POLITECNICO
MILANO 1863

Implementation Specification

Hypermedia Applications

Student	Person Code
Davide Di Marco	10667065
Stefano Fossati	10569836
Davide Maffi	10630074
Marco Romanini	10613151

Contents

1 Abstract

This design report is about the Webapp of StartMeUp, a hypothetical venture capital. All of the company's information are available on the website, along with a mechanism to get in touch with them if you want to collaborate or suggest a new project for them to take on. Information about the firm is provided in the areas of investment, portfolio projects, and staff. The website's objective is to take the user on a delightful tour through the business and its guiding principles.

2 Teamwork

2.1 Components creation

The work is divided into 4 principal parts associated to each group member but we didn't work in a compartmentalized way. Once one person had found a nice way to solve an implementation problem, the solution was shared with the others in order to have consistent code. The aforementioned 4 parts and each respective person are in the Table 1.

All Areas and Area pages, Header and Footer	Davide Di Marco
All Projects and Project pages, Database	Stefano Fossati
Homepage, Team and Person pages, Animations	Davide Maffi
About Us and Contact Us pages, Forms handling	Marco Romanini

Table 1: Parts of the website associated to each team member

2.2 Work division

The StartMeUp webiste can be found here and we also structured a Github repository in order to parallelize the work on the different parts being consistent with the standard way of managing a team work. First we locked the *main* in order to not push there until the website was ready to be published, then we created a common branch called *develop* where the code from each of us could be merged with the already working parts. Finally each member created one or more personal branches as he is used to work. The work proceeded almost in parallel for the entire development except for some critical points of synchronization. For instance the reuse of the components was allowed by the push of that code on the *develop* and then pulled to the requiring branch in order to integrate them.

3 Documentation

3.1 Website specifications

The project, named StartMeUp, is based on a made-up venture capital firm that aims at giving investment opportunities in various hi-tech areas of interest, such as Green Energy, Medical Devices & Equipment, Accessible Technology, Artificial Intelligence, and Smart Mobility. Users can find info about the company and its work or contact it through the available forms (to propose projects or became a member) or contact info.

3.2 Hosting Service

The hosting service used is Vercel, suggested by the Professors for being free of charge. It is highly compatible with the chosen Database (Supabase) and one of the most famous Version Control Systems (GitHub).

3.3 Rendering mode

The rendering mode used is Universal (server-side + client-side) since it is the default one for Nuxt projects. When the browser requests a URL, the server returns a fully rendered HTML page to the browser. Whether

the page has been generated in advance and cached or is rendered on the fly, at some point, Nuxt has run the TypeScript code in a server environment, producing an HTML document. Users immediately get the content of our application, contrary to client-side rendering. This step is similar to traditional server-side rendering. To not lose the benefits of the client-side rendering method, such as dynamic interfaces and pages transitions, the browser loads the TypeScript code that runs on the Server in the background once the HTML document has been downloaded. The browser interprets it again (hence Universal rendering) and takes control of the document and enables interactivity. Making a static page interactive in the browser is called *Hydration*.

3.4 Structure of the project

The whole project has a default layout that incorporates the two components that are always present: header and footer. The central part is then hydrated and filled with the page based on the URL path the user has landed on, since our rendering mode is Universal. Some useful and recurrently used code lines were developed as components.
The project has also an error page where the user lands when an error occurs. This page keeps the header and the footer but gently redirects the user to the Homepage.

3.4.1 Structure of the link/pages

The projects opens in a Homepage, where there is a short introduction about the StartMeUp company and helps the user understand the possibilities and the main areas of interest of the Venture capital.
On the top there are links to the principal pages, those are hidden behind a menu button as the screen becomes smaller. From the main page the user can reach all the useful pages, from the Area of interest of the company to all the projects already developed and even the team information. There are also connections to submitable forms and the About Us page if the user wants to further understand what the company stands for. Inside the Areas page, there is a list of all the areas the company has invested into. The user can click on any area to understand more about that particular one and discover some of the already successful projects. In the Projects page, the most relevant project tab shows the 4 most successful, which are shown directly to the user once the page is opened. With a click on the other tabs under the header, the user can then navigate to all the developed projects, grouped by area. Into the Team page, all the people working in the company are presented through their picture, name and role. The user can show more information by clicking to the personal card and he/she can also contact the chosen person. In the end the user can understand better the goal of the company by clicking on the About us section or he/she can become a member or submit a project, that will be later taken into account, through the apposite forms.

