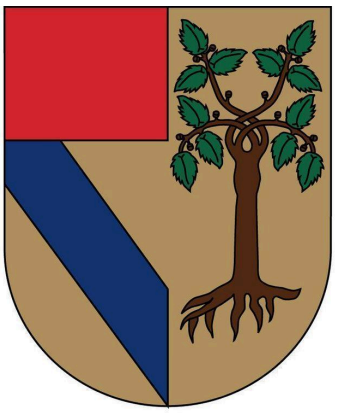

Development team log

Carlos J. Zepeda
Martín G. Torres
Andrés G. Gómez

School of engineering

Universidad Panamericana De Guadalajara

DAW2: Web Development

Eng. Gabriel C. Cortés

26 November 2024

The Prototype

28/08/24

Tasks Completed:

- Creation of the project's basic structure: The main structure of the project was designed and organized, including the fundamental sections and divisions. This provided a solid foundation for future development and allowed us to visualize how the final product would look.
- Initial CSS styling: Basic styles were implemented using CSS to give a consistent appearance to the different sections of the project.

Challenges Faced:

- **Organization of the site structure:**
    - **Problem:** Creating the site structure was particularly challenging, as we needed to organize the elements in a way that was both appropriate and functional for end-users.
    - **Solution:** To overcome this difficulty, the team carefully studied the original website on which our project is based. This research allowed us to establish a reasonable and coherent structure.
- **Dropdown menu:**
    - **Problem:** During the implementation of the dropdown menu, difficulties arose in getting it to work correctly.
    - **Solution:** After reviewing relevant documentation, consulting with ChatGPT, and watching several tutorials on YouTube, we successfully implemented a functional dropdown menu.
- **Overlaying an image with text:**
    - **Problem:** A challenge emerged when trying to overlay an image with text while applying a blur effect to the image.
    - **Solution:** Similar to the dropdown menu, this was resolved by reviewing documentation, consulting with ChatGPT, and watching videos on YouTube. The blur effect was successfully applied to the image while keeping the text clear and legible.

29/08/24

Tasks Completed:

- First log update.

Challenges Faced:

- **Writing the logs:**

- **Problem:** We had never created this type of documentation in a project before, so we were unsure of how to structure it.
- **Solution:** After conducting research online, we found several high-quality formats that we decided to use.

30/08/24

Tasks Completed

- **Image carousel implementation**.

Challenges Faced

- **Bootstrap carousel**:
  - **Problem**: Attempted to simplify the process with Bootstrap but encountered limitations.
  - **Solution**: Adjusted the approach and created a custom carousel.

03/09/24

Tasks Completed

- **Palette color change**.
- **Download button creation**.
- **Navbar implementation**.

Challenges Faced

- **Finding an agreed color scheme**: Required extensive discussions to finalize.
- **Fixed download button issues**: Resolved challenges related to its behavior.

04/09/24

Task Failed

- **Bootstrap for image deployment**: Attempt to use Bootstrap to display additional information for images about Lake Chapala.

Challenges Faced

- **Bootstrap code adjustments**:
  - **Problem**: Difficulty adapting Bootstrap components to images.
  - **Solution**: Attempted code modifications but faced time constraints.

Migration to Node.js

07/10/24

Tasks Completed

- **Integration of Node.js and Express**: Migrated the backend to Node.js and Express, ensuring a functional routing system and preparing for EJS and MongoDB integration.

Challenges Faced

- **Routing adjustments**:
    - **Problem**: Conflicts arose during routing setup.
    - **Solution**: Reviewed documentation and resolved conflicts through structured debugging.

08/10/24

Tasks Completed

- **Added MongoDB**: Enabled efficient management of large datasets.
- **Bootstrap enhancement**: Improved layout design while resolving compatibility issues with older Bootstrap versions.
- **Implemented pagination**: Handled 12,000 records efficiently.

Challenges Faced

- **Bootstrap version conflicts**:
    - **Solution**: Reverted to compatible Bootstrap components and adjusted styling.
- **Table pagination**:
    - **Solution**: Researched and implemented best practices for handling large datasets.

09/10/24

Tasks Completed

- **Log updates**: Reflected all tasks and challenges in detailed logs.

Challenges Faced

- **Delayed updates**:
    - **Solution**: Compiled accurate information by reflecting on past tasks.

24/10/24

Tasks Completed

- **Improved table design**: Enhanced responsiveness and download button validation.

