POLITECNICO
MILANO 1863

Technology Report
for
Centro MiLA Website

Course Name
Hypermedia Applications (Web and Multimedia)

Professors
[redacted]; [redacted]; [redacted]

Running Prototype
https://hyp-technology-project.vercel.app/

Authors
Ivan [redacted] [redacted]
Annachiara [redacted] [redacted]
Lorenzo [redacted] [redacted]
Matteo [redacted] [redacted]

Milan, Italy
17 July, 2024

Technology Report

1. Introduction

1. Group name: team MiLA
2. Group composition:
    a. Annachiara [redacted] [redacted]
    b. Ivan [redacted] [redacted]
    c. Lorenzo [redacted] [redacted]
    d. Matteo [redacted] [redacted]
3. Link website: https://hyp-technology-project.vercel.app/
4. Link GitHub repository: https://github.com/LorenzoProSky/HYP-Technology-Project

2. Teamwork split

The work regarding the design and development of the entirety of the "Hypermedia Applications (Web and Multimedia)" project was dived as follows among the group members, from here after identified by their name and surname (with a " * " is denoted a relatively small contribution to that particular task):

Group coordination: Lorenzo [redacted]
GitHub repository management – Design document: Matteo [redacted]
GitHub repository management – Technology project: Lorenzo [redacted]

Content Design - C-IDM scheme and Content Tables: Matteo [redacted]
Navigation Design - Abstract pages: Ivan [redacted]
Low fidelity wireframes: Annachiara [redacted] & Lorenzo [redacted]
Design system: Annachiara [redacted]
High fidelity wireframes: Annachiara [redacted] & Lorenzo [redacted] *
Presentation - Commented high fidelity wireframes: Annachiara [redacted]
DB Design - ER diagram: Lorenzo [redacted]

Website creation and base structure: Lorenzo [redacted]
Group accounts creation: Lorenzo [redacted]
Webserver setup: Matteo [redacted]
Database setup: Lorenzo [redacted]
Information collection and creation: Annachiara [redacted] & Ivan [redacted]

Design system implementation: Ivan [redacted]
Components design: Annachiara [redacted]
Components implementation: Lorenzo [redacted] & Ivan [redacted] *
Footer and header design: Annachiara [redacted]
Footer and header implementation: Ivan [redacted] & Matteo [redacted]
Page structure: Ivan [redacted]

Single page implementation:

- Homepage: Ivan [redacted]
- About Us: Ivan [redacted]
- Locations: Ivan [redacted]

- Our People: Lorenzo [redacted]
- Person: Matteo [redacted]
- Our Activities: Lorenzo [redacted]
- Our Services: Lorenzo [redacted]
- Service: Ivan [redacted]
- Our Projects: Lorenzo [redacted]
- Project: Matteo [redacted]
- What You Can Do: Annachiara [redacted]
- Volunteering: Ivan [redacted]
- Donate: Annachiara [redacted]
- Contact Us: Ivan [redacted]

Database structure implementation and database population: Lorenzo [redacted]
Webserver and APIs implementation: Matteo [redacted]
APIs integration: Matteo [redacted] & Ivan [redacted]

Chatbot design: Annachiara [redacted]
Chatbot implementation: Ivan [redacted] & Matteo [redacted]

SEO: Matteo [redacted]
Accessibility: Matteo [redacted]

Design document writing: Annachiara [redacted]
Technology document writing: Lorenzo [redacted] & Matteo [redacted]*
Latex documents management: Matteo [redacted]

Final site and documents review: Ivan [redacted], Annachiara [redacted] Lorenzo [redacted] Matteo [redacted]

The work was fairly split among the team members based on individual competences and skills, interests and availability, in such a way to enable as much parallelism and synergy as possible, while exploiting personal knowledge and providing the possibility to members to explore new topics of their interest.