3.4.2 Available Server Endpoints ”Server APIs”

GET

- /api/areas
- /api/areas/$area_title
- /api/projects
- /api/projects/$project_title
- /api/team
- /api/team/$surname_name
- /api/team/$surname_name/projects
- /api/about_us

POST The POSTs are implemented server side and their APIs are available. The cackend logic is simple without data saved in the database.

- /api/contact_us/pp_form
- /api/contact_us/wwu_form

3.4.2.1 /api/areas

- **Request**
  Type of request: GET
  No params
- **Response**
  - StatusCode = 200 → The response contains an array of Area object that corresponds to the info in the areas table in the database (the Area object is defined below)

```
export interface Area {
    area: string,
    area_title: string;
    area_description: string[];
    area_color: string;
    area_image: string[];
    projects: ProjectAreaPreview[];
}
```

```
export interface ProjectAreaPreview {
    project_title: string;
    main_image: string;
    short_description: string;
    most_relevant: number;
}
```

  - StatusCode = 500 → StatusMessage = ”Error form areas get”
  - StatusCode = 404 → StatusMessage = ”Not found”

3.4.2.2 /api/areas/$area_title

- **Request**
  Type of request: GET
  area_title → parameter that corresponds to the area’s key in the DB
- **Response**
  - StatusCode = 200 → The response contains an Area object (defined as below)

```
export interface Area {
    area: string,
    area_title: string;
    area_description: string[];
    area_color: string;
    area_image: string[];
    projects: ProjectAreaPreview[];
}
```

```
export interface ProjectAreaPreview {
    project_title: string;
    main_image: string;
    short_description: string;
    most_relevant: number;
}
```

  - StatusCode = 500 → StatusMessage = ”Error form area_id get”
  - StatusCode = 404 → StatusMessage = ”Not found”

3.4.2.3 /api/projects

- **Request**
  Type of request: GET
  No params
- **Response**
  - StatusCode = 200 → The response contains an array of Project object that corresponds to the info in the areas table in the database (the Project object is defined below)

```
export interface Project {
    project_title: string;
    short_description: string;
    project_description: string[];
    start_date: Date;
    end_date: Date;
    team: ProjectSupervisor;
    areas: AreaProjectPreview[];
    budget: number;
    main_image: string;
    gallery_images: ImageGallery[];
    most_relevant: number;
}
```

```
export interface ProjectSupervisor {
    id: number;
    name: string;
    surname: string;
}
```

```
export interface ImageGallery {
    description: string;
    url: string;
}
```

```
export interface AreaProjectPreview {
    area: string;
    area_title: string;
    area_color: string;
}
```

– StatusCode = 500 → StatusMessage = ”Error form projects get”
– StatusCode = 404 → StatusMessage = ”Not found”

3.4.2.4 /api/projects/$project_title

- **Request**
  Type of request: GET
  project_title → parameter that corresponds to the project’s key in the DB
- **Response**
  – StatusCode = 200 → The response contains an Project object (defined as below)

```
export interface Area {
    area: string,
    area_title: string;
    area_description: string[];
    area_color: string;
    area_image: string[];
    projects: ProjectAreaPreview[];
}
```

```
export interface ProjectAreaPreview {
    project_title: string;
    main_image: string;
    short_description: string;
    most_relevant: number;
}
```

– StatusCode = 500 → StatusMessage = ”Error form project_title get”
– StatusCode = 404 → StatusMessage = ”Not found”

3.4.2.5 /api/team

- **Request**
  Type of request: GET
  No params
- **Response**
  – StatusCode = 200 → The response contains an array of Person object that corresponds to the info in the areas table in the database (the Person object is defined below)

```
export interface Person {
    id: number;
    name: string;
    surname: string;
    role: string;
    age: number;
    address: string;
    phone: string;
    email: string;
    img: string;
    pitch: string[];
}
```

- StatusCode = 500 → StatusMessage = ”Error from people get”
- StatusCode = 404 → StatusMessage = ”Not found”