Challenges Faced

- **MongoDB connection issue**:
  - **Solution**: Used a personal MongoDB Atlas database for functionality testing.

Migration to React

15/11/24

Tasks Completed:

- **Initial Migration to React:**
  - Migrated the `index.js` and its related content from EJS to React.
  - Implemented a functional structure for React with dynamic rendering of components using JSON files.
- **Navbar Implementation:**
  - Created a fully functional Navbar using React-Bootstrap.
  - Integrated dropdown menus and navigation links that dynamically scroll to specific sections of the page.
- **Dynamic Rendering with JSON:**
  - Converted hardcoded data in the original EJS files into structured JSON.
  - Ensured all dynamic content in the index page (e.g., sections, titles, and images) is rendered dynamically using reusable components.
- **Main Sections:**
  - Developed reusable section components (`Section`, `MainDescriptionSection`, and `GalleryCarousel`) for the index page.
  - Successfully rendered the content of the Home page, including dynamic image galleries and descriptions.
- **Basic CSS Styling:**
  - Adapted key CSS styles for React compatibility.
  - Styled the Navbar, carousel, and main sections to resemble the original design.

Challenges Faced:

- **Dynamic Data Management:**
  - **Problem:** Handling dynamic data while keeping components reusable and organized.

    - **Solution:** Structured the project to load content dynamically via JSON files, ensuring all components rely on props for data input.
- **Bootstrap Integration:**
    - **Problem:** Ensuring React-Bootstrap components worked seamlessly with existing styles and functionality.
    - **Solution:** Verified correct imports for React-Bootstrap components and adjusted custom styles to avoid conflicts.
- **Navigation and Routing:**
    - **Problem:** Setting up smooth navigation between sections of the page using React-Router.
    - **Solution:** Successfully implemented routing for internal links, ensuring smooth scrolling and correct targeting.
- **Carousel Customization:**
    - **Problem:** Styling the gallery carousel to match the original layout while keeping captions responsive.
    - **Solution:** Applied custom CSS to adjust image scaling, caption placement, and overall responsiveness.

16/11/24

Tasks Completed:

- **Migration of Data and Graphs Sections to React:**
    - Migrated the `Datos` and `Graficas` sections from EJS to React, ensuring functionality and responsiveness.
    - Created dynamic, reusable components for rendering tables (`Datos`) and graph sections (`Graficas`).
- **Implementation of Dynamic Data Handling:**
    - Imported large datasets directly into the `Datos` component using JSON.
    - Ensured `Graficas` renders sections dynamically based on `graficasData.json`, maintaining consistency with the original design.
- **Compact Pagination in Data Table:**
    - Designed a compact paginator for the `Datos` section to prevent overflow on pages with large datasets.
    - Included "First," "Previous," "Next," and "Last" buttons for enhanced navigation.
- **Enhanced User Navigation:**
    - Improved hash-based navigation to ensure redirection to the root (`/`) and scrolling to the correct section dynamically.
    - Adjusted the Navbar to handle transitions between sections and routes seamlessly.
- **Graph Section Refinement:**
    - Added titles, descriptions, and images to each graph section dynamically.

    - Styled the graph sections with captions and descriptions, ensuring consistency with the application's overall design.
- **Custom CSS Adjustments:**
    - Adapted existing styles for the `Datos` and `Graficas` sections to React.
    - Ensured compatibility with Bootstrap's responsive design.

Challenges Faced:

- **Dynamic Table and Pagination:**
    - **Problem:** Handling large datasets dynamically while maintaining performance.
    - **Solution:** Implemented a paginated approach with a compact range, allowing users to navigate easily without overwhelming the UI.
- **Navigation Between Routes:**
    - **Problem:** Smooth scrolling to specific sections after switching routes was inconsistent.
    - **Solution:** Refined the hash navigation logic to first redirect to the root (`/`) and then scroll to the desired section using the native `href` behavior.
- **JSON-based Component Rendering:**
    - **Problem:** Adapting hardcoded content into JSON files while preserving the structure and reusability.
    - **Solution:** Designed reusable components that accept data through props and dynamically render content.
- **Styling for Graph Captions and Tables:**
    - **Problem:** Adjusting captions and table styles to match the original layout while ensuring responsiveness.
    - **Solution:** Applied media queries and custom CSS to enhance the presentation of small images and wide tables.