3. Documentation

Project description:

MiLA is a support centre for women and children victim of violence. It offers comprehensive range of services and initiatives, with the goal of improving women' situations and empowering them. In fact, MiLA not only focuses on providing immediate assistance for those in difficult situations, but it also provides tools and opportunities for women to concretely improve their life and achieve lasting happiness.
With the developed website (described in this document), MiLA and its team aims at widening the reach of their words and their presence in the Milan area. In addition, the platform will provide a better and more unified experience for women and children in need, enhancing support and accessibility.

Application architecture and Hosting services used:

The chosen software application architecture is the well-established Three-Tier Architecture, since it offers great separation of concerns, scalability, maintainability and security. Following this principle the website was divided into frontend (presentation tier), backend application server (application tier) and backend database (data tier).

The MiLA website (frontend) was built utilizing the Nuxt/Vue framework with Typescript (due to static typing). This choice stems from several benefits provided by the framework: ease of use and flexibility, modular architecture, performance, scalability and maintainability.
The chosen platform for the hosting of the webserver (application server) was Vercel, while for the hosting of the database was Supabase. The reasons behind these choices are the following: ease of use and readiness of the products, developer experience and accessibility, cost-effectiveness and ease of integration.
The project was developed in a shared fashion utilizing Git over GitHub.

Following the principles of the Three-Tier Architecture, the frontend presents content from MiLA centres, by requesting images and other assets to the backend server via API calls. Subsequently, the server fetches the corresponding information from the database, which stores all relevant data for the association. For the implementation of the chatbot, the site makes specific API calls to the ChatGPT service of OpenAI, utilizing their APIs to retrieve responses from a virtual assistant tailored on the website's needs.

The chosen rendering mode for the website is SSR (Server-Side Rendering), due to the following considerations:

- the website is primarily informative and not highly interactive (CSR would not be a good fit);
- the website needs to load images from the database via the webserver (SSG would need client-side fetching, which might harm the user experience by affecting the perceived performance).

So, SSR has been chosen because:

- it reduces client-side complexity (the client does not need to handle fetching and rendering images after the initial page load);
- it offers great performance and real-time data, and given that pages will not be requested many times there introduced latency will be minimal;
- due to the manageable number of estimated concurrent request, it will not lead to scalability issues.

Project structure:

The organization of the website's pages and links takes place inside the "pages" folder, in which the pages are subsequently categorized in their respective folders based on the navigation items.
The entirety of the website is divided into:

1. "Home" page, the main page of the website.
2. "About Us" section, which offers various information regarding the MiLA association and its centres, more specifically its history and mission, its team and its locations.
3. "Activities" section, which provides a comprehensive list of all the services and projects that the MiLA association offers. "Service" are core activities conducted within MiLA centres, designed to provide meaningful and essential support to women, while "projects" are initiatives typically carried out outside the MiLA centres, focusing on spreading information and raising awareness. A project can be present or past, if it is no longer offered.

4. “What You Can Do” section, which explains how people can help the MiLA association to further improve its services and make a meaningful impact, via volunteering and donation.
5. “Contact Us” page, which offers visitors all the needed information to contact MiLA centres.

The “home” page and all root pages of a section are named index.vue, while the others follow a classical naming convention based on their topic. In addition, the pages regarding a single topic (person, service, project) have the name [id].vue, because there is a single page for each of them, which then gets populated by the information of the single specific topic (person, service, project), also including the id.

Summary of Routes:

1. Home page: /
2. About Us: /about-us
    a. Our Locations: /about-us/locations
    b. Our People: /about-us/people
        i. Single Person: /about-us/people/[person_id]
3. Our Activities: /activities
    a. Our Projects: /activities/projects
        i. Single Project: /activities/projects/[project_id]
    b. Our Services: /activities/services
        i. Single Service: /activities/services/[service_id]
4. What You Can Do: /what-you-can-do
    a. Volunteering: /what-you-can-do/volunteering
    b. Donate: /what-you-can-do/donate
5. Contact Us: /contacts

The implementation of the structure, styling and functionalities of header, with navigation menu, and footer is written in the default.vue file in the “layouts” folder. This was done since header and footer are common to all pages and this approach promotes consistency, ease of page creation and separation.