3.4.2.6 /api/team/$surname_name

- **Request**
  Type of request: GET
  surname_name → parameter that corresponds to the person name and surname in the DB
- **Response**
  - StatusCode = 200 → The response contains a Person object (defined as below)

```
export interface Person {
    id: number;
    name: string;
    surname: string;
    role: string;
    age: number;
    address: string;
    phone: string;
    email: string;
    img: string;
    pitch: string[];
}
```

  - StatusCode = 500 → StatusMessage = ”Error from person get”
  - StatusCode = 404 → StatusMessage = ”Not found”

3.4.2.7 /api/team/$surname_name/projects

- **Request**
  Type of request: GET
  surname_name → parameter that corresponds to the person name and surname in the DB
- **Response**

– StatusCode = 200 → The response contains an array of Project object that corresponds to the info in the project table in the database (the Project object is defined below)

```
export interface Project {
    project_title: string;
    short_description: string;
    project_description: string[];
    start_date: Date;
    end_date: Date;
    team: ProjectSupervisor;
    areas: AreaProjectPreview[];
    budget: number;
    main_image: string;
    gallery_images: ImageGallery[];
    most_relevant: number;
}
```

```
export interface ProjectSupervisor {
    id: number;
    name: string;
    surname: string;
}
```

```
export interface ImageGallery {
    description: string;
    url: string;
}
```

```
export interface AreaProjectPreview {
    area: string;
    area_title: string;
    area_color: string;
}
```

– StatusCode = 500 → StatusMessage = ”Error from person-projects get”

– StatusCode = 404 → StatusMessage = ”Not found”

3.4.2.8 /api/about_us

- **Request**
  Type of request: GET
  No params

- **Response**

  – StatusCode = 200 → The response contains an array of AboutUs object that corresponds to the info in the about_us table in the database (the AboutUs object is defined below)

```
export interface AboutUs {
    id: number;
    title: string;
    subtitle: string;
    paragraph: string;
    image_url: string;
    color: string;
    icon: string;
    alternative_text: string;
}
```

  – StatusCode = 500 → StatusMessage = ”Error form about_us get”

  – StatusCode = 404 → StatusMessage = ”Not found”

3.4.2.9 /api/contact_us/pp_form

- **Request**
  Type of request: POST
  Body:The body contains an object of a PPForm (defined as below)

```
export interface PPFormData {
    name: string;
    surname: string;
    email: string;
    phone: string;
    proj_name: string;
    proj_desc: string;
}
```

- **Response**
  - StatusCode = 200 → StatusMessage = ”OK”
  - StatusCode = 500 → StatusMessage = ”Invalid form”

3.4.2.10 /api/contact_us/wwu_form

- **Request**
  Type of request: POST
  Body:The body contains an object of a WWUForm (defined as below)

```
export interface WWUFormData {
    name: string;
    surname: string;
    email: string;
    phone: string;
    brief_intro: string;
}
```

- **Response**
  - StatusCode = 200 → StatusMessage = ”OK”
  - StatusCode = 500 → StatusMessage = ”Invalid form”

3.5 List of Components implemented

- AreaSection:
  Area Section is a component used in the all areas display. It main purpose is to retrieve from the Database all the useful information regarding the area and show it on screen dynamically. For each area present it displays an image width full based on the screen visualizing it and in the center the Name of the area and the color associated with it. The color is useful to identify the area and will be reused later on, in the projects list as a tag.

- ProjectInfo:
  Project info is the component that shows all the useful information about a project, referring to Supervisor, Budget, Start Date, End Date and Related Areas. This is shown on the right part of the screen, next to the project description, that is handled by another component. The becomes hidden behind a drop-down menu when the screen is resized under a certain dimension.

- ProjectInfoRow:
  This is a subcomponent used by ProjectInfo, its purpose is display each single row composed by an SVG, a short description and the belonging class.

- ProjectByArea:
  Here, an area and its relative projects are composed to be later shown. A single area, with its relative color, is displayed and has all its own project listed. The ProjectPreview component is reused to show the list of all available project belonging to that specific area, obviously, project belonging to multiple areas are displayed once each area. In order not to be too cumbersome, the projects could be hidden by clicking on the arrow next to the area name, so that those project that does not complain to the interest of the user could be avoided, making the page more readable and easier to navigate through.

