Examen de Project de développement web

Rédigé en anglais

Julien Pinson

EAFC Namur-Cadets 2022 – 2023

Introduction

This document will cover every changes over the initial scope. All of these features are available in the backend, but not always in the frontend.

Backend

The backend is based on Laravel 9, using a REST api approach, but keeping the session based authentication. This means the routes are only available to the frontend using CSRF cookies.

The frontend can use the POST/PUT/PATCH/DELETE only if they get a CSRF cookie using the

/sanctum/csrf-cookie route.

The frontend must then extract the value of the cookie and specify the header **X-XSRF-TOKEN** with that value.

All the routes are available in the routes/web.php due to the session based authentication.

Frontend

For the frontend, I chose to use the framework Angular 15, with a material design library.

Documentation

A PHP doc is provided in the /docs/app folder.

A postman collection is available to test the backend routes. It does include automatically the CSRF token.

Core features:

Account creation:

Available in frontend

route : POST /api/v1/register

Using these fields:

- Username (Unique)
- Email (Unique)
- Password
- Last name
- First name
- Country (foreign key)
- Birthdate
- Address
- Postal code
- Address Country (Foreign key)
- Phone
- Site Role (Foreign key)
- Picture

Side effects :

- A new user has the “guest” SiteRole by default.
- When using the /register route, the User is automatically authenticated through a session cookie.

Login to the application

Available in frontend

route : POST /api/v1/login

Using these fields :

- Email
- Password

The User is authenticated through a session cookie.

User can view their profile:

Not available in Frontend

Related routes, either available for the authenticated User or a Secretary / Administrator

Retrieve personal data:

GET /api/v1/users/{userId}

Update :

PUT /api/v1/users/{userId}

PATCH /api/v1/users/{userId}

Delete :

DELETE /api/v1/users/{userId}

Export : (CSV / JSON)

GET /api/v1/users/{userId}/export

Change picture :

POST /api/v1/users/{userId}/picture

Remove picture:

DELETE /api/v1/users/{userId}/picture

A User:

Cannot update their own SiteRole.

Cannot delete their account if their SiteRole isn’t guest.

User can see and enrol to formations :

Available in Frontend

Related routes of this feature :

Formations:

Get all formations, they can include Courses, be sorted and filtered :

GET /api/v1/formations

Enrollments :

Can be retrieved using:

GET /api/v1/users/{userId}?includeRelations=enrollments

Create :

Using the userId, formationId

POST /api/v1/enrollments

Delete :

DELETE /api/v1/enrollments/{enrollmentId}

<u>An enrollment :</u>

Cannot be deleted if their status isn't "Pending"

Cannot be updated by an administrator if their status isn't "Pending"

Back-office :

Can be accessed by the Secretary and Administrator SiteRole, they do have different authorizations.

Show a list of users:

Not available in frontend

Related routes:

Get all users: (Filtering, Sorting, Pagination)

GET /api/v1/users

Specific user actions

(See the user profile section)

Export all users (Filtering, Sorting) (CSV / JSON)

GET /api/v1/users/export

Changing role :

Can be done through the User update. Including the siteRoleId in the body is only available for Administrators.

Administrators :

Cannot delete users that aren't guests

Can apply the role "Banned" to a user, this user will only be able to log out.

(Banned users are limited on the frontend)

Validate an enrol from a user to a formation:

Available in frontend

Related routes:

GET all enrollments (Filtering, Sorting, Pagination)

GET /api/v1/enrollments

Export all enrollments (Filtering, sorting) (csv/json)

GET /api/v1/enrollments/export

Update an enrollments (With a message and a Status)

PUT /api/v1/enrollments/[enrollmentId}

PATCH /api/v1/enrollments/[enrollmentId}

Administrators :

Cannot update an enrollment if the Status isn't pending.

Cannot delete an enrollment if the Status isn't pending.

Side effects :

When an enrollments is set with the status Approved

- Their SiteRole is not "User"
- They're automatically registered as a CohortMember of the current academic year Cohort related to the formation. This Cohort can be created automatically if it doesn't exist. They get the CohortRole "Student"

Assign or remove a student or a teacher to group, this group can be used to enrol members to a specified course.

Not available in frontend

A group is called a “Cohort”, student and teacher are CohortRoles.

Related routes:

Get all cohorts (Filtering, sorting, pagination)

GET /api/v1/cohorts

Get a specific cohort data

GET /api/v1/cohorts/{cohortId}

Export all cohorts (Filtering, sorting) (csv/json)

GET /api/v1/cohorts/export

Create a cohort

POST /api/v1/cohorts

Update a cohort

PUT /api/v1/cohorts/{cohortId}

PATCH /api/v1/cohorts/{cohortId}

Delete a cohort

DELETE /api/v1/cohorts/{cohortId}

Get all cohort members

GET /api/v1/cohorts/{cohortId}/users

Get a specific cohort member

GET /api/v1/cohorts/{cohortId}/users/{userId}

Add a user to a cohort, with a specific CohortRole

POST /api/v1/cohorts/{cohortId}/users/

Update a user role of a cohort member

PUT /api/v1/cohorts/{cohortId}/users/{userId}

PATCH /api/v1/cohorts/{cohortId}/users/{userId}

Remove a user from a cohort

DELETE /api/v1/cohorts/{cohortId}/users/{userId}

Enrol all the cohort members (except teachers) to a course, a new Grade is created for them.

