Symfony Framework

Santosh Kalwar, 14.05.2024

Symfony Topics

General Discussions

Symfony Reflection Quiz

Symfony Full-stack app architecture

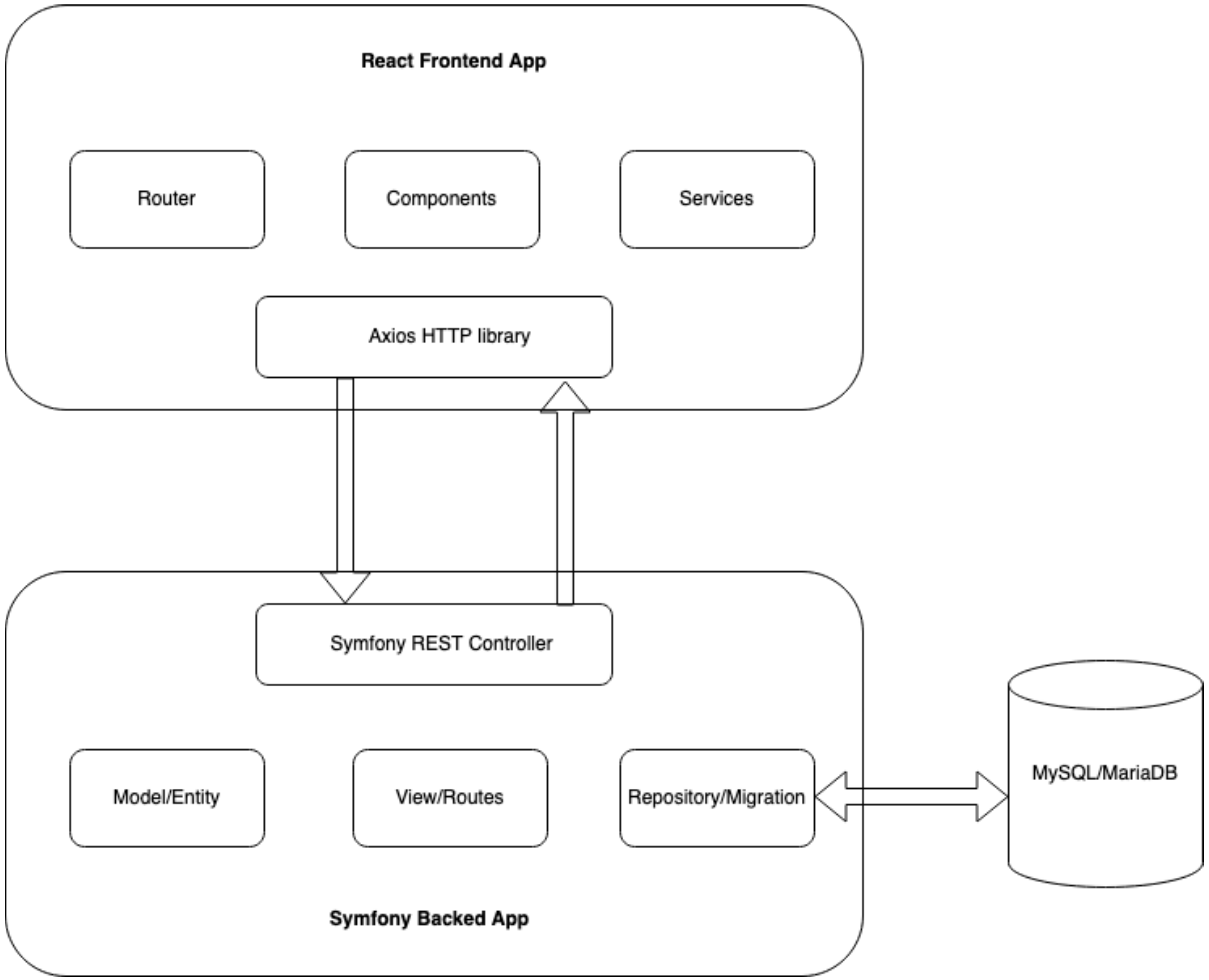
Install Vite-bundle

CORS issue

Full-stack app: Creating / Reading

Classroom Practice: Updating / Deleting

Full-stack app architecture



REACT + VITE + RESTful API + Symfony 7 + MySQL/MariaDB

Let’s use last week Symfony classroom practice 2 application: make symfony api application

What it contains?

