Micro Services

Sommaire

01. Résumé

02. Organisation

03. Architecture

04. Fonctionnalités et attentes du sujet

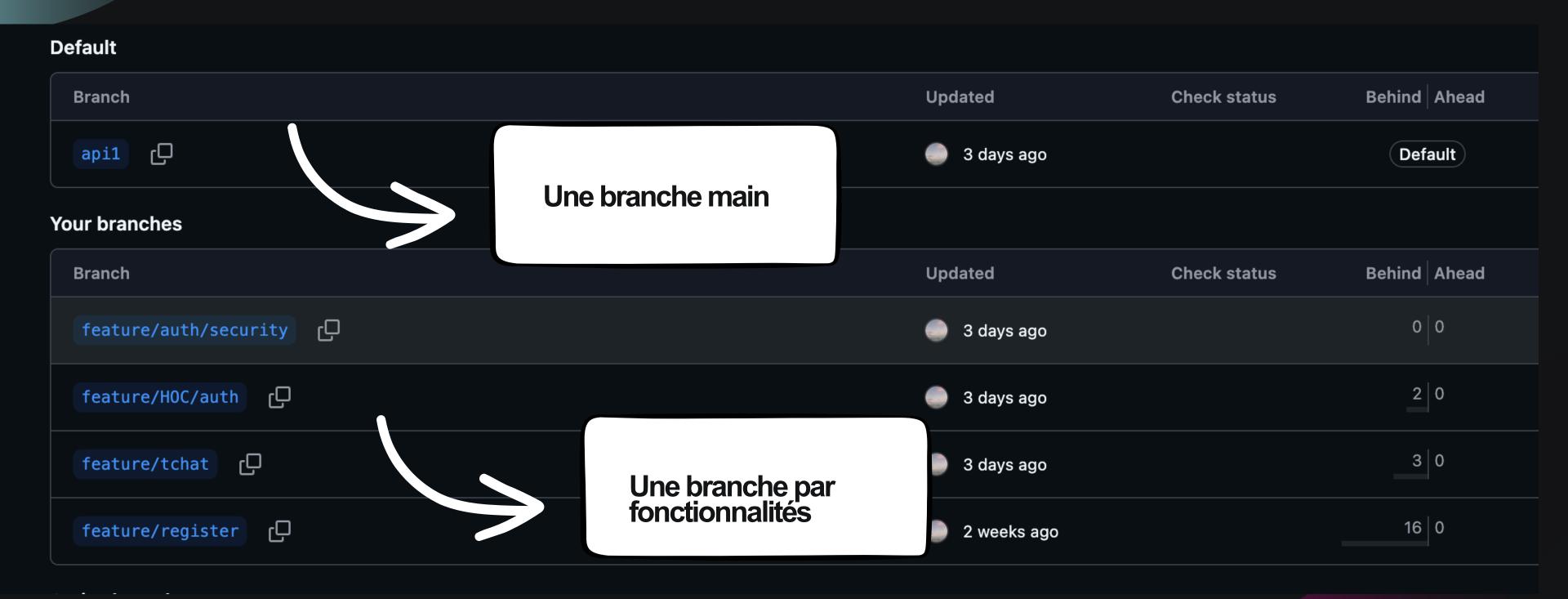
05. Bonus

O'I Résumé

Conception et développement de deux micro-services, Auth et Tchat, intégrés via un Connecteur pour assurer leur interconnexion et leur communication.

02 Organisation

Gitflow



Maquette

Discussion Accueil Micro-services Pseudo Accueil, user connecté Profil Micro-services Mot de passe

03 Architecture

```
micro-services/
api-gateway/
frontend/
services/
    auth-service/
   tchat-service/
utils/
docker-compose.yml
infrastructure/
shared/
```

04 Fonctionnalités et attentes du sujet

Service d'authentification

- Inscription, connexion et gestion des sessions des utilisateurs.
- Utilisation de tokens JWT généré à la connexion pour la validation des requêtes.
- Gestion des informations des utilisateurs (pseudo, email, etc.).
- Utilisation d'une base de données MySQL pour stocker les utilisateurs.

Service de tchat

• Création et gestion des discussions.

• Récupération, stockage et manipulation des messages.

 Utilisation d'une base de données NoSQL pour stocker les discussions et les messages.

Connecteur

• Centralisation de toutes les requêtes HTTP.

 Validation des tokens JWT et redirection des requêtes vers les services appropriés.

• Gestion des erreurs HTTP 401 pour les tokens invalides ou absents.

Frontend

- Interface utilisateur en React avec Vite, et Tailwind.
- Interaction avec le connecteur pour envoyer des requêtes aux différents microservices.
- Notifications (toasts) pour les alertes utilisateurs.
- Messagerie graphique.

Fonctionnalités et attentes du sujet

Fonctionnalités du sujet	Statut
Architecture Micro Services	
Mise en place d'une architecture de microservices	▽
Auth-Service (PHP + Slim + MySQL)	
Création de l'authentification avec JWT	$\overline{m{arphi}}$
Utilisation de Slim pour gérer l'authentification	$\overline{\checkmark}$
CRUD Utilisateur	×
Utilisation de MySQL pour gérer les utilisateurs	▼
Tchat-Service (Node.js + Express + MongoDB)	
Création du service de messagerie	▽
Utilisation de MongoDB pour le stockage des messages	▼
CRUD Messagerie	×
Mise en place de Node.js, Express	
Frontend (React.js, Vite et Tailwind)	
Envoi des requêtes au Connecteur	▼
Installation de react tostity pour gérer les toasters	
Bonus	
Client React	▽
Messagerie graphique	

05 Bonus

• Messagerie graphique.

• Client React.

Merci