POST /api/v1/cohorts/{cohortId}/courses

Create, update or delete a formation, add/remove courses from formations.

Not available in frontend

Related routes:

Get all formations :

GET /api/v1/formations

Get a specific formation

GET /api/v1/formations/{formationId}

Create a new formation

POST /api/v1/formations

Update a specific formation

PUT /api/v1/formations/{formationId}

PATCH /api/v1/formations/{formationId}

Delete a specific formation

DELETE /api/v1/formations/{formationId}

Add a course to a specific formation

POST /api/v1/formations/{formationId}/courses

Delete a course from a specific formation

DELETE /api/v1/formations/{formationId}/courses/{courseId}

Formations :

Cannot be deleted if the formation have related Cohorts

Deleting the formation removes all the courses from the FormationCourse table.

Create, update, delete a course

Not available in frontend

Related routes :

Get all courses (Filtering, Sorting, Pagination)

GET /api/v1/courses

Get a specific course

GET /api/v1/courses/{courseId}

Export all courses (Filtering, sorting) (csv/json)

GET /api/v1/courses/export

Create a course

POST /api/v1/courses

Update a course

PUT /api/v1/courses/{courseId}

PATCH /api/v1/courses/{courseId}

Delete a course

DELETE /api/v1/courses/courseId}

Courses :

Cannot be deleted if they’re part of a formation (in FormationCourse table)

Out of scope tables and routes:

Country:

Related routes:

Get all countries (Filtering, Sorting, Pagination)

GET /api/v1/countries

Get a specific country

GET /api/v1/countries/{countryId}

Export all countries (Filtering, sorting) (csv/json)

GET /api/v1/countries/export

Create a country

POST /api/v1/countries

Update a country

PUT /api/v1/countries/{countryId}

PATCH /api/v1/countries/{countryId}

Delete a country

DELETE /api/v1/countries/{countryId}

EducationLevels :

Related routes :

Get all education levels (Filtering, Sorting, Pagination)

GET /api/v1/educationLevels

Get a specific education level

GET /api/v1/educationLevels/{educationLevelId}

Export all education levels (Filtering, sorting) (csv/json)

GET /api/v1/educationLevels/export

Create a education level

POST /api/v1/educationLevels/

Update a education level

PUT /api/v1/educationLevels/{educationLevelId}

PATCH /api/v1/educationLevels/{educationLevelId}

Delete a education level

DELETE /api/v1/educationLevels/{educationLevelId}

Statuses:

Related routes:

Get all statuses (Filtering, Sorting, Pagination)

GET /api/v1/statuses

Get a specific status

GET /api/v1/statuses/{statusId}

Export all statuses (Filtering, sorting) (csv/json)

GET /api/v1/statuses/export

Create a status

POST /api/v1/status

Update a status

PUT /api/v1/status/{statusId}

PATCH /api/v1/status/{statusId}

Delete a status

DELETE /api/v1/status/{statusId}

Cohort roles:

Related routes:

Get all cohort roles (Filtering, Sorting, Pagination)

GET /api/v1/cohortRoles

Get a specific cohort role

GET /api/v1/cohortRoles/{cohortRoleId}

Export all cohort roles (Filtering, sorting) (csv/json)

GET /api/v1/cohortRoles/export

Create a cohort role

POST /api/v1/cohortRoles

Update a cohort role

PUT /api/v1/cohortRoles/{cohortRoleId}

PATCH /api/v1/cohortRoles/{cohortRoleId}

Delete a cohort role

DELETE /api/v1/cohortRoles/{cohortRoleId}

Site roles:

Related routes:

Get all site roles (Filtering, Sorting, Pagination)

GET /api/v1/siteRoles

Get a specific site role

GET /api/v1/siteRoles/{siteRoleId}

Export all site roles

GET /api/v1/siteRoles/export

Create a site role

POST /api/v1/siteRoles

Update a site role

PUT /api/v1/siteRoles/{siteRoleId}

PATCH /api/v1/siteRoles/{siteRoleId}

Delete a site role

DELETE /api/v1/siteRoles/{siteRoleId}

Grades:

Store all the user course grades.

By default they're created with an empty score, which means the student hasn't been assessed yet.

This is used to determine if a user should be allowed in course subscription.

Related routes:

Get all grades (Filtering, Sorting, Pagination)

GET /api/v1/grades

Get a specific grade

GET /api/v1/grades/{gradeId}

Export all grades (Filtering, sorting) (csv/json)

GET /api/v1/grades/export

Create a grade

POST /api/v1/grades

Update a grade

PUT /api/v1/grades/{gradeId}

PATCH /api/v1/grades/{gradeId}

Delete a grade

DELETE /api/v1/grades/{gradeId}

Grades:

Cannot be created if the User has already an ongoing grade, or if they already passed with a grade superior to 49.

Cannot be updated when a score has been assessed.

Cannot be deleted, because they're meant to archive the student progress through their tuition.