- Backend with RESTful API
- See ProjectController.php
- MySQL/MariaDB connection

What is lacking?

- REACT
- VITE
- UI (user interface)

Before moving forward, do the following:

- Test the backend works with POSTMAN
- Make sure you can insert, edit, update and delete
- Check in phpmyadmin those operations are working properly

Install Vite-Bundle

Let’s use last week Symfony classroom practice 2 application: make symfony api application

Install the Vite-Bundle by running the following command in your Symfony project root directory (meaning “web” as you are using Symfony-MAMP):

composer require pentatrion/vite-bundle

If prompt ask for: *Do you want to execute this recipe?* **Yes**

https://github.com/lhapaipai/vite-bundle

https://symfony-vite.pentatrion.com/guide/installation.html

Install Vite-Bundle

```
    Do you want to execute this recipe?
    [y] Yes
    [n] No
    [a] Yes for all packages, only for the current installation session
    [p] Yes permanently, never ask again for this project
    (defaults to n): y
  - Configuring pentatrion/vite-bundle (>=1.0): From github.com/symfony/recipes-contrib:main
Executing script cache:clear [OK]
Executing script assets:install public [OK]

 What's next?


Some files have been created and/or updated to configure your new packages.
Please review, edit and commit them: these files are yours.

 pentatrion/vite-bundle  instructions:

         Getting started using pentatrion/vite-bundle

  Configure your transformations:
    1. Add if necessary a configuration file in config/packages/pentatrion_vite.yaml.
    2. Verify the Vite configuration in /vite.config.js.
    3. Install npm and run npm i.
    4. Start the development server: npm run dev.
    5. Start coding into assets/app.js
```

Install Vite-Bundle

Check following places in your project

- **/assets** folder
- /vite.config.js
- Config/bundles.php
- base.html.twig

```
{% block stylesheets %}
    {{ vite_entry_link_tags('app') }}
{% endblock %}

{% block javascripts %}
    {{ vite_entry_script_tags('app') }}
{% endblock %}
```

- your “assets” folder this is your main frontend folder, you can rename as you wish but if you rename then you must also change folder name from **vite.config.js**

