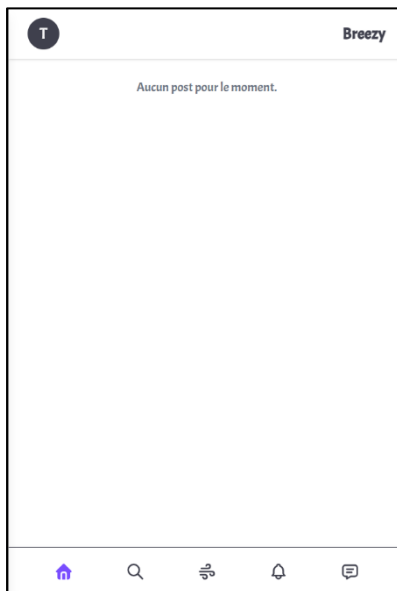


Figure 22 : Page de première venue

La page ci-contre est la première page que voit un utilisateur quand il arrive sur l’application. Elle est composée :

- D’une zone supérieure lui permettant d’ouvrir le menu quand il clique sur l’avatar de son profil.
- D’une zone centrale où s’afficheront les posts du feed de l’utilisateur, un exemple avec quelques posts est présentée plus tard.
- D’une zone inférieure contenant tous les onglets accessibles : de la zone de recherche, la publication de posts, jusqu’au messages privés.

Cette page ci montre le menu qui permet aux utilisateurs de :

- Consulter leur profil.
- Modifier la langue (parmi 5).
- Modifier le thème de l’application.
- Suivre leur nombre d’abonné/ d’abonnement

De plus, même si ce n’est pas disponible pour l’instant, nous avons prévu d’ajouter un page favori qui permettra de suivre tous les posts que l’utilisateur met en favoris pour ne pas oublier des posts

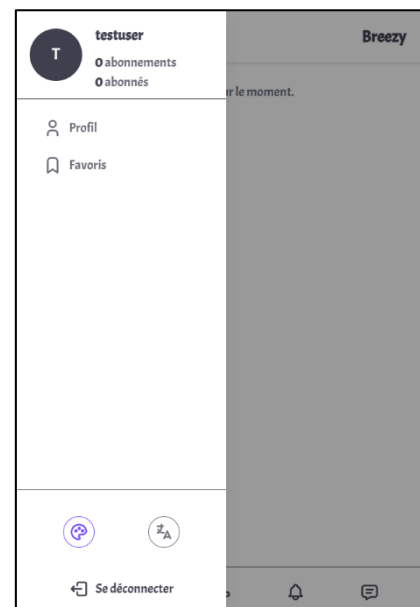


Figure 23 : Sidebar

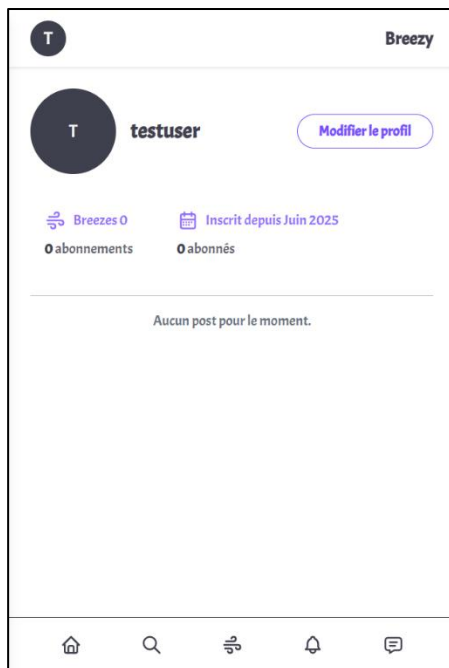


Figure 24 : Page de profil

La page de modification du profil permet aux utilisateurs de personnaliser leur avatar en ayant l'occasion d'importer une photo de profil.

De plus elle permet aussi de modifier la bio de l'utilisateur pour par exemple partager ses activités du moment. Une zone est aussi disponible en dessous de son profil pour retrouver l'historique des tous les posts publiés par l'utilisateur.

