POLITECNICO
MILANO 1863

IMPLEMENTATION REPORT

HYPERMEDIA APPLICATIONS

Prof. Garzotto Franca - A.Y. 2022/2023
10/07/2023

[Wizarding Ventures Website]

[GitHub Repository]

Submitted by

Web Wizards

Roberto Giandomenico	roberto.giandomenico@mail.polimi.it
Francesca Grimaldi	francesca1.grimaldi@mail.polimi.it
Andrea Infantino	andrea1.infantino@mail.polimi.it
Federico Lamperti	federico.lamperti@mail.polimi.it

Index

1 Work Organization

As a team, it was very important to split the areas of work between us, to cooperate better and to have disjointed responsibilities.
We provide a table containing the principal areas of work.
However we wanted to learn as much as possible from the implementation of this project, so everyone did a little bit of everything.

Member	Main Areas of Work
Giandomenico	UX Design, Low-Fidelity Wireframes, DB Data Insertion, Implementation, Styling, Report creation
Grimaldi	UX Design, DB Data Insertion, Implementation, Styling, Accessibility, SEO, Report creation
Infantino	UX Design, DB Design, Implementation, Styling, Extra Functionalities
Lamperti	UX Design, DB Design, Implementation, Interaction Scenarios, Report creation

2 Our Project

2.1 Project Explanation

The project was about the creation of a website for a venture capital, in our case *Wizarding Ventures*. The project has been implemented using the Vue3 and Nuxt3 frameworks.
The final website consists of several pages and for most of them the content is retrieved from the database.

2.2 Chosen Theme

We opted for a Harry Potter theme to stick and be coherent with the name chosen for our group, *Web Wizards*. Our hypothetical Venture Capital firm has been founded in this fictional magical world by the main characters of the story.

2.3 Hosting Service

The platform we adopted to host our website is Vercel, in combination with Supabase to store our data in a Postgres DB.
First of all, this solution allowed us to create a more dynamic website, with respect to the GitHub Pages alternative.
We opted for SSR (*Server-Side Rendering*) since we believe that the server on which we are relying is very powerful and we wanted to lighten processes on client machines to allow every kind of device to access the website.
Moreover, Vercel is integrated with Git so the project is directly deployed every time there are changes to the repository.

2.4 Project's Structure

2.4.1 Pages

The website was implemented following this page structure:
pages/

- all_areas/[id].vue
  /index.vue
- our_team/[id].vue
  /index.vue
- projects/[id].vue
  /index.vue
- projects_by_area/[id].vue
  /index.vue
- about_us.vue
- contacts.vue
- cookie_policy.vue
- index.vue
- most_relevant_projects.vue
- portfolio.vue
- privacy_policy.vue

All_areas/[id].vue: page providing the detailed description of the specific area of investment. Besides the main information (logo, name, description), it also includes a preview of the projects related to it. From here, the user can access the next and previous area, ordered by the area_id in the database.

All_areas/index.vue: landing page containing all the areas of investment, with an abstract for each of them and a link to access their specific page. At the bottom of the page there are links to '*All Projects*', '*Most Relevant Projects*', '*Projects by Area*'.

Our_team/[id].vue: page for each member of the Venture Capital firm, with their personal information, description and links to their social media accounts and full CV. Users can see a preview of the projects supervised by the person, as well as those participated just as team member. They can navigate back to '*Our Team*' page and to the next and previous person, ordered by the person_id in the database.

Our_team/index.vue: page displaying all the members of the firm, ordered by *role* as default setting. If they wish, users can then choose another sorting criterion, *alphabetically* or *alphabetically reversed.* The person's preview card includes their picture, full name and role. Clicking on the card the user is led to the page of the selected person.

Projects/[id].vue: page for each single project that the company has invested in. Includes its logo, description and problem, with informational interactive links leading to the pages of the concerning areas, project supervisor and team members. Users can move to the next and previous project, ordered by the project_id in the database.

Projects/index.vue: page displaying the titles of all the projects of the company, ordered *alphabetically* as default setting. If they wish, users can then choose another sorting criterion: *alphabetically reversed*, by *relevance*, *newest first* or *oldest first.* They can navigate to '*Projects by Area*' and '*Most Relevant Projects*' pages.

Projects_by_area/[id].vue: page containing the list of projects for a specific area, with their logo and title. Links to the previous and next *Project by Area pages* are provided, along with a link to the full area page.

Projects_by_area/index.vue: page displaying all the area cards, with their logo and their name. Clicking on the card the user is taken to the specific page containing all the projects related to that area. From here, users can access '*Most Relevant Projects*' and '*All Projects'* pages.