Install Vite-Bundle

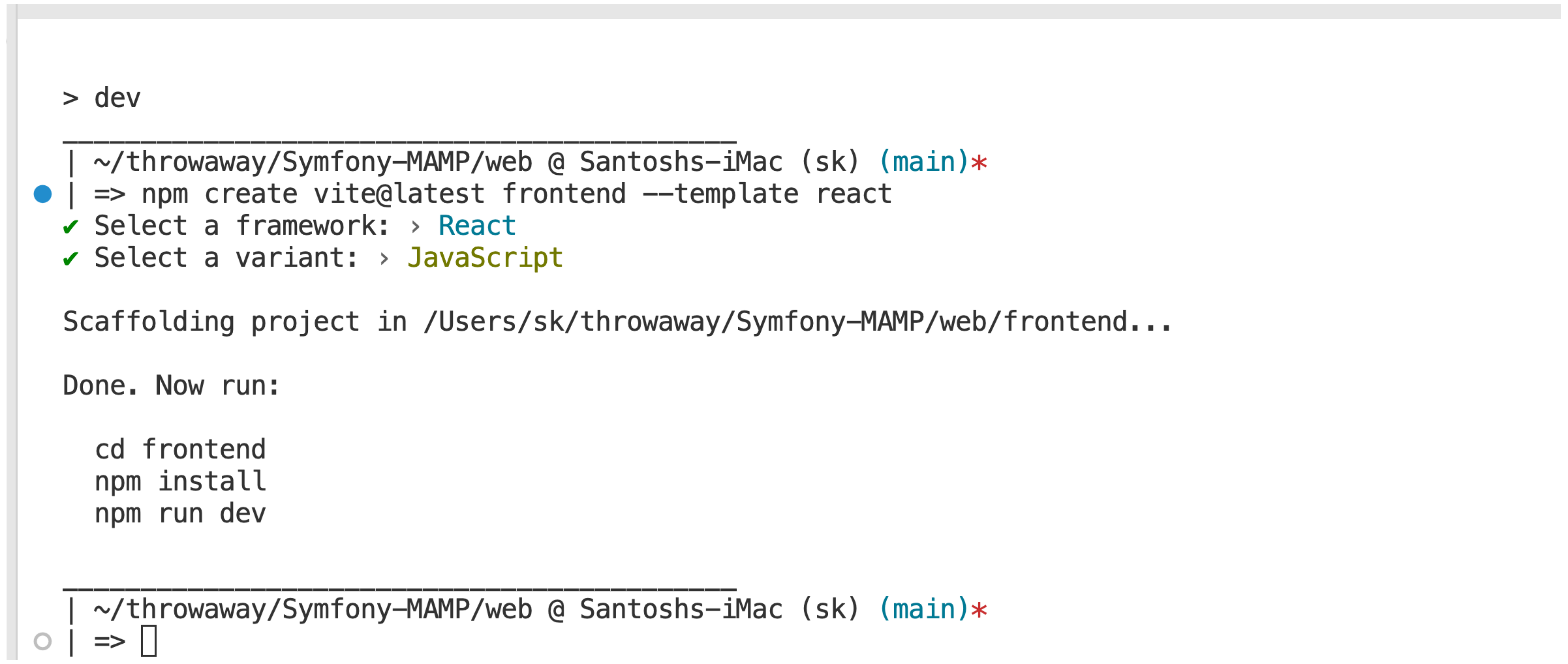
What happens if you install front-end dependencies and and run application

- npm install
- npm run dev

OR, just run it once lazily: npm i && npm run dev

Now you can create a new React application using Vite within your Symfony project. Navigate to the web folder and run:

npm create vite@latest frontend - - template react



```
> dev
______________________________________________
| ~/throwaway/Symfony-MAMP/web @ Santoshs-iMac (sk) (main)*
● | => npm create vite@latest frontend --template react
✔ Select a framework: › React
✔ Select a variant: › JavaScript

Scaffolding project in /Users/sk/throwaway/Symfony-MAMP/web/frontend...

Done. Now run:

  cd frontend
  npm install
  npm run dev

______________________________________________
| ~/throwaway/Symfony-MAMP/web @ Santoshs-iMac (sk) (main)*
○ | =>
```

Install Vite-Bundle

Configure the Vite Bundle in “web” folder to work with your new React app by updating the **vite.config.js** file and the Symfony configuration files as needed.

```
app: './frontend/src/main.jsx',
```

Develop your REACT application as you would do it normally.

