

Conventions de sécurité Web

But

Cette page résume les mécanismes de sécurité réellement présents dans le projet. Elle ne remplace pas un audit de sécurité; elle sert de guide pour éviter de contourner les protections existantes lorsqu'on ajoute une feature.

Authentification et autorisation

Le backend utilise des JWT et plusieurs guards spécialisés :

- `AuthGuardUser` : utilisateur authentifié;
- `AuthGuardAdmin` : rôle admin;
- `AuthGuardAdminUser` : admin ou admin_user;
- `AuthGuardGame` : accès utilisé par les appels du jeu.

Les tokens peuvent être lus depuis l'en-tête `Authorization: Bearer ...` ou, selon le flux, depuis le cookie `token`. Les guards backend restent la source d'autorité; les guards Angular servent surtout à contrôler la navigation.

Flux d'autorisation

sequenceDiagram

participant C as Client

participant G as Guard NestJS

participant J as JwtService

participant E as Endpoint

```
C->>G: Requête + token
G->>J: Vérifie signature / expiration
J-->>G: Payload
G->>G: Vérifie rôle si nécessaire
alt Autorisé
    G->>E: Continue
    E-->>C: Réponse
else Refusé
    G-->>C: 401 Unauthorized
end
```

Configuration HTTP

Dans `src/main.ts` :

- CORS est configuré avec credentials;
- les requêtes JSON et URL-encoded ont une limite de taille;
- l'en-tête HSTS `Strict-Transport-Security` est ajouté;
- le serveur principal écoute sur le port `3000`.

Uploads du support

`SupportAttachmentService` applique plusieurs contrôles avant de stocker ou servir une image :

- validation du contenu binaire réel plutôt que confiance dans le MIME envoyé par le navigateur;
- types d'image autorisés limités;
- normalisation en WebP;
- limite de dimensions et de taille;
- nom de fichier aléatoire et normalisé;
- protection contre les chemins arbitraires via `path.basename` et validation du format;
- vérification que la pièce jointe appartient à la conversation/utilisateur demandé;
- quotas de stockage utilisateur et global;
- nettoyage des uploads orphelins.

Sécurité des fichiers

flowchart TD

U[Upload] --> BIN[Valider signature binaire]

BIN --> SIZE[Vérifier dimensions / taille]

SIZE --> NORMAL[Normaliser en WebP]

NORMAL --> NAME[Générer nom aléatoire]

NAME --> QUOTA[Vérifier quotas]

QUOTA --> SAVE[Écrire avec création exclusive]

READ[Lecture] --> PATH[Vérifier nom / chemin]

PATH --> OWNER[Vérifier conversation + utilisateur]

OWNER --> FILE[Servir le fichier]

Secrets et configuration

Le projet contient encore des secrets historiques dans la configuration. Ils ne doivent **jamais** être copiés dans le wiki, le frontend ou les logs. Toute nouvelle intégration devrait préférer les fichiers d'environnement ou une gestion de secrets dédiée plutôt qu'ajouter une nouvelle clé en clair dans le code.

IA et contrôle d'usage

Le système IA possède des limites de requêtes, budgets, timeouts et plafonds de tokens configurés dans `AI_CONFIG`. Le dossier `src/node.js/server/ai/guard/` contient également la logique de garde liée aux coûts.

Checklist pour une nouvelle route

- Choisir le guard backend correspondant au niveau d'accès.
- Ne jamais se fier uniquement à l'UI ou au routing Angular.
- Valider les IDs liés à l'utilisateur côté serveur.
- Limiter et valider les entrées volumineuses ou fichiers.
- Ne pas exposer de secret dans une réponse API ou dans Angular.
- Préférer une erreur contrôlée à une exception brute contenant des détails internes.
- Vérifier les permissions avant de lire un fichier ou une ressource appartenant à un autre utilisateur.

Revision #1

Created 2026-09-18 18:06:10 UTC by Mathis

Updated 2026-09-18 18:06:16 UTC by Mathis