About_us.vue: descriptive page with images and links to '*Portfolio*', '*Most Relevant Projects*', '*Our Team*' and '*Contacts*'.

Contacts.vue: page with all the useful information to reach the organization (telephone numbers, map with the headquarters location, ...). A form is also present for users to send opinions and messages.

Cookie_policy.vue: textual page with the Cookie Policy of the website.

Index.vue: homepage with a preview of nearly every page of the website and links to them. It includes the company motto and textual content along with images and charts.

Most_relevant_projects.vue: page displaying logos and titles of the 5 most important and remarkable projects for the firm according to a score stored in the database.

Portfolio.vue: descriptive page with images and links to '*All Projects*', '*Most Relevant Projects*' and '*Projects by Area*'. It also includes a preview (with logo and title) of the firm's top project.

Privacy_policy.vue: textual page with the Privacy Policy of the website.

2.4.2 Server Endpoints

GET api/all_areas/[id]: endpoint that retrieves the details of an area. Its id is passed as a URL parameter. The response is either an area Object (with its id, name, fulldescription and related projects) or an error with its message.

GET api/all_areas/index: endpoint that retrieves all the areas. The response is either an Object containing all the areas (with their id, name and short description) or an error with its message.

GET api/our_team/[id]: endpoint that retrieves all the information of a specific person. Their id is passed as a URL parameter. The response is either a person Object (with their id, name, surname, email, role, description, cv_link, hiring_date, id and title of the projects they took part in) or an error with its message.

GET api/our_team/index: endpoint that retrieves the full list of people working at the firm. The response is either an Object containing all the people (with their id, name, surname and role), ordered alphabetically, or an error with its message.

GET api/projects/[id]: endpoint that retrieves the details of a project. Its id is passed as a URL parameter. The response is either a project Object (with its id, score, title, presentation text, problem text, foundation year as well as the respective area e members involved) or an error with its message.

GET api/projects/index: endpoint that retrieves all the projects. The response is either an Object containing them all (with their id, title, score, foundation_year), ordered alphabetically, or an error with its message.

GET api/projects_by_area/[id]: endpoint that retrieves the projects of an area. Its id is passed as a URL parameter. The response is either an area Object (with its id, name and related projects) or an error with its message.

GET api/getAreas: endpoint that retrieves the total number of areas. The response is either an integer number or an error with its message.

GET api/getPeople: endpoint that retrieves the total number of people. The response is either an integer number or an error with its message.

GET api/getProjects: endpoint that retrieves the total number of projects. The response is either an integer number or an error with its message.

GET api/getAreas: endpoint that retrieves the total number of areas. The response is either an integer number or an error with its message.

GET api/portfolio: endpoint that retrieves the project that has the highest score in the database. The response is either a project Object (with its id and title) or an error with its message.

GET api/top_projects: endpoint that retrieves the first five projects in the database with the higher score. The response is either a collection of projects (each one with its id and title) or an error with its message.

POST api/contactForm: endpoint that inserts a message to the corresponding table of the database (the user only inserts fields "*email*" and "*message*", while "*id*" and "*sent_at*" are automatically generated). The response is either the record inserted or an error with its message.

2.5 Components Implemented

To reduce the redundancy of code, we decided to implement the following Vue components:

- *ContactForm*: this component contains the code that, with the use of Vuetify, creates a form with two fields, e-mail and text message. The validity of the fields is checked by regular expressions, and a different message is shown to the user after the submit based on its outcome;
- *Footer*: the bottom bar that appears in every page of the website, containing the clickable company logo, links to privacy and cookie policy and the company social media profiles;
- *Header*: the top bar that appears in every page of the website, containing the clickable company logo and the links to the pages: 'Most Relevant Projects', 'Portfolio', 'Projects', 'Projects by Area', 'Areas', 'Our Team', 'About Us', 'Contacts'. This component is particularly responsive since, on smaller screen, all the pages links disappear to be replaced with a so-called hamburger menu that, when activated, opens a sidebar menu with the same entries to make the website mobile-friendly;
- *HomepageChart*: this component contains the interactive doughnut chart created with the Chart.js library;
- *MyBackButton*: button that appears in every page, except for the Homepage, that allows the user to go back to the previously visited page;
- *MyTitle*: component that appears on top of every page, except for Homepage, that shows the back button and the title of current page. The title bar hides itself when the user scrolls down but reappears as soon the user scrolls up. In most of the webpages the title is not clickable, except for the single person page, where it leads to the full Our Team page. Props: **'title'** is the title to be displayed, **'link'** is the link to be redirected to.
- *PersonCard*: component that contains picture, full name and role of the team member. Props: **'id'** person id, **'name'** person name, **'surname'** person surname, **'role'** person role, **'link'** link to the page of the specific person;
- *ProjectCard*: component that contains the logo and the name of the project. Props: **'id'** project id, **'title'** project title, **'link'** link to the page of the specific project, **'img_bool'** boolean value to state whether the logo should be displayed or not;
- *ProjectCarousel*: component made with Vuetify that creates a carousel with the project logo and project name. The shown projects are passed through an array. Props: **'projectsArray'** the array of projects to be shown;