The design system and other general styling properties are written in the general.css file in the “assets/styles” folder. This was done to ensure consistency through the whole website, ease of page creation and separation.
Website’s images are all fetched from the database, apart from the ones that are present in multiple static pages. These few images are stored locally in the “assets/images” folder, due to their nature and use.

The remaining parts of the project follow the classical structure and meaning of Nuxt projects. To conclude, the “components” folder will be discussed in detail in the following chapter.

Webserver endpoints:
Notice: the notation “:<value>” expresses a dynamic parameter of the URL.

Webserver API endpoint:	**Description:**	**Used for (frontend endpoint):**
/people	Retrieve data for all the people	/about-us/people
/services	Retrieve data for all the services	/activities/services

/projects	Retrieve data for all the projects	/activities/projects
/people/:person_id	Retrieve data for specific person with id=person_id	/about-us/people/[id]
/services/:service_id	Retrieve data for a specific service with id=service_id	/activities/services/[id]
/projects/:project_id	Retrieve data for a specific project with id=project_id	/activities/projects/[id]

The webserver is deployed on Vercel and accessible at the URL: https://hyp-project-backend.vercel.app/.

Components:

The website's components are entirely custom-made using only HTML, CSS, and TypeScript. These components are categorized into four folders: buttons, cards, chatbot and header under the "components" folder of the Nuxt project. The "components/global" folder is separately discussed in the imported modules chapter.

Components rely on NuxtLink for navigation, while internal reactive states for styling changes are based on user interaction (hover, active, focused).
All props for the components are required, if not differently specified. All types of the props are String, if not differently specified.

Buttons

1. MainButton.vue: primary action button with different lengths and navigation capabilities.
   Props:
   - buttonText: text displayed on the button.
   - buttonLength: button's width (short, medium, long).
   - isDisabled (Boolean): disabled status, default false.
   - to: URL or route to navigate to when clicked.
2. SecondaryButton.vue: secondary action button with different lengths and navigation capabilities.
   Props:
   - buttonText: text displayed on the button.
   - buttonLength: button's width (short, medium, long, smartphone, tablet).
   - isDisabled (Boolean): disabled status, default false.
   - to: URL or route to navigate to when clicked (optional).
3. ExitButton.vue: button designed for emergency use, allowing immediate exit from the website, taking the user to google.com. This feature provides the possibility and reassurance to women victims of violence to not be caught by their abusing partners, while seeking help.
4. BackwardButton.vue: navigation button (with backward icon) used to backtrack to the previous page.
   Props:
   - buttonText: text displayed on the button.
   - isDisabled (Boolean): disabled status, default false.
   - to: URL or route to navigate to when clicked.
5. ForwardButton.vue: navigation button (with forward icon) used to highlight the presence of a hyperlink.
   Props:

    - buttonText: text displayed on the button.
    - isDisabled (Boolean): disabled status, default false.
    - to: URL or route to navigate to when clicked.
6. TextButton.vue: simple text-only button.
   Props:
    - buttonText: text displayed on the button.
    - isDisabled (Boolean): disabled status, default false.
    - to: URL or route to navigate to when clicked.
7. LinkButton.vue: simple link button, to highlight the presence of a hyperlink.
   Props:
    - buttonText: text displayed on the button.
    - isDisabled (Boolean): disabled status, default false.
    - to: URL or route to navigate to when clicked.
8. ChatbotButton.vue: specialized button for initiating chatbot interaction.
   Emits:
    - chatbotButtonClicked: used to notify the parent component (the generic layout default.vue) that the chatbot panel must be shown or hidden.