Il retrouve aussi des informations comme son nombre d'abonnement/d'abonné ainsi que sa date d'inscription sur l'application.

La page de recherche permet à un utilisateur de rechercher d'autres comptes utilisateurs auxquels il pourra s'abonner. Elle permet aussi de rechercher tous posts pourvu d'un tag spécifique en commençant sa recherche par un dièse (#).

Pour finir, cette page garde en mémoire un historique des recherches, améliorant ainsi avantageusement l'expérience utilisateur.

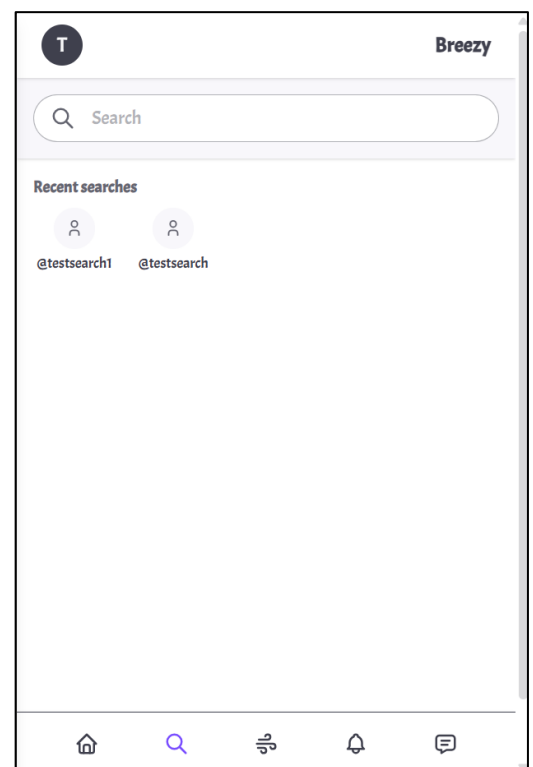


Figure 25 : Page de recherche

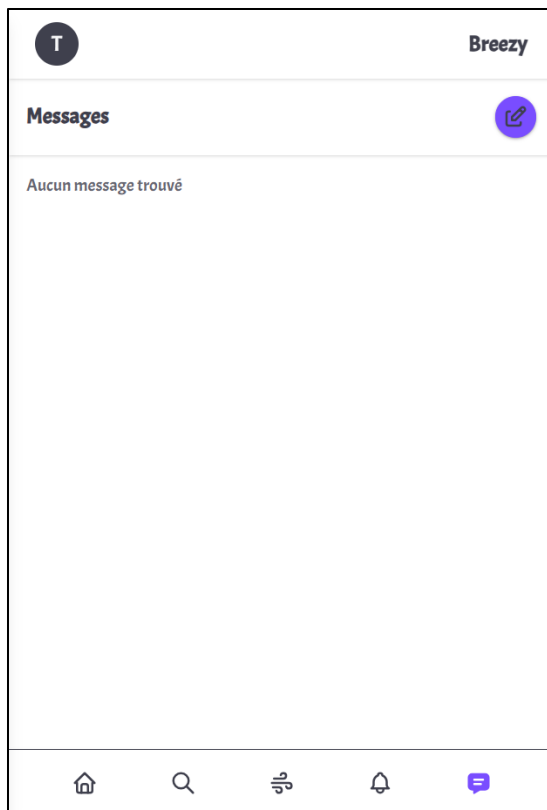


Figure 26 : Page des messages

Cette page affiche toutes les conversations privées de l'utilisateur. Dans l'exemple, aucun message n'a encore été échangé, d'où le message « *Aucun message trouvé* ». En haut à droite, un bouton flottant en forme de stylo permet de démarrer une nouvelle conversation. Cette interface offre une expérience simple pour gérer ses messages, avec un accès direct depuis l'onglet de messagerie (icône en bas à droite de la navbar).