- *ScrollDownButton*: it is a button displayed at the bottom of the Homepage that make the web page slide after the introductory image;

- *ScrollToTop*: button that scrolls up to the top of the current page of the website, located in the bottom-right corner of the screen;

- *SingleArea*: component that contains the symbol and the name of the area. Props: **'id'** area id, **'name'** area name, **'link'** link to the page of the specific area.

2.6 Extra Modules

In this section we provide a list of all the external modules we installed to improve the website, each one with a short description of how we practically used it in our project:

- *Vuetify*: library used to implement some Vue components (carousels and form);
- *Sass*: necessary to style Vuetify components;
- *Font-awesome*: toolkit that allowed us to use some icons;
- *Chart.js*: used to create an interactive doughnut chart;
- *Nuxt Content*: used to manage static content, stored in json files (i.e., long texts in privacy policy);
- *Supabase*: used to create an easily manageable *Postgres Database*.

2.7 Accessibility and SEO

2.7.1 Accessibility

Accessibility is very important nowadays to allow everyone to enjoy content on the web.
This is what we focused on:

- every page has text that can be easily read due to an appropriate font dimension and a good contrast between text and background colors;
- some decorative HTML tags, for example <*strong*>, were used to highlight more relevant contents;
- every image has an *alternative text* that web readers can read, except for decorative ones that in which the *alt* attribute is empty;
- every button and link present a label to describe their functionality;
- the website can also be navigated using a keyboard.

We wanted to design every page and the structure of the entire website to be as understandable as possible to avoid and correct mistakes and to help users in finding the information they want.
To do so, we relied on the following tools and extensions that helped us to get the best possible result:

- *Color ally* : to test contrast between text and background colors;

- *Chrome Screen reader* : to ensure that web readers can effectively read everything important on the screen;
- *Lighthouse* and *Wave* : to test accessibility and performances, in an accurate way.

2.7.1.1 Responsiveness

Our website is designed to work well on every screen or window size and this is possible due to some techniques that we applied during the styling of pages.
Indeed, every dimension is coded with dynamic measures such as the CSS operator % or using viewports width or height.
Furthermore, CSS classes behave in different ways according to screen dimensions using the *CSS media queries*.

2.7.2 SEO

We want our website to be visited by many people.
Search Engine Optimization is essential to achieve this result since it sets the conditions for web crawlers to assign higher scores to such websites.
To do so, we used some techniques:

- the addition of title, description and keywords attributes inside the *meta tag* of head, with information regarding every single page;
- the addition of various *meta tags* for a better appearance of the website link when shared on social media (e.g. "*og:image*", "*twitter:card*"...);
- every link has a descriptive text with its specific meaning (e.g. *"Check all the projects"),* avoiding general ones (e.g. "*Click Here*").

We verified that the optimization was done in the best way possible using *Lighthouse* and *Meta SEO Inspector*.

2.8 Extra Functionalities

We added some features that we were not mandatory, but that can have a great impact on website usability.

2.8.1 Sorting

In pages where there are lists there is the possibility of changing the displaying order of the items to help users find what they are looking for in a quicker way. For example, in Our Team the user can decide to display the list of members alphabetically or by role.