Cards

1. DiscoverCard.vue: card for showcasing a brief preview of a page's content and concept, helping users understand the destination and purpose of the page they are about to visit.
   Props:
    - title: title of the card, incorporating the page's name.
    - text: small description of the card, providing insight into the page's content.
    - to: URL or route to navigate to when clicked.
2. ActivityCard.vue: card for showcasing a brief preview of an activity's content and concept, used for both projects and services.
   Props:
    - imageSrc: source image URL of the card.
    - title: name of the activity.
    - text: short description of the activity.
    - to: URL or route to navigate to when clicked.
3. ManagerCard.vue: card for showcasing which projects or services a person is manager of.
   Props:
    - managerName: name of the manager.
    - type: type of managed activity (project or service).
    - text (array of String): names of managed activities.
    - to (array of String): URLs or routes to navigate to when clicked.
4. PersonCard.vue: card for showcasing a brief description of a team member of MiLA.
   Props:
    - imageSrc: source image URL of the card.
    - name: name of the person.
    - job: job title of the person.
    - text (optional): short description of the person.
    - to: URL or route to navigate to when clicked.
5. ProjectCard.vue: card for showcasing a brief preview of a project's content and concept, with additional useful project-related information.
   Props:
    - imageSrc: source image URL of the card.

   - title: name of the project.
   - text: short description of the project.
   - when: time-related information about the project.
   - where: location-related information about the project.
   - to: URL or route to navigate to when clicked.
   - type: type of project (present or past).
6. ServiceCard.vue: card for showcasing a brief preview of a service’s content and concept, with additional useful service-related information.
   Props:
   - imageSrc: source image URL of the card.
   - title: name of the service.
   - text: short description of the service.
   - when (array of String): time schedules in which the service is offered in the respective centre.
   - where (array of String): centre locations in which the service is offered.
   - to: URL or route to navigate to when clicked.
7. MapCardBig.vue: card for showcasing all MiLA centre-related information.
   Props:
   - imageSrc: source image URL of the card’s map.
   - name: centre’s name.
   - address: centre’s address.
   - email: email contact information.
   - phone: phone contact information.
   - hours: centre’s opening hours.
   - to: URL or route to navigate to when clicked.
8. MapCardSmall.vue: card for showcasing a subset of MiLA centre-related information.
   Props:
   - title: centre’s name.
   - address: centre’s address.
   - email: email contact information.
   - phone: phone contact information.
   - to: URL or route to navigate to when clicked.

Chatbot

1. ChatAgent.vue: component used to wrap and display a single message returned by the chatbot agent in the chatbot panel.
   Props:
   - text: content of the message to be displayed by the chat agent.
2. ChatUser.vue: component used to wrap and display a single message inputted by the user in the chatbot panel.
   Props:
   - text: content of the message entered by the user.
3. Chat.vue: chatbot panel in which the conversation between the user and the agent takes place. This component takes care of displaying the list of messages exchanged between the user and the agent using the ChatUser.vue and ChatAgent.vue components.

Header
This directory contains the building blocks for the header creation. These components are all used in the default.vue layout under the layout folder to assemble the header.

1. HeaderDropDownMenu.vue: drop down menu component that is shown when the corresponding high-level link in the header is hovered by the mouse (or focused by the keyboard).
   Props:
   - linkNames (Array): names of the links to be displayed inside the dropdown menu.
   - to (Array): actual links corresponding to the linkNames to redirect the user on click.

   Emits:
   - focusedLink: used to notify the parent component (header in default.vue) when one the links inside the dropdown menu is focused (either hovered by the mouse or focused by the keyboard).
   - blurLink: used to notify the parent component (header in default.vue) when *none* of the links inside the dropdown menu is focused (either hovered by the mouse or focused by the keyboard).
2. MobileHeaderOpenCloseButton.vue: button to toggle the visibility of the mobile version for the header (only displayed when targeting a mobile device with touch support).
   Props:
   - controlsId: id of the mobile header component. This prop is passed from the parent component to communicate to the MobileHeaderOpenCloseButton.vue what is the id for the mobile header whose visibility is controlled by the button itself. It is fundamental for accessibility enhancement as the button can specify the aria-controls attribute over the mobile header.
   - iconName ('MenuIcon' | 'MobileExitIcon'): name of the Icon to be displayed for the button. The icon changes based on the visibility of the associated mobile header and it is managed by the parent component through this prop.