CORS issue

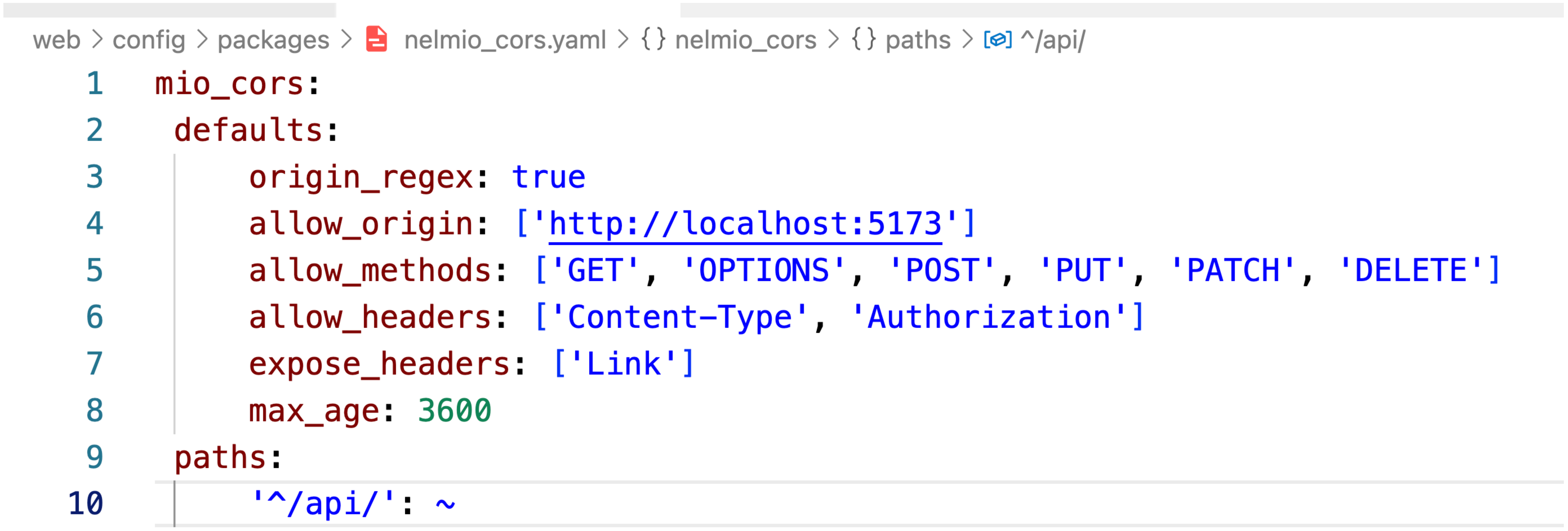
To ensure there is no CORS (Cross-Origin Resource Sharing) issue when your backend server is running on port 8007 and your React frontend is on http://localhost:5173, you'll need to set up proper headers on your backend to allow requests from your frontend origin. Here's how you can do it:

Configure Symfony to Allow CORS: You can use the nelmio/cors-bundle in Symfony web folder to set up CORS headers. First, install the bundle with Composer:

composer require nelmio/cors-bundle

Set Up CORS Configuration: After installing the bundle, configure it by creating a ***nelmio_cors.yaml*** file in the ***config/packages/*** directory of your Symfony project with the following content (shown in next slide)