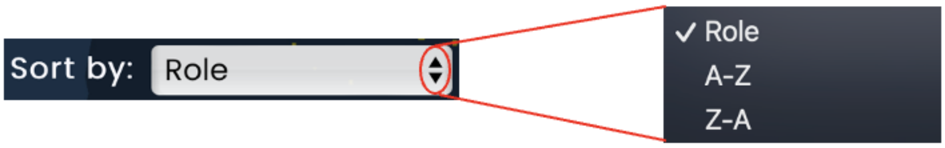
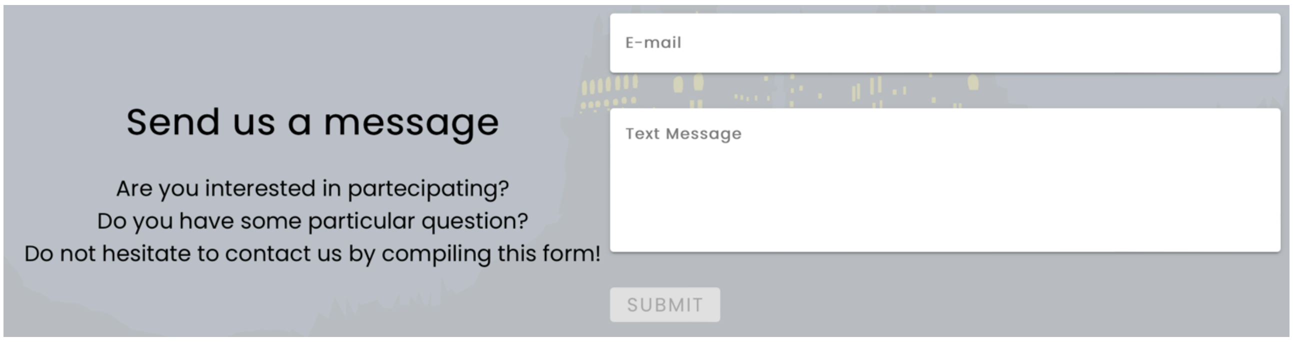


Figure 8.1: example of sorting menu

2.8.2 Contact form

In the Contact us page, we added a form using the Vuetify library, with E-mail and Text Message fields through which the user can send feedback or questions to the company. Both fields, obviously are prevented to be empty, and, moreover, the correctness of E-mail field is checked by means of a regular expression in the *ContactForm* component. Then if the two fields are suited, the content can be sent to the server with a POST request to the database. At this point, the server saves the e-mail and the content of the message in a specific table in the database, making all the messages reachable at any time for Wizarding Ventures employees. The client then opens a little modal window to show either the success message (Figure 8.3), if the request has been correctly sent, or an error one otherwise.


Send us a message

Are you interested in partecipating?
Do you have some particular question?
Do not hesitate to contact us by compiling this form!

Figure 8.2: Contact Form

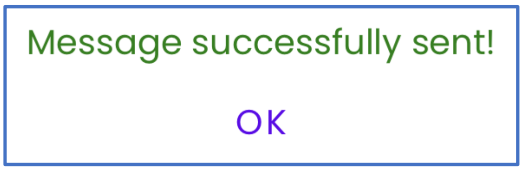

Message successfully sent!

Figure 8.3: Message successful response

2.8.3 Homepage Chart

In the homepage we added a chart that graphically shows the number of projects, members and areas of investment, using the modules of Chart.js.
Note that by clicking on one field in the above legend you can show/hide the fields and by hovering you can read the actual number of instances in the relative category.

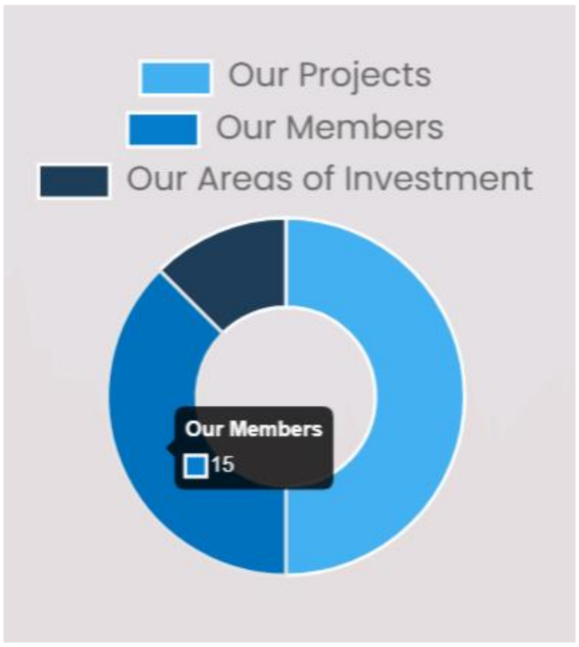


Figure 8.4: Homepage chart

2.8.4 Scroll up button

A button that allows you to scroll up to the top of the current page is present throughout the website, in the bottom-right corner of the screen.

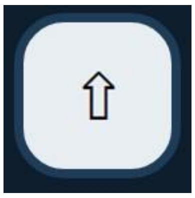


Figure 8.5: Scroll up button