Cette interface permet à l'utilisateur de rédiger un *Breeze*, c'est-à-dire une publication courte. Elle contient :

- Un champ de texte où l'utilisateur peut écrire son message.
- Un bouton **Cancel** pour annuler la publication.
- Un bouton **Breath** (souffler), visuellement associé à une icône de vent, pour publier le message.
- Une icône d'ajout d'image, située en bas à gauche, permettant d'intégrer un média à la publication.

L'interface reste fidèle au design global de l'application : minimaliste, intuitive, et optimisée pour l'usage mobile avec une barre de navigation en bas de l'écran.



Figure 27 : Page de création de Breeze

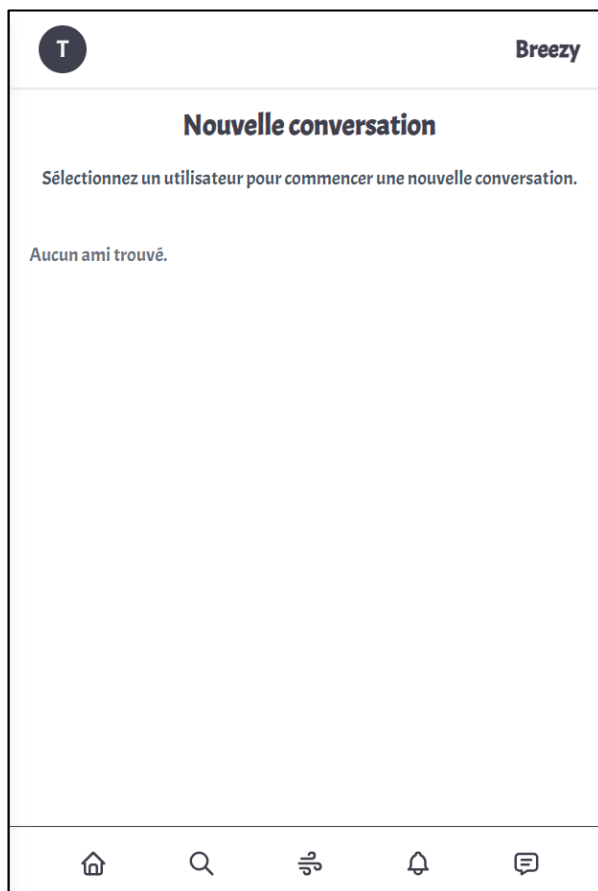


Figure 28 : Page d'affichage des conversations

Cette page correspond au fil d'actualité de l'utilisateur.

Elle affiche les publications (appelées *Breezes*) des personnes auxquelles l'utilisateur est abonné. Chaque publication contient :

- Le nom et l'avatar de l'auteur,
- La date de publication,
- Le contenu du Breeze,
- Le nombre de likes.

L'utilisateur peut parcourir les Breezes en défilant vers le bas. Dans cet exemple, on voit trois publications de l'utilisateur *theooc*. La barre de navigation inférieure, identique à celle de la page précédente, permet de naviguer dans l'application.

Cette page permet à l'utilisateur de démarrer une nouvelle conversation privée avec un autre membre. Elle affiche un message indiquant qu'il faut sélectionner un utilisateur pour commencer.

Dans l'exemple visible, aucun ami n'est disponible pour initier une discussion, ce qui est signalé par le message « *Aucun ami trouvé.* ».

L'interface reste claire, avec un design épuré centré sur l'information principale. Une barre de navigation est visible en bas de l'écran, permettant un accès rapide aux pages principales de l'application (Accueil, Recherche, Publication, Notifications, Messages).

