Technical Documentation

Introduction

Group name: TheCores

Technical documentation for the ptoject of the course "Hypermedia Application (Web and Multimedia)" of Politecnico di Milano

Group composition

Members:

- Samuele Scherini - 10674683
- Alessandro Sironi - 10680296
- Elisa Tognini - 10699654

Links

Github repository: https://github.com/AlessandroSironi/HYP-Project-23 Website Link: https://coreventure.vercel.app

---

Work breakdown

We developed the project by defining common guidelines for development. Each member of the group was able to touch on all aspects relating to the website development, from the design and implementation of the components to the backend. In particular:

Member	Main focus
Samuele Scherini	Focus on component development and CSS styling
Alessandro Sironi	Focus on page implementation and database queries
Elisa Tognini	Focus on pages and components implementation

Documentation

Chosen Theme

We have decided to build a website for a venture capitalist company called "coreventure - Core Investments". The aim of the company is to help tech startups and companies, that are willing

to innovate, to achieve success and unlock their full potential.

Technologies and Motivations

- For the *server side* development we have used **Nuxt Nitro API endpoints**.
- For the *hosting* our application we have used **Vercel** hosting service.
- Since the website is data-driven, for the *database implementation* we have used **Supabase** SQL database, based on **PostgreSQL**.
- Since visibility for a company's website is critical, we decided to use the **Server-Side-Rendering (SSR)** option to render pages on the server side. This choice allows web crawlers to better index the website, resulting in better ranking in search results, CSR has been discarded immediatly also because the website does not require an high level of interactivity.

We have also decided to use **TypeScript** instead of **Javascript** to write more robust and bug-free code, also the use of **TypeScript** allow us to define types related to the Database entities and it granted us a better developer experience enabling auto-completion, for example, in the props definitions and assignment of the components.

Project Structure

In this section it is described the structure of the project, from the pages directory to the components list.

Pages and links

The following list contains all pages present in the website with the url and a short description.

Page	Url	Description
Home	`/`	Homepage of the website, contains various links to the other pages and summarized informations.
About	`/about`	Contains the description of the main activities of the company and what it can do for the startups.
Team	`/persons`	Contains the list of employees working for the company.
Person	`/persons/[id]`	Display the information regarding a specific employee and the projects he/she has supervised or worked on.
Top Projects	`/projects/top`	Display the most relevant projects the company helped grow.
All Projects	`/projects`	Contains the list of all the projects the company has worked on.
Project	`/projects/[id]`	Display the information regarding a specific project.

Page	Url	Description
Areas	`/areas`	ontains the list of all the areas related to the projects the company has decided to invest in.
Area	`/areas/[id]`	Display the information regarding a specific area, allow the user to directly navigate to the related projects.
Contacts	`/contacts`	This page contains the contact information and give the user a mail form.

Components

We have developed the components of the website trying to maximize their *reusability*. This, in addition to allowing *modularity*, also has benefits on the *usability* of the system. In fact, the user views familiar components while browsing the website.

Some components however are used only once but we still decided to create a component in order to clean the code in the main pages.

In this section each component is described in details with all the props and emits (if present).

Please keep in mind that the Complex types, for example Area, Employee ecc. are directly related to the entities in the database (this is because the types were generated using supabase CLI commands see more).

- ***AreaCard***

  this is the card used to display an Area in the "Areas" page. Each card is a link to the single area page.

Prop	Type	isRequired	Note
`area`	`Area`	`yes`	The area object to be displayed in the card

- **AreaTag**

  this component serves as a link to an area or to the most relevant projects page from a project

Prop	Type	isRequired	Note
`area`	`Area`	`no`	The linked area object

Prop	Type	isRequired	Note
`is_most_relevant`	`boolean`	`no`	If it is true the link goes to /projects/top

- **Carousel**

  This component shows the ProjectCards of all projects that are marked as most relevant in a slideshow. It is possible to move through the slideshow by clicking the arrows or by dragging on mobile.

  This component, for now, is used to show the most relevant projects.

Prop	Type	isRequired	Note
`projects`	`Project[]`	`yes`	The list of projects to be displayed

- **ComplexParagraph**

  This is a generic paragraph that can display an image next to a block of text. The image is optional and can be positioned on the left (default value), or on the right of the text.

Prop	Type	isRequired	Note
`image_url`	`string`	`no`	The image to be displayed
`is_image_right`	`boolean`	`no`	The position of the image (if true image is on the right)
`image_alt`	`string`	`no`	The alt attribute of the image
`width`	`number`	`no`	The width of the image
`height`	`number`	`no`	The height of the image

  To display the actual text the component uses the **slot** tag.

- **EmployeeCard**

  This is the card used to display an Employee in the "Team" page, each card is a link to the single Employee page

Prop	Type	isRequired	Note
`employee`	`Employee`	`yes`	The employee object to be displayed in the card

---

- **Footer**

  This is the default footer present on the bottom of all pages, it is a fixed component and it is responsive for mobile devices.

- **GenericButton**

  This is the general button used in the website, it accepts a value (text to display on the button) and a boolean called altStyle which defines which color scheme the button should have.
  It also accepts an emit called func, which is the function invoked when clicking the button. Please note that the function logic is obviously defined in the parent of this component to achieve reusability.

