

Services Angular et communication avec l'API

Vue d'ensemble

Le frontend ne doit pas appeler le backend directement depuis chaque composant. La logique partagée est regroupée dans `uniksimFrontEnd/src/app/service/`, avec plusieurs familles spécialisées.

Flux standard

```
sequenceDiagram
    participant P as Page / Component
    participant S as Service Angular
    participant I as AuthInterceptor
    participant C as Controller NestJS
    participant B as Service backend

    P->>S: Action utilisateur
    S->>I: Requête HttpClient
    I->>I: Ajoute / gère auth
    I->>C: HTTP
    C->>B: Logique métier
    B-->>C: Résultat
    C-->>S: Réponse
    S-->>P: État / données UI
```

Auth

`service/auth/` contient :

- `AuthGuard.service.ts` : guards de navigation selon le type d'utilisateur.
- `authInterceptor.service.ts` : intercepteur HTTP enregistré globalement dans `app.module.ts`.

Accès aux données et APIs

`service/requestAndData/` contient les services historiques de communication et de données :

- `braincloudRequest.service.ts`
- `dataBaseRequest.service.ts`
- `dataSource.service.ts`

Avant de créer un nouveau service HTTP générique, vérifier si l'appel appartient déjà à l'un de ces services ou à un service fonctionnel plus spécifique.

Support

`service/support/` sépare le support en trois responsabilités :

- `support.service.ts` : appels et logique du support.
- `support-socket.service.ts` : namespace `/support`, rooms de conversations et événements temps réel.
- `support-state.service.ts` : état frontend partagé du support.

Notifications

`service/notifications/` contient notamment :

- `user-notification.service.ts` : inbox locale, synchronisation HTTP et fusion par UID des notifications transmises par AppComponent.
- `notification-sound.service.ts` : gestion du son lors de nouvelles notifications.

Présence et WebPanel

- `service/presence/admin-presence.service.ts` : données de présence utilisées par l'interface admin.
- `service/webPanel/web-panel.service.ts` : logique du WebPanel.

Choisir où placer une nouvelle logique

flowchart TD

```

NEW[Nouvelle logique frontend] --> SHARED{Utilisée par plusieurs pages ?}
SHARED -- Non --> PAGE[La garder près de la Page / Component]
SHARED -- Oui --> DOMAIN{Sous-système existant ?}
DOMAIN -- Auth --> AUTH[service/auth]
DOMAIN -- Support --> SUP[service/support]
DOMAIN -- Notifications --> NOTIF[service/notifications]
DOMAIN -- Présence --> PRES[service/presence]
DOMAIN -- Données / API générique --> DATA[service/requestAndData]
DOMAIN -- Autre --> NEWSVC[Nouveau service spécialisé]

```

Points d'attention

- Éviter de dupliquer les appels HTTP dans plusieurs pages.
- Garder les composants centrés sur l'UI et déléguer la logique partagée aux services.
- Pour le temps réel, toujours prévoir le chargement initial HTTP et la mise à jour Socket.IO.
- Lorsqu'un endpoint change, vérifier le service Angular correspondant et non seulement la page visible.

Temps réel du compte et transport partagé — septembre 2026

`service/realtime/realtime-socket.ts` centralise connexion JWT, événements observables, reconnexion et acknowledgements. `service/account/account-socket.service.ts` utilise `/account` pour présence, télémétrie consentie, jeu actif et `notification.created`. `service/support/support-socket.service.ts` utilise `/support` pour messages, tickets et typing, et restaure la conversation active à la reconnexion. Les composants de support et de présence admin consomment le service

du compte pour les signaux de présence.

`AccountSocketService` joue le son et émet la notification; `AppComponent` la transmet à `UserNotificationService.upsert()`. Les modèles sont séparés dans `Models/Notification.ts`, `Presence.ts` et `Support.ts`; les constantes communes vivent dans `realtime.constants.ts` et `account.constants.ts`. Voir [Support temps réel](#).

Revision #2

Created 2026-09-18 17:45:55 UTC by Mathis

Updated 2026-09-30 14:22:51 UTC by Mathis