Backend pour le suivi des élèves - Step1

Kit Starter

Nous allons commencer par créer la structure de notre projet.

```
npm init adonisjs@latest api-suivi-eleves-etml -- --db=mysql -K=api --auth-guard=access_tokens
```

Question : Sais tu à quoi servent les options de la commande ci-dessus ?

Création du modèle `Student` et la migration `Student`

```
node ace make:model -m student
```

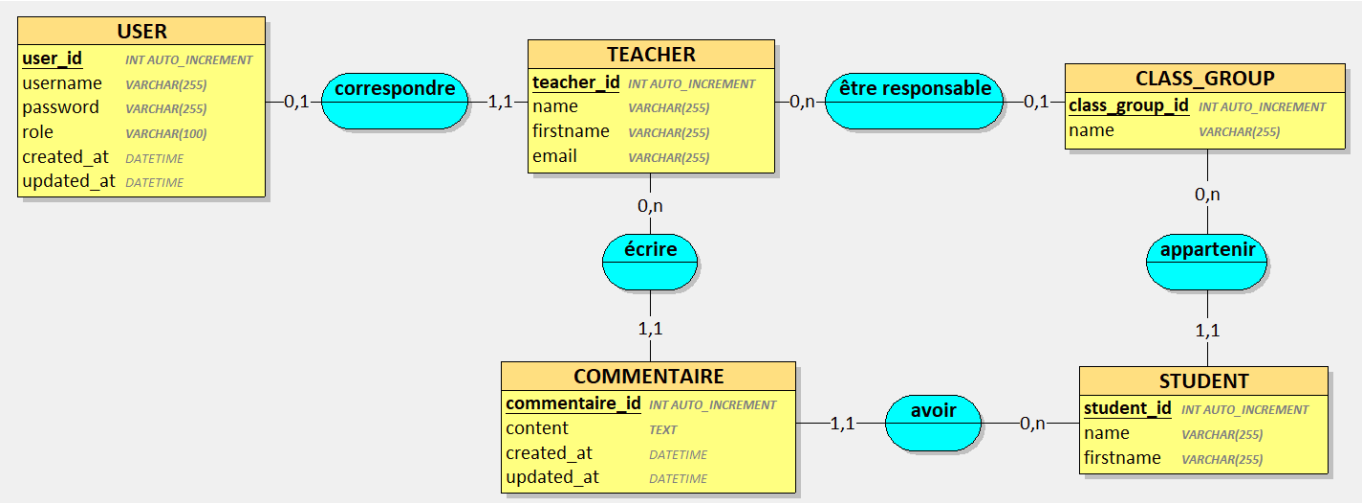
Cette commande créé les fichiers pour le modèle et la migration :

- `app/models/student.ts`
- `database/migrations/create_students_table.ts`

Question : Sais tu à quoi sert l'option `-m` de la commande ci-dessus ?

Ces fichiers ne contiennent que la structure et c'est à vous de les compléter.

Dans un 1er temps, on ne se préoccupe pas des relations entre les tables mais seulement les champs.



J'ajoute les champs nom et prénom au modèle :

```
import { DateTime } from 'luxon'
import { BaseModel, column } from '@adonisjs/lucid/orm'

export default class Student extends BaseModel {
  @column({ isPrimary: true })
  declare id: number

  @column()
  declare name: string

  @column()
  declare firstname: string

  @column.dateTime({ autoCreate: true })
  declare createdAt: DateTime

  @column.dateTime({ autoCreate: true, autoUpdate: true })
  declare updatedAt: DateTime
}
```

Question : Peux tu me décrire les caractéristiques du champ `updatedAt` ?

J'ajoute les champs nom et prénom à la migration :

```
import { BaseSchema } from '@adonisjs/lucid/schema'

export default class extends BaseSchema {
  protected tableName = 'students'

  async up() {
    this.schema.createTable(this.tableName, (table) => {
      table.increments('id')

      // Ajout des colonnes name et firstname
      table.string('name').notNullable()
      table.string('firstname').notNullable()

      table.timestamp('created_at')
      table.timestamp('updated_at')
    })
  }

  async down() {
    this.schema.dropTable(this.tableName)
  }
}
```

Nous devons maintenant exécuter la migration.

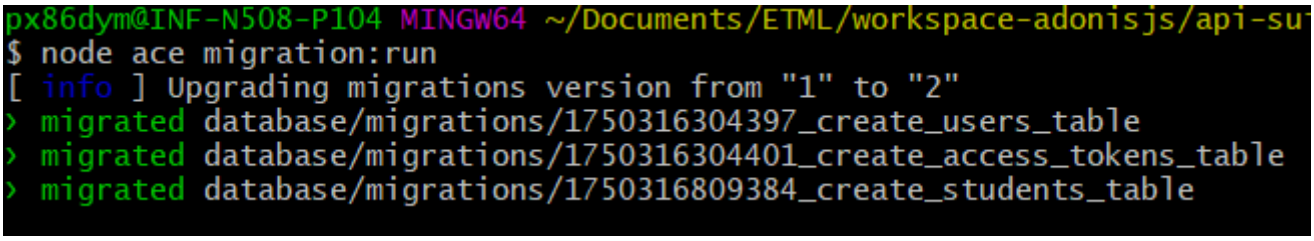
Pour que l'ORM Lucid puisse communiquer avec notre base de données MySQL, nous devons modifier le `.env` qui contient les variables d'environnement.

```
TZ=UTC
PORT=3333
HOST=localhost
LOG_LEVEL=info
APP_KEY=cErBRZVRFGcwGajbovxwCZHPU5C92N0t
NODE_ENV=development
DB_HOST=127.0.0.1
DB_PORT=6033 # Port pour MySQL via Docker
DB_USER=root
DB_PASSWORD=root # Mot de passe pour MySQL via Docker
DB_DATABASE=db_api_eleves_step_by_step # Nom de la base de données
```

Il est important de créer la base de données `db_api_eleves_step_by_step`.

Maintenant à l'aide du CLI, on peut exécuter les migrations :

```
node ace migration:run
```



A noter :

Les migrations pour les tables `users` et `auth_access_tokens` seront exécutées également. Nous nous en préoccuperons lorsque nous mettrons en place l'authentification.

Vérification de la création des tables

A vous de vous connecter à votre base de données MySQL et de vérifier que la table `students` est bien présente.