17/11/24

Tasks Completed:

- **Implementation of Download Button:**
    - Added a functional button to allow users to download data in `.csv` format.
    - Utilized a helper function to convert JSON data into CSV strings, ensuring compatibility with diverse datasets.
    - Integrated `FileSaver.js` to handle the generation and download of the files.
    - Ensured that the downloaded files are named dynamically based on the dataset.
- **Pilot Tests for Graphical Visualizations:**
    - Set up initial tests for dynamic graphs using Chart.js.

    - Rendered bar and line charts to visualize sample datasets extracted from the application state.
    - Implemented a basic layout to accommodate future graph types and adjustments.
- **Custom Styling for Download Button and Graph Sections:**
    - Designed and styled the download button to match the application's UI/UX.
    - Enhanced graph sections with dynamic titles, captions, and tooltips to improve user interaction.

Challenges Faced:

- **File Encoding Issues:**
    - **Problem:** The downloaded `.csv` files initially had encoding issues with special characters.
    - **Solution:** Implemented UTF-8 BOM in the generated CSV strings to ensure proper rendering of special characters.
- **Graph Rendering Performance:**
    - **Problem:** Rendering delays were observed when datasets exceeded 10,000 data points.
    - **Solution:** Limited the initial dataset size during testing and began researching performance optimization techniques.
- **Dynamic CSV Header Generation:**
    - **Problem:** Ensuring that CSV headers were generated dynamically from the data structure.
    - **Solution:** Iterated over the dataset keys to build headers programmatically.

20/11/24

Tasks Completed:

- **Integration of Third-Party Weather API:**
    - Integrated the OpenWeatherMap API to fetch real-time weather data for the Lake Chapala region.
    - Configured API calls to dynamically retrieve weather conditions for the selected location (Ajijic as the default).
    - Implemented secure storage for the API key using environment variables (`REACT_APP_API_WEATHER_KEY`).
    - Displayed weather details, including temperature and a weather icon.
- **Dynamic Weather Display:**
    - Designed a new React component (`API.jsx`) for fetching and rendering weather information.
    - Structured the component to handle asynchronous API requests with `fetch`.
    - Added error handling to gracefully display messages when the API call fails or returns invalid data.

- **Styling and Layout Adjustments:**
    - Styled the weather display section with a responsive card design to ensure usability on various devices.
    - Incorporated visual elements, such as icons from OpenWeatherMap, to improve user engagement.

Challenges Faced:

- **API Authorization in Deployment:**
    - **Problem:** API calls returned "unauthorized" during deployment on Netlify due to incorrect environment variable configuration.
    - **Solution:** Updated the `REACT_APP_API_WEATHER_KEY` in Netlify's environment settings and verified its accessibility in the deployed build. Cleared the cache and re-deployed the application.
- **Error Handling for API Failures:**
    - **Problem:** Inconsistent API responses caused unexpected rendering issues in the weather component.
    - **Solution:** Implemented comprehensive error handling to check the API response status and display fallback content if necessary.
- **Responsive Weather Section:**
    - **Problem:** The initial layout of the weather display did not scale well on smaller screens.
    - **Solution:** Adjusted the CSS using media queries to ensure proper alignment and readability across all device sizes.

21/11/24

Tasks Completed:

- **Error Handling Page:**
    - Created an error handling page to display a descriptive error message whenever an error occurs.
    - Added a test button to create custom errors and activate the error page for testing purposes.
- **React Graphs:**
    - Developed graphs that retrieve their data from the backend and the database.
    - Displayed graphical visualizations of data from the last year.

Challenges Faced:

- **Connection Between Backend and Graphs Component:**
    - **Problem:** Encountered many errors when creating functions in the context and experienced long waiting times when fetching data from MongoDB and passing it to the graphs component.

    - **Solution:** Pulled the data through the backend, then processed it in the context so the graphs could access and render the data efficiently.
- **Display of the Graphs:**
    - **Problem:** The graphs were not fully visible within the window.
    - **Solution:** Used CSS `@media` queries to ensure the graphs responded to window size, making them visible regardless of the device.

27/11/24

Tasks Completed:

- **Styling:**
    - Applied final styling changes to ensure coherence across the entire page.
    - Made various color scheme modifications and implemented effects on the new elements added during this delivery.

Challenges Faced:

- **Compatibility:**
    - **Problem:** Compatibility issues arose with the React app, specifically related to the Navbar style and colors in certain areas.
    - **Solution:** Adjusted CSS styles to align with React components and fixed inconsistencies in the Navbar appearance.