Prop	Type	isRequired	Note
`value`	`string`	`yes`	The string displayed in the button
`altStyle`	`boolean`	`yes`	It represent the color scheme of the button

Emits:

Emit	Type	Note
`func`	`function`	The function to be called when the button is clicked

---

- **GenericLink**

  This is the general link used in the main sections of the website to navigate between pages, it accepts a name (text to display on the link), a boolean called altStyle which defines which color scheme the link-button should have and it also accepts an url which is the url we go to when clicking the link-button.

Prop	Type	isRequired	Note

Prop	Type	isRequired	Note
`name`	`string`	`yes`	The string displayed on the link
`altStyle`	`boolean`	`yes`	It represent the color scheme of the link button
`url`	`string`	`yes`	The url link

- **InfoCard**

  This is the card used to display single pieces of information in various pages as attention grabbing accents.

Prop	Type	isRequired	Note
`icon_locator`	`string`	`yes`	The icon to be displayed in the card
`title`	`string`	`yes`	Title displayed in the card

  To display the actual text the component uses the **slot** tag.

- **LoaderComponent**

  This is the spinner used in various pages to show the users when data is still fetching from the database.

- **Navbar**

  This is the default navigation bar menu present on top of all pages, it is a fixed component (with sticky attribute) and it is responsive for mobile devices.

- **NextPrev**

  Displays a next and previous button that allows the user to navigate between areas or projects, dynamically. Displays two buttons and a text that adapts to the context.

Prop	Type	isRequired	Note
`label`	`string`	`no`	The string displayed between the arrows

Emits:

Emit	Type	Note
`onPrev`	`function`	The function to be called when the "prev" arrow is clicked
`onNext`	`function`	The function to be called when the "next" arrow is clicked

---

- **OverlappingParagraph**

  This is the card used to display single pieces of information in various pages as attention grabbing accents.

Prop	Type	isRequired	Note
`image_url`	`string`	`yes`	background image of the paragraph
`image_alt`	`string`	`yes`	Alternative text of the image

  To display the actual text the component uses the **slot** tag.

- **ProjectCard**

  This is the card used to display a single Project in the "Projects" page, as well as ineach individual *Person* page. Each card is a link to the project's page

Prop	Type	isRequired	Note
`project`	`Project`	`yes`	project object to be displayed in the card

- **SocialButton**

  This component accepts an icon and a url link, it can also accept a text to be displayed on the right of the icon.

Prop	Type	isRequired	Note
`icon`	`string`	`yes`	icon representing the link to the social media
`url`	`string`	`yes`	link url string
`text`	`string`	`no`	text that can be br displayed next to the icon

---

Modules and libraries

Here is the list of the official nuxt 3 modules we have decided to import.

Module	Description
supabase	A wrapper around supabase-js to enable usage and integration of supabase DB within Nuxt.
google-fonts	This module is used to import google fonts inside the application
nuxt-icon	This module gives us access to modern and standard icons
image-edge	This library is used to better display images in the website, from a performance and accessibility point of view
pinia	This library allow us to easily implement stores in the website

Extra functionalities implemented

The main extra functionality we have decided to implement is the possibility to filter the projects by area (or most relevant) in the projects page.

This functionality introduces a certain level of complexity in the application and in the following section it is described what we did to achieve the final result.

Filtering the projects

The filter functionality alone is not too complex, the best choice, from a design point of view, was to create "filter chips" that when toggled apply a filter on the projects array.

The biggest problem was that, when the user clicked on the "discover projects" link in the Area page the desired behaviour is to redirect to the Projects page with the apllied filter.

After a long internal discussion we decided that the best way to integrate the filter with the whole application, achieving the desired behaviour, was to implement a global store containing the filtered area name and that can be set when clicking the "discover projects" link.

This solution also allow us to "remember" the filtering choice made by the user, who will se the filter applied even after re-visiting the page.

Best Practices Compliance

All the principles of **accessibility** have been followed correctly. Here are some of the measures adopted:

- Short descriptions of non-text content (such as images) are provided
- Labels for form controls, input, and other user interface components are provided
- Headings, lists, input fields, and content structures are marked-up properly
- Foreground and background color combinations provide sufficient contrast-ratio
- Images of text are avoided
- Pages have clear titles and are organized using descriptive section headings
- There is more than one way to navigate between relevant pages
- The purpose of a link is evident
- Buttons, links, and other active components are large enough to make them easier to activate by touch

From the **SEO guidelines** point of view we developed with attention to SEO score. In addition to what is listed above, all pages have been provided with appropriate meta tags.

> All the pages of the website have been analyzed with the following tools:
>
> - **WAVE extension**: no errors or contrast error have been detected
> - **Google Lighthouse**: all Performance, Best Practices and Accessibility tests have been successfully passed with an average between 90 and 100.
>
> WARNING: Please notice that the performance score produced by lightouse can sometimes be lower than 90 because of the vercel infrastructure that we cannot control (in localhost the performance score observed when analyzing the build are always higher than 90).