Frontend Architecture & API Integration

Here, the frontend of our web page is going to be discussed. Please get familiar with VUE.js documentation if you haven't done so yet:

https://vuejs.org/guide/introduction

A RESTful API guide could also be useful:

https://restfulapi.net/

Our frontend is a **Single Page Application (SPA)** built with **Vue.js**. We use a modular structure where every page and component encapsulates its own HTML, CSS, and JavaScript logic.

1. Directory Structure & Organization

We separate the code based on its “responsibility” in the app.

Directory	Purpose
`src/views/`	The main page containers. These are “Smart” components that usually handle data fetching for a whole page (e.g., `HomeView.vue`).
`src/components/`	Reusable UI sections. These are “Dumb” or “Presentational” components (e.g., `HeroSection.vue`, `FooterSection.vue`).
`src/components/admin/`	Specialized management modules for the Admin Dashboard. These handle the CRUD logic for news, teams, and members.
`src/services/`	The API layer. All communication with the Spring Boot backend happens here.
`src/router/`	The navigation logic. Maps URLs to specific Views.

2. Routing (Adding New Pages)

```
// src/router/index.js
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: "/",
      name: "home",
      component: HomeView,
    },
    // Add new routes here
  ]
})
```

3. Communication with Backend (API Service)

We **never** make direct `fetch` or `axios` calls inside a `.vue` component. Instead, we use a centralized service layer in `src/services/api.js`. This allows us to manage global headers, authentication tokens, and base URLs in one place.

Step 1: Register the API Endpoint

```
export const aboutUsAPI = {
  get: () => apiRequest("/organization"),
  update: (id, data) =>
    apiRequest(`/organization/${id}`, {
      method: "PUT",
      body: JSON.stringify(data),
    }),
};
```

Step 2: Use the API in a Component

```
import { aboutUsAPI } from '@/services/api';

const fetchOrganization = async () => {
```

```
  try {
    const data = await aboutUsAPI.get();
    // Do something with data...
  } catch (err) {
    console.error("API Error:", err);
  }
};
```

4. Admin Panel Architecture

The Admin Panel follows a “Manager” pattern. The `AdminView.vue` acts as the main wrapper, while the actual editing tools are found in `src/components/admin/`.

Managers: Components like `NewsManager.vue` or `EventsManager.vue` contain the forms and logic needed to edit content.

Data Flow: Typically, a Manager fetches data on mount, allows the user to edit it, and sends a `PUT` or `POST` request back through the `services/api.js` layer.

5. TODO: COOKIES, TOKEN BASED SESSIONS

Revision #7
Created 2026-04-14 12:36:02 UTC by Illia Guzerya
Updated 2026-04-15 09:23:01 UTC by Illia Guzerya