Backend pour le suivi des élèves - Step13

Dans cette étape, nous souhaitons mettre en place la gestion des rôles.

Il y a 2 rôles :

- Un admin peut tout faire !
- Un enseignant peut :
    - créer des commentaires
    - modifier SES commentaires
    - supprimer SES commentaires

Bouncer

AdonisJS v6 fournit un système de gestion d'autorisations appelé `Bouncer`, qui permet d'encapsuler proprement la logique métier liée aux permissions dans des fichiers dédiés.

Pourquoi utiliser Bouncer ?

Avec `Bouncer`, toute la logique "est-ce que cet utilisateur peut faire X sur Y" peut être déplacé hors du contrôleur, ce qui rend notre code:

- plus lisible,
- plus modulaire
- plus facile à tester
- plus facile à réutiliser

Installation et configuration de bouncer

Installation :

```
npm install @adonisjs/bouncer
```

Configuration :

```
node ace configure @adonisjs/bouncer
```

Création de la permission

Sans suprise par le CLI :

```
node ace make:policy Comment
```

Implémentons maintenant les permissions dans le fichier `app/policies/comment_policy.ts` :

```
import User from '#models/user'
import Comment from '#models/comment'
import { BasePolicy } from '@adonisjs/bouncer'
import Teacher from '#models/teacher'

export default class CommentPolicy extends BasePolicy {
  private async isOwner(user: User, comment: Comment): Promise<boolean> {
    const teacher = await Teacher.query()
      .where('id', comment.teacherId)
      .where('userId', user.id)
      .select('id') // on réduit la charge
      .first()
    return !!teacher
  }

  // Peut mettre à jour un commentaire
  async update(user: User, comment: Comment) {
    return user.role === 'admin' || this.isOwner(user, comment)
  }

  // Peut supprimer un commentaire
  async delete(user: User, comment: Comment) {
    return user.role === 'admin' || this.isOwner(user, comment)
  }

  // Peut créer un commentaire (par défaut : tous les enseignants)
  async create(user: User) {
    return user.role === 'teacher' || user.role === 'admin'
  }

  // Peut voir un commentaire (par défaut : tous les enseignants)
  async view(user: User, comment: Comment) {
    return user.role === 'teacher' || user.role === 'admin'
  }
}
```

Utilisation de la permission

Pour la mise à jour d'un commentaire :

```
  /**
   * Mettre à jour le commentaire de l'élève student_id
   *
   * Si un admin modifie le commentaire créé par un enseignant, nous ne modifions
pas le teacherId.
   */
  async update({ params, request, response, bouncer }: HttpContext) {
    // Récupération des données envoyées par le client et validation des données
```

```
    const { content } = await request.validateUsing(commentValidator)

    // Vérifie que le commentaire appartient bien à l'élève
    const comment = await Comment.query()
      .where('id', params.id)
      .where('student_id', params.student_id)
      .firstOrFail()

    // Vérifie les permissions de l'utilisateur connecté
    if (await bouncer.with(CommentPolicy).denies('update', comment)) {
      return response.unauthorized({
        message:
          "Vous n'êtes pas l'auteur de ce commentaire. Vous n'avez pas le droit de
le modifier",
      })
    }

    // Mise à jour
    comment.content = content
    await comment.save()

    // Réponse 200 OK avec le commentaire mis à jour
    return response.ok(comment)
  }
```

Pour la suppression d'un commentaire :

```
  /**
   * Supprimer le commentaire de l'élève student_id
   */
  async destroy({ params, response, bouncer }: HttpContext) {
    // Récupère le commentaire à supprimer
    const comment = await Comment.query()
      .where('id', params.id)
      .where('student_id', params.student_id)
      .firstOrFail()

    // Vérifie les permissions de l'utilisateur connecté
    if (await bouncer.with(CommentPolicy).denies('delete', comment)) {
      return response.unauthorized({
        message:
          "Vous n'êtes pas l'auteur de ce commentaire. Vous n'avez pas le droit de
le supprimer",
      })
    }

    // Suppression du commentaire
    await comment.delete()

    // On utilise `response.noContent` pour retourner un code HTTP 204 sans
contenu
    return response.noContent()
```

```
    }
  }
```

Tester !

Tester la modification d'un commentaire avec un "mauvais" enseignant

1. Je crée un commentaire avec le user GCR
2. Je tente de le modifier avec le user AMG

ATTENTION : Pour cela, je dois mettre le token de AMG dans la route

`http://localhost:3333/students/48/comments/1`

ça marche !

Tester que le user admin peut tout faire

Je créé un User via Bruno

Je change son rôle directement depuis la DB