Project Overview

The frontend is built using React 18.2, TypeScript 5.4, Material UI 5.15 library and project uses NX 18.2 build tools.

The backend is built using PHP 8.3 and Laravel 11.0 framework. MariaDB 10.8 serves as the database system.

All components of the software are containerized using Docker. The application has been tested and confirmed to function on Windows 10 (Docker v24.0) and Ubuntu 22.04 (Docker v26.0).

Step-by-step guide to running the application.

1. Begin by cloning the repository to your local machine:

```
git clone git@github.com:toomastahves/courses.git
```

2. Navigate to cloned repository and launch application containers using Docker Compose:

```
cd courses
docker compose up -d
```

3. Once applications are running, execute database migrations and seed data by entering the backend container:

```
docker exec courses-backend /bin/bash -c "php artisan migrate --seed"
```

4. The applications should now be available on following URLs:

```
Frontend: http://localhost:5000/
Backend: http://localhost:5001/api/documentation
```

Alternative Front-End Setup Guide (without Docker)

This alternative method is useful if you encounter issues with Docker containers. Tested it with Node 20 and NPM 10 installed globally on a system:

1. Navigate to the frontend directory:

```
cd /courses/courses-frontend
```

2. Install necessary dependencies:

```
npm install
```

3. Start local development server to serve application:

```
npm run serve
```

4. Open web browser and visit application:

```
http://localhost:5000/
```

Alternative Back-End Setup Guide (without Docker)

Before you begin, ensure you have PHP 8.3 and Composer 2.7 installed and configured on your system.

1. Navigate to back-end application directory:

```
cd /courses/courses-backend
```

2. Install necessary dependencies:

```
composer install
```

3. Start application server by using command:

```
php artisan serve –port=5001
```

4. Run database migrations and seed database with initial data:

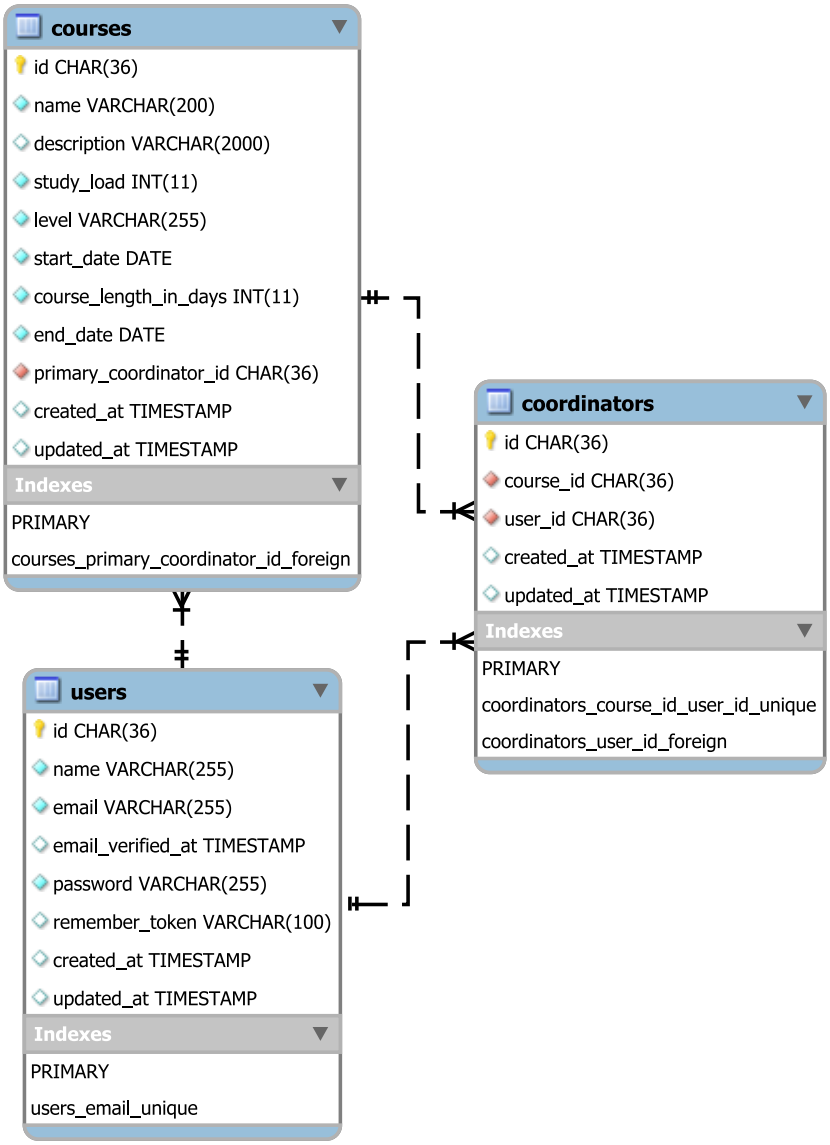
```
php artisan migrate --seed
```

5. Open web browser and confirm the application is active:

```
http://localhost:5001/api/documentation
```

Toomas Tahves 19.04.2024

Database diagram



Toomas Tahves 19.04.2024

Implemented functionalities.

Backend:

- Implemented API endpoints for CRUD operations on courses, retrieval of user lists and management of course coordinators.
- Added request field validation on the server side.
- Added Swagger API documentation (missing some fields and response details).
- Implemented migration scripts and seed data generation.

Frontend:

- Built list view displaying course details such as name, description, study load, study level, and start/end dates.
- Implemented course creation view, allowing the addition of multiple coordinators.
- Implemented course detail view with update capabilities for all course attributes.
- Added validation on the client side, implemented a delete function for courses.
- Added menu and loading spinner components.

Deployment:

- Application setup and deployment done using Docker Compose

Areas for improvement

- Implement unit testing to ensure code reliability and maintenance.
- Implement filtering and sorting functionalities to the data table.
- Setup separate build processes for development and production environments.
- Minimize code redundancy by combining code between course detail and add views.

Challenges

- Took a course to refresh my knowledge in PHP and Laravel: https://www.udemy.com/course/laravel-beginner-fundamentals/
- Invested time to ensure code was operational within containerized environments.
- Faced complexities in identifying various edge cases and potential errors. Implemented validations and wrote code to maintain application stability against unforeseen events. Acknowledging that additional improvement is needed to cover all use cases.