   Emits:
   - toggleMobileMenu: notifies the parent component (MobileHeader.vue) that the button has been clicked so that the mobile header can change visibility status.
3. MobileHeader.vue: complete layout of the mobile version for the header. This will only be shown on devices supporting a touch screen interaction with the user (substituting in this way the desktop version). It does not use props or emits.
4. MobileHeaderPlusMinusButton.vue: button to toggle the visibility of submenus inside the MobileHeader.vue component (only for the mobile version of the header).
   Props:
   - controlsId: string to identify the id of the submenu whose visibility is controlled by the button. It enhances accessibility of the website as the plus/minus button can declare the aria-controls attribute over the specified id of the submenu.

   Emits:
   - toggleSubmenu: notifies the parent component (MobileHeader.vue) when the button has been clicked and therefore the corresponding submenu must be shown or hidden.

Imported modules:

- 'nuxt-icon' https://nuxt.com/modules/icon: module to easily and efficiently integrate and display icons, ensuring a consistent and visually appealing user interface.
  Regarding this package, the folder "components/global/" must be utilized to store svg vue components of icons, as clearly stated in the documentation "Note that MyComponent needs

to be inside components/global/ folder…”. It was deemed the best alternative to utilize this methodology to handle icons, given the small number needed and chosen by the team (downloading the entire library of icons would have been overkill). In addition, storing icons locally avoids the need to fetch them from Iconify.

- ‘openai’ https://www.npmjs.com/package/openai: module to simplify the integration of the chatbot on the website. It is used to create new threads of communications with the virtual assistant used as chatbot whenever a user opens the chat.

Extra functionalities:

Complex services relationship
As a necessary pre-requisite explanation, a person is manager of zero or more services/projects, while a service/project has a single manager, this has requested and denoted with “responsible” in the specifications.
In addition, a service is provided by one or more people and offered in one or more centre locations. This creates a complex relationship between the provided service, the people providing the service and the locations offering the service, which can be solved via a ternary relationship. As a consequence, a person can be manager in the above-described fashion, and be a provider of zero or more services, even different ones from the ones they manage.
Concluding, these characteristics enable the feature of having the same service offered in different locations, provided by different people in each one.

Navigation button
In “Our People”, “Our Projects”, “Our Services” pages there is the “navigation button” used for pagination of the various elements. If there is only a single page to be displayed, then the button is not present. The number showing the total number of pages and the active one are dynamic, based on the number of elements per page and the total number of elements to be displayed. When the user goes to a new page they are immediately taken to the correct section of the page.
In “Single Person”, “Single Project”, “Single Service” pages there is also the “navigation button” but implemented with two slight modifications: it leads to the respective “Our” pages, instead of displaying the number of pages and it leads to the previous/next person, project or service page, instead of paginations.
The “navigation button”, even though present in multiple pages, was not implemented as a component due to technical limitations and time constraints, stemming from the strong intertwining of the button with the page itself.

Toggle project button
Projects can be present or past and to let the user choose which category of projects to visualize in “Our Projects” page, it was implemented the “toggle project button”. It switches from present projects to past projects and vice versa. It also works seamlessly in conjunction with the “navigation button” in the page, ensuring the correct behaviour of the page, paginations and buttons.
In addition, to highlight this difference among projects, the present projects’ cover images are in colour, while past projects’ cover images are in black and white.
The “toggle project button” was not implemented as a component due to it being used in a single page and it being strongly intertwined with the page itself.

Multiple version header (for desktop devices and mobile devices with touch-screen support)
The header of the website is designed in two different versions. The desktop version uses dropdown menus whose visibility is toggled by hover and focus events on the high-level links that are fixed on the navigation bar (About Us, Our Activities, What You Can Do).
This approach is not ideal for mobile devices that have touch screen support. Therefore, a completely

different version is provided with a touch-based navigation. As the page is loaded, the touch capabilities of the client device are inspected, and the appropriate header is displayed.

