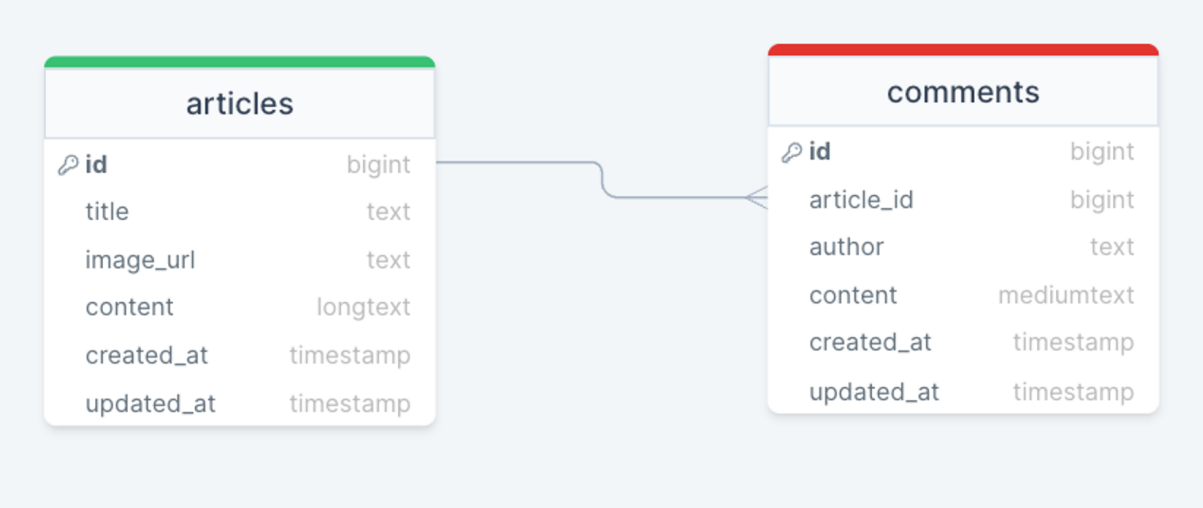
Projet Laravel

Le projet porte sur le développement d'une API REST avec Laravel.

Ce cas pratique permettra d'approfondir les fonctionnalités clés de Laravel :

- La création de tables via des migrations ;
- L'utilisation de l'ORM Eloquent ;
- La génération de données de test (Factories, Seeders) ;
- La création d'endpoints ;
- La validation de donnée ;
- Le formatage des réponses.

```
GET|HEAD   api/articles ..................................... ArticleController@index
POST       api/articles ..................................... ArticleController@store
GET|HEAD   api/articles/search/{search} ..................... ArticleController@search
GET|HEAD   api/articles/{article} ........................... ArticleController@show
PATCH      api/articles/{article} ........................... ArticleController@update
DELETE     api/articles/{article} ........................... ArticleController@destroy
GET|HEAD   api/articles/{article}/comments ................ ArticleController@comments
GET|HEAD   api/comments ..................................... ArticleController@index
POST       api/comments ..................................... ArticleController@store
GET|HEAD   api/comments/{comment} ........................... ArticleController@show
PATCH      api/comments/{comment} ........................... ArticleController@update
DELETE     api/comments/{comment} ........................... ArticleController@destroy
```



1) Créez un nouveau projet Laravel.

2) Créez des migrations pour les tables `articles` et `comments` (n'oubliez pas la clé étrangère).

3) Définissez les modèles `Article` et `Comment`. Dans `Article`, établissez une relation `comments()` pour accéder aux commentaires liés. Dans `Comment`, créez une relation `article()` pour référencer l'article associé.

4) Utilisez des factories pour générer des données factices pour les articles et les commentaires. Créez des seeders, tels que `ArticlesSeeder` et `CommentsSeeder`, pour remplir la base de données avec ces données.

Ajouter 30 articles et 150 commentaires avec des données aléatoires.

5) Utilisez des Api Ressources pour formater les réponses de l'API, assurant ainsi la cohérence et la clarté des données renvoyées par l'API.

Voici les données que nous voulons retourner :

Article :

- `id`
- `title`
- `image_url`
- `content`

Comment :

- `id`
- `article_id`
- `author`
- `content`

Exemple de réponse :

```
// Article { "id": 1, "title": "Consequatur voluptatum velit non.
Et quo sint maiores quas. In molestiae ipsam accusamus dolores.",
"image_url": "http://www.muller.com/", "content": "Rerum vel cumque
tempora animi nihil repellat. Repellendus numquam necessitatibus
hic vel ea est. Et doloribus quisquam doloremque voluptatem." }
```

```
// Comment { "id": 1, "article_id": 9, "author": "Gardner Harber",
"content": "Velit repellendus recusandae et cupiditate sunt." }
```

6) Créez les FormRequests `ArticleRequest` et `CommentRequest` pour valider les données avant le traitement.

Voici les règles de validations :

Article :

- `title` : min 10 caractères et max 30 caractères, obligatoire.
- `image_url` : url valide, obligatoire
- `content` : min 20 caractères et max 100 caractères, obligatoire

Comment :

- `article_id` : article_id existe (lien doc), obligatoire.
- `author` : min 5caractères et max 30 caractères, obligatoire.
- `content` : min 5caractères et max 50 caractères, obligatoire.

💡 N'oublier pas de renvoyer du JSON en cas de données non valides.

7) Ajouter les contrôleurs `ArticleController` et `CommentController` avec les méthodes nécessaires pour gérer les opérations CRUD sur les articles et les commentaires.

💡 Rappel commande :
`php artisan make:controller ArticleControllerController --api`
`php artisan make:controller CommentController --api`

Rajouter les deux méthodes suivantes dans `ArticleController`

```
public function search(string $search){} public function
comments(string $id){}
```

8) Ajouter les routes dans votre fichier `routes/api.php` pour mapper chaque méthodes des deux Controllers aux routes.

9) Implémenter chaque méthodes de vos `Controllers`

Notation :

- Qualité code + README + GIT + config Postman: 2 pts
- Autres fonctionnalités Laravel : 4 pts
- Migration valide : 1 pts
- Models valides avec relation : 2 pts
- Factories et Seeder valides : 1.5 pts
- JsonResource valide : 1.5 pts
- Request valide : 2

- Routes :
  - `GET api/articles` : 0.25 pts
  - `POST api/articles` : 0.5 pts
  - `GET api/articles/search/{search}` : 1 pts
  - `GET api/articles/{article}` : 0.25 pts
  - `PATCH api/articles/{article}` : 0.5 pts
  - `DELETE api/articles/{article}` : 0.5 pts
  - `GET api/articles/{article}/comments` : 1 pts

  - `GET api/comments` : 0.25 pts
  - `POST api/comments` : 0.5 pts
  - `GET api/comments/{comment}` : 0.25 pts
  - `PATCH api/comments/{comment}` : 0.5 pts
  - `DELETE api/comments/{comment}` : 0.5 pts