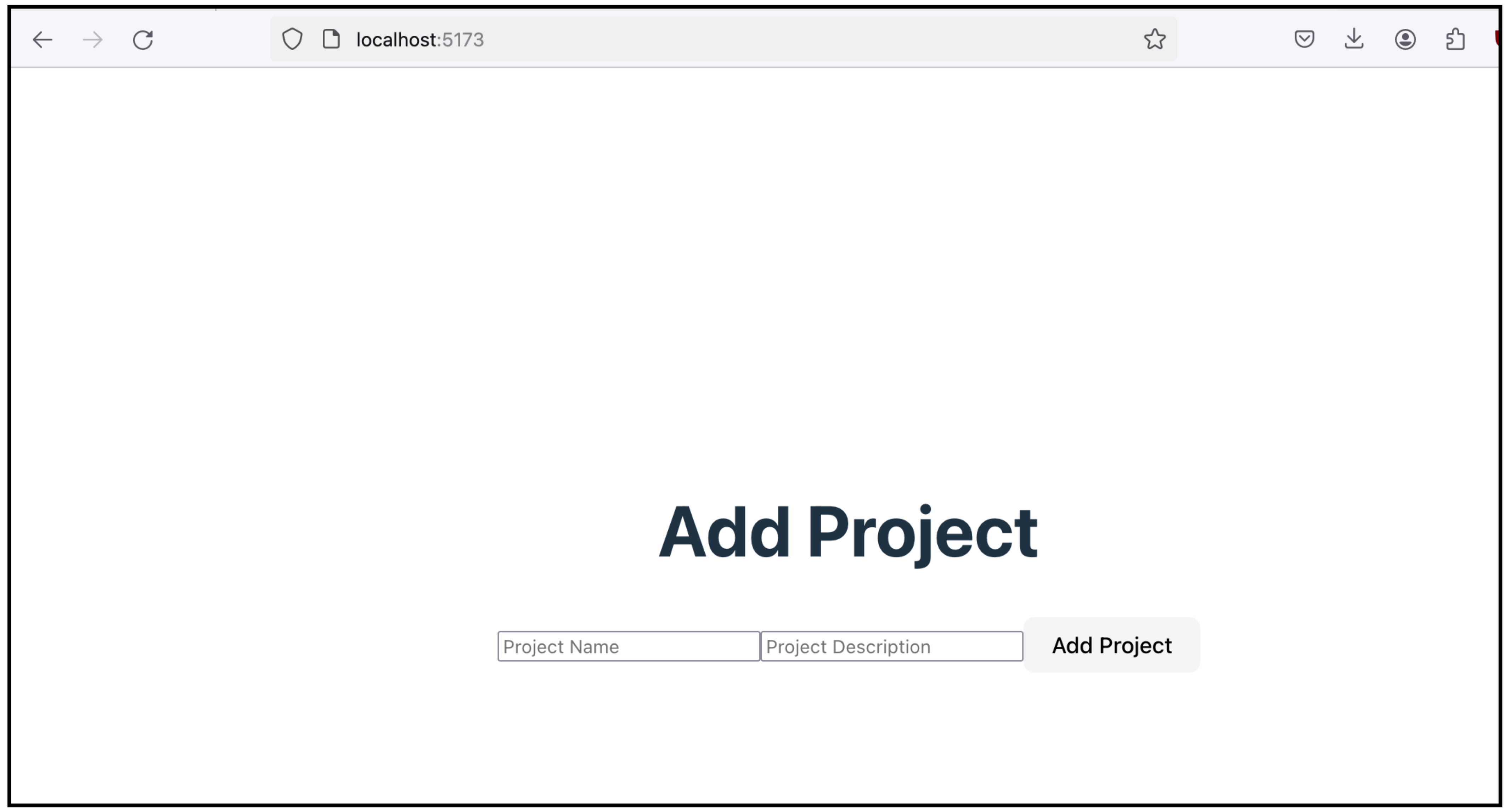
CORS issue



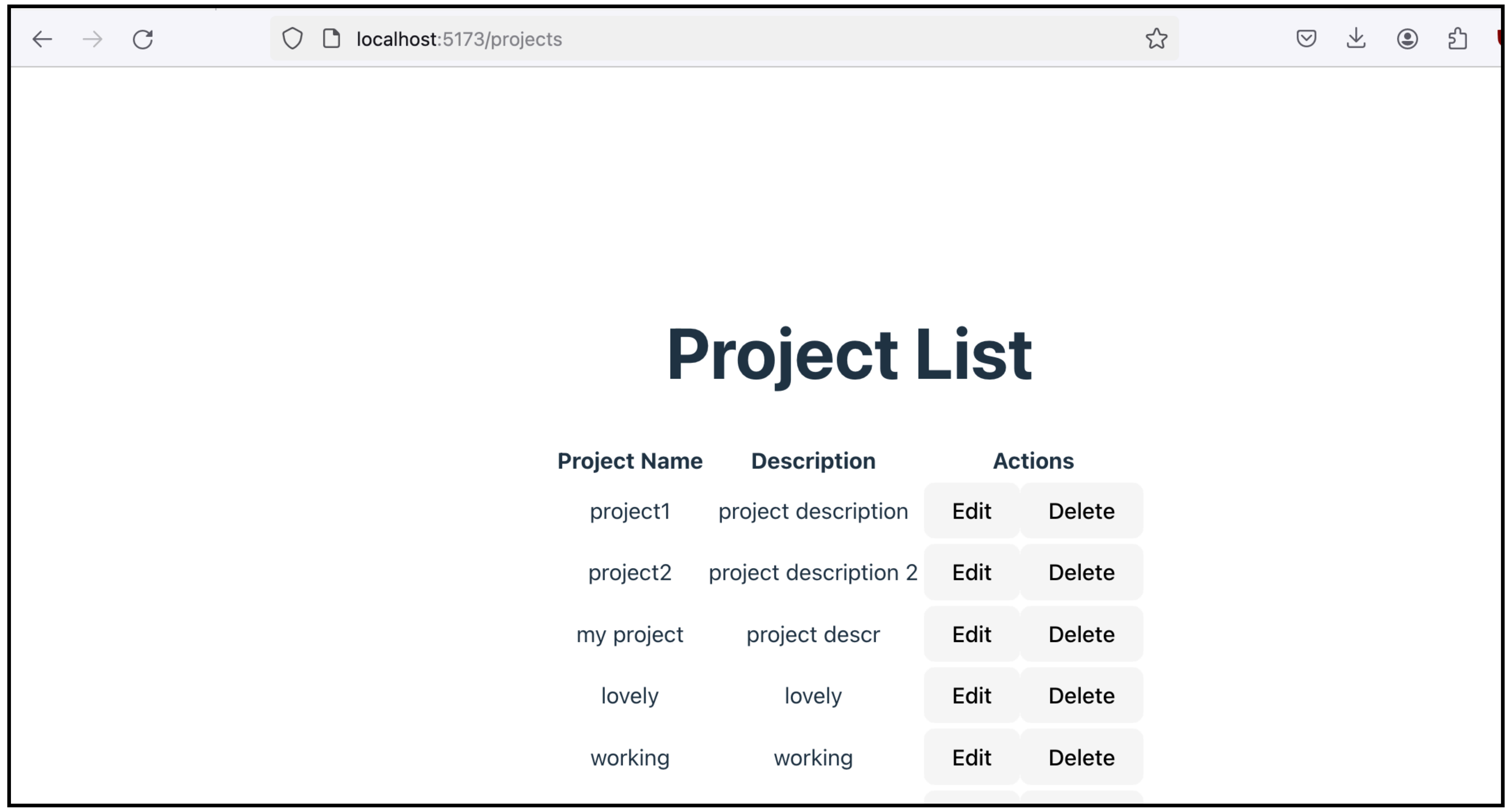
web > config > packages > nelmio_cors.yaml > {} nelmio_cors > {} paths > ^/api/

```yaml
mio_cors:
  defaults:
        origin_regex: true
        allow_origin: ['http://localhost:5173']
        allow_methods: ['GET', 'OPTIONS', 'POST', 'PUT', 'PATCH', 'DELETE']
        allow_headers: ['Content-Type', 'Authorization']
        expose_headers: ['Link']
        max_age: 3600
  paths:
        '^/api/': ~
```

Full-stack app - Creating / Reading


localhost:5173

Add Project


localhost:5173/projects

Project List

Project Name	Description	Actions	
project1	project description	Edit	Delete
project2	project description 2	Edit	Delete
my project	project descr	Edit	Delete
lovely	lovely	Edit	Delete
working	working	Edit	Delete

Classroom Practice : Add Updating / Deleting

Your classroom task now is to add:

- Updating feature / when a user clicks on “Edit” it should display that data and users should be able to update either project name or description or both

- Delete a project

- Ask user for confirmation before deleting a project

- Fix a bug when no project name or description is provided, it adds empty record

- Any css/tailwind changes you want to add