Chatbot:

To start off, the chatbot integrated on the website is backed up by a virtual assistant hosted on the OpenAI developer API. The virtual assistant has been designed thinking about the role of an advisor and confidant that can provide rapid psychological and emotional support to the women landing on our website. In the design phase, the prompts have been written down in such a way that the virtual assistant will reply in a calm, compassionate and kind way to the user, providing a word of advice when asked for. The replies are not too long as the chat space is limited.
From the implementation point of view, the chatbot integration is carried out in the Chat.vue component, in which a new thread for the user is created whenever the user clicks on the button to open the chat. The streaming API offered by OpenAI is then used to send the user request to the assistant and retrieve the response.
The official OpenAI documentation has been used to implement the chatbot:
https://platform.openai.com/docs/introduction

Accessibility:

Accessibility is another important aspect of the website as all people with different kinds of impairments should be able to access the content. Different accessibility best practices have been followed during the development process to enhance the experience of people with visual, mobility or cognitive impairments (the website doesn't show any audio or video so also people with hearing impairments have the possibility to fully access the content).

People with visual impairments can have sight problems of different severity. For people with minor problems, the website colours and background images have been chosen with a sufficient contrast compared to the text colour, making the content easy to read. The spacing between the various parts of a page also provide a good visibility enhancement as the content is not cluttered.
Furthermore, for people that are blind (or severely visually impaired), the website supports full navigation with the keyboard, combined with ARIA attributes that are used throughout the whole website to populate the accessibility tree. This is the basic API for screen reader technologies that allows visually impaired people to read the content of the website aloud as the keyboard focus moves on the screen.
Good examples of these best practices can be found in the header, footer and chat areas of the website, where special care has been taken since they are the fundamental navigation blocks of the website (and are present in all the other pages).
The keyboard navigation and screen reader behaviour have been tested using ChromeVox to verify that the information provided was accessible enough.

For people with mobility impairments, the website has a convenient distancing of all the most important buttons and navigation links, allowing for an easy touch or click experience.

As for people with cognitive impairments, the content of the website is minimalistic, essential and free of distractions. Also, the concepts explained are exposed in a very clear and simple language.

Finally, responsive design must be mentioned among the accessibility best practices as accessibility is a broader concept that doesn't involve only people with disabilities. The website is fully accessible from any device, either mobile or desktop and with touch screen support or not.

To test the accessibility of the website, the Lighthouse software integrated in Google Chrome has been employed, reaching high scores in all pages.
The main documentation used to study accessibility best practices is from the MDN website: https://developer.mozilla.org/en-US/docs/Web/Accessibility.

SEO:

Search Engine Optimisation is fundamental to make the website rank high on search engines like Google. SEO best practices have been used on the website to enhance the probability of ranking higher in some user research on the web.
First off, Server-Side Rendering has been enabled on the Nuxt server to build the HTML pages on the server side and provide them ready when required. This reduces the search engine crawler's latency in downloading rendering the page, resulting in possibly highest scores for the website.
The default head configuration for the whole website can be found in the file nuxt.config.ts. The head tags define the most important properties that have to be propagated throughout the website. More specifically:

- The language.
- The set of characters.
- The viewport configuration.
- The robots meta tag (to explicitly tell the crawler to index the page) is set to index, follow.

Furthermore, all the pages of the website are provided with:

- A title, remarking the content and the most important keywords for the page.
- A description briefly explaining the content of the page.
- A set of keywords, which are related to the content of the page and target specific words that users might type in the search engine to look for the information of the page. The keywords are provided both in English and in Italian as the centre is imagined to be settled in Milan (so most visitors will be Italian and use Italian words to search the content).

On top of everything, the most relevant thing the website offers for SEO is the originality and quality of the content. The information displayed on the website is unique and well-presented in a catchy way to provide the best user experience and therefore the highest ranks on search engines.

The decisions taken for SEO are driven by the guidelines provided by Google on their website, clearly stating what is helpful and what is useless when accounting for SEO: https://developers.google.com/search/docs/fundamentals/seo-starter-guide.
Testing for SEO scores has been thoroughly carried out using Lighthouse on Google Chrome.