- About Us:
  The About us section is meant to explain who we are and let the user familiarize with our work through multiple section, from a simple explanation about our company to which are our investment focus. Each section is characterized by the title, a color, an image, a little SVG on the left that is placed always besides its section and a description. All this is retrieved from the database and dynamically shown in the page.

- BackButton:
  Component that implements a button useful to go back to the area or project that the user was visualizing before. In the case of a small size screen it is a simple SVG circle with a left pointing arrow in the top-left corner of the screen, but it can be enriched with a label on bigger size screens.

- Dialog:
  This is a HTML dialog component that can ask the user a question and can redirect to a different website (Google Maps for locations) or contact someone (calling a phone number or emailing). In order to achieve this different goals, a function has to be passed to the component that will be triggered when the user confirms the action.

- Footer:
  The footer is a flex component with a left side composed of the Logo to be coherent with the header disposition (slightly bigger that the one on the top) and on the right useful links to orientate in the Website. Next, an horizontal line separates these components from other contact info like our group name on the left and link to socials like Facebook, Instagram, Github and Discord on the right. The disposition changes as soon as the screen reaches a dimension lower than 640px the flex direction changes from row to col with logo and useful link moving to the center, in order to be more appealing on a smartphone.

- FormSentDialogBox:
  This is the component used to show a positive response after a project is submitted and confirm that the request was sent.

- Header:
  The header component is a reusable items composed of a screen-wide div with the Logo on the left and a flex component on the right. The right part consist of all the label useful to reach the main pages of the Webapp like Areas, Projects, various form to contact the team and also a button to switch from dark to light mode. Since the header has many label, in order to have a responsive and a good looking design as soon as the the screen reaches a width lower than 1024 pixels, the right part of the header became hidden and a burger-menu icon appears.
  This button allows the menu to be showed when clicked and keep the design clean and responsive even on lower screen dimensions.

- InfoBox:
  This component is the responsible for the displaying of the Company info like address, company phone, location and so on. All the information are linked to an app that can handle the request like, clicking on the address the Dialog component is invoked to inform the user that another tab with the Google Maps Display of the Company location.

- ProjectPreview:
  This component is widely used. The main purpose is to display a single project with its main image, a blurred part on the bottom containing the name of the project and eventually a short description and a set of tags depending on the areas it belongs. Is used in a single area to show projects relative to that specific

area, obviously here the tag is omitted since is easy to retrieve the area in which the user currently is.
Is used to show all the projects available in the Projects tab and also inside a member page to display the projects that member has taken part into.

- ProposeProject:
  ProposeProject is the component responsible for showing all the fields needed to be filled in order to propose a project to StartMeUp. Is also responsible for all the assertion on the Name, e-mail address format constraint, phone number and so on.

- WorkWithUs:
  This last component is used to compose a work request proposal, where users can introduce themselves, leave all the contact details and even submit a resumé. This form as up above re-utilizes the dialog component in order to prevent mistake while compiling or sending.

3.6 Extra components used

- Supabase, database used for retrieving information
- Vercel, platform for frontend developing and static websites
- Nuxt, web framework of the Next.js family
- Tailwind, CSS framework with packed and customizable classes
- Vuetify, framework used only to implement the timeline in the ”About Us” page

3.7 Extra functionalities

- A switch that changes from light to dark mode and viceversa, also the starting mode is based on the favourite theme of the browser.
- The *dialog* helps the user to visually see that he/she is leaving the website in order to reach external links or contact a person.
- The *details* HTML element is very useful because, when the screen width is small, long text paragraph will be reduced into smaller box that could be expanded on click. This helped us to keep the same design over a plethora of different screen sizes.

3.8 Accessibility features

All the important buttons and items have been marked with labels and title so that a text-2-speech software would allow a person to interact with the website in the correct way. Also the images have an alternative text that helps both for accessibility purpose and having a placeholder whether the picture cannot be or (in case of slow connection) is not already downloaded from the database.

4 Reference

- GitHub Repo: https://github.com/stefanofossati/hypermedia-website
- Website: https://hypermedia-website.vercel.app/
  https://hypermedia-website-stefanofossati.vercel.app/