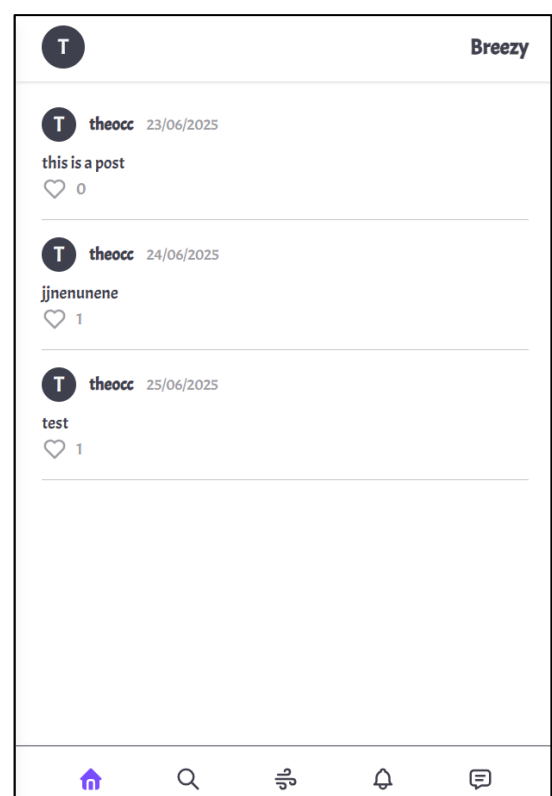


Figure 29: Page du fil d'actualité

## 6. Axes d'améliorations

- **Messagerie et notifications instantanées (Websockets)**
  - Intégrer les Websockets permettrait d'assurer une communication en temps réel pour les messages privés et les notifications, améliorant ainsi l'interactivité de l'application.
- **Optimiser les ressources (docker-compose lent)**
  - Une optimisation du fichier docker-compose.yml et de la configuration des services permettrait de réduire les temps de démarrage et la consommation de ressources, améliorant ainsi le confort de développement.
- **Avoir plus de détails sur les profils**
  - Enrichir les profils utilisateurs avec des informations supplémentaires (bio, localisation, date d'inscription...) offrirait une meilleure contextualisation sociale et favoriserait les interactions.
- **Ajouter un @ aux profils et notifier les mentions**
  - L'ajout d'un système de mention avec le symbole @ et des notifications associées permettrait de dynamiser les conversations entre utilisateurs et de renforcer leur engagement.
- **Ajouter des fonctionnalités de modération/administration**
  - Développer une interface de modération avec des droits spécifiques pour les administrateurs renforcerait la sécurité de la plateforme et le respect des règles communautaires.
- **Signalement de contenu**
  - Permettre aux utilisateurs de signaler des contenus inappropriés offrirait un premier niveau de modération participative, indispensable à toute plateforme sociale ouverte.
- **Ajout de photo/vidéo sur les posts**
  - Intégrer le support des médias enrichirait les publications, diversifierait les contenus partagés et rendrait la plateforme plus attrayante visuellement.
- **Notifier les likes**
  - Ajouter des notifications pour les mentions « J'aime » renforcerait l'engagement des utilisateurs en valorisant l'interaction et la reconnaissance sociale.
- **Suspendre et bannir des comptes**
  - La possibilité pour les administrateurs de suspendre ou bannir des comptes serait essentielle pour garantir le respect des conditions d'utilisation et préserver un environnement sain.

## 7. Conclusion

Le projet Breezy nous a permis de mettre en pratique un large éventail de compétences techniques et organisationnelles. En partant d'une simple idée de réseau social léger, nous avons réussi à concevoir, structurer et développer une application fonctionnelle, fondée sur une architecture moderne en micro-services. L'utilisation de technologies actuelles comme React, Node.js, Docker ou MongoDB, combinée au déploiement sur un serveur distant, nous a confrontés à des défis concrets proches de ceux rencontrés dans un environnement professionnel.

Au-delà de l'aspect technique, ce projet nous a appris à mieux travailler en équipe, à gérer nos priorités et à adapter notre organisation grâce à des méthodes agiles. La communication au sein du groupe et la répartition des rôles ont joué un rôle clé dans l'avancement du projet.

Même si certaines fonctionnalités restent perfectibles ou à compléter, le résultat final est déjà un socle solide. Breezy répond aux objectifs initiaux, tout en ouvrant la voie à de nombreuses pistes d'amélioration pour l'avenir.

Ce projet a été une expérience formatrice, tant sur le plan technique que collaboratif, et nous sommes fiers du travail accompli en équipe.