Dev Log - Web Application Development

Date: 05/10/2025

Project/Module: Portafolio - 04 Bootstrap Showcase

Goals for the Documented Session

- Learn how to implement specific Bootstrap elements in HTML.
- Use the Grid System to create a more organized and responsive layout.
- Build a well-structured HTML document.

Task Completed

I worked on Portafolio for exercise 04, which consisted of organizing and improving my personal webpage using Bootstrap. The main objective was to strengthen my HTML and CSS skills while implementing Bootstrap to optimize time. In the end. In the end, I achieved a good result, although it took me quite a long time to finish.

Use of Gen AI

I used GitHub Copilot for small details that I found difficult to implement, especially when working with different Bootstrap versions.

One of my main issues was that I followed Bootstrap documentation without realizing that some examples were from version 4 and others from version 5. This caused my webpage to stop working properly because some components weren't compatible. AI helped me understand this mistake about mixed versions. After that, I modified the classes and made sure all my HTML used Bootstrap 5 consistently.

Example Prompt:
"Explain me why the code for the navbar and accordion doesn't work on my personal website.'"

Output:

- You had mixed attributes (p. ej. `data-toggle` in the navbar) — in Bootstrap 5 you must use `data-bs-...`.
- You loaded both Bootstrap 4 and 5 at the same time, which breaks dynamic components like the accordion.
- Bootstrap 4 uses `data-toggle`, while Bootstrap uses `data-bs-toggle`. Since you already use `data-bs-toggle`, you should only keep Bootstrap 5.
- You also had duplicate Bootstrap scripts - remove one of them.

Edits: I corrected all class names from the old version so that everything worked properly.

Reflection: I learned the importance of checking and choosing the correct Bootstrap version before coding, to make sure everything is consistent and functional.

Investigation

Some specific doubts I googled to make it fast my research and to continue coding.

Prompt: *"Should the navbar go inside the header?'"*

Output: I found this answer on Stack Overflow:

https://es.stackoverflow.com/questions/234948/d%C3%B3nde-es-correcto-anidar-la-etiqueta-nav-nav-para-crear-un-men%C3%BA-en-html5

Edits: I placed the Bootstrap navbar code inside the <header> tag.

Reflection: I learned that the <nav> element can be inside <header> to create a more semantic and organized structure.

Prompt: *"How can I link to a specific ID in another HTML file'"*

I asked this because I wanted to add city sections on my destination page and link them from the navbar to make it more dynamic.

Output:

To create a link from one HTML file to a specific element (ID) in another HTML file, use the formal:

`filename.html#element_id` inside the `<a>` tag's `href` attribute.

Edits: I used this structure with my file names and IDs.

Reflection: I learned how to make internal links between HTML files using IDs, which helps improve website navigation.

What I Learned

I learned how to correctly apply Bootstrap 5 components and avoid mixing versions. I also improved my understanding of semantic HTML structure and internal linking between pages.

Challenges

The biggest challenge today was dealing with the Bootstrap version. It took me a lot of time to realize why my code wasn't working, but now I understand hoy to manage versions better.

Resources used

- https://es.stackoverflow.com/questions/234948/d%C3%B3nde-es-correcto-anidar-la-etiqueta-nav-nav-para-crear-un-men%C3%BA-en-html5
- https://getbootstrap.com/docs/4.5/components/alerts/

Next Steps

- Double.check Bootstrap versions before starting any project.
- Learn more strategies to code faster and understand Bootstrap´s class system better, since the names can be long and confusing at first.

Personal Reflection

Even though this exercise took me a lot of time, I feel proud of what I accomplished. I learned to pay more attention to details, especially with library versions and documentation. Now I feel a little bit more confident working with Bootstrap and building better-structured web pages.

Date: 09/10/2025

Project/Module: Portafolio - 04 Bootstrap Showcase

Goals for the Documented Session

- Learn how to implement specific Bootstrap elements in HTML.

- Learn how to make a webpage look good on different screen sizes.

Task Completed

I continued working on exercise 04 from my Portafolio. Today, my goal was to finish the final and review the instructions carefully before submitting the project.

Use of Gen AI

I used GitHub Copilot for small details that I found difficult to implement.

One of my main issues was making my website look good in different screen sizes without breaking its structure or design.

Example Prompt 1:
“I have this navbar from Bootstrap, but when I make the screen smaller, the dropdown for 'Favorite Spots' doesn't appear. There should be an icon with three horizontal lines, and the button is there, but it's not visible. Could you explain why?”

Output:

The issue you described —where the “hamburger icon” is there but not visible— is very common in Bootstrap.

This happens because the icon doesn't appear automatically unless the navbar has a predefined color class (like `navbar-light` or `navbar-dark`) and a background color that matches it.

Edits: I modified the classes and added the correct color and background.

Reflection: I learned the importance of using the correct classes in Bootstrap to make components work properly.

Example Prompt 2: *“Explain this instruction: The result of this exercise must behave in a responsive way, that means that it should look good in a screen or in a phone or tablet.’”*

Output: My code has responsive behavior mainly because of this tag:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This allows the layout to adjust to different screen sizes (without it, the website would look the same on mobile and desktop).

```
<div class="container">
   <div class="row">
      <div class="col-md-8">...</div>
      <div class="col-md-4">...</div>
   </div>
</div>
```

This makes sections automatically adjust into one or multiple columns depending on the screen width.

Edits: I checked that my structure followed this format in all sections.

Reflection: I learned how the grid system and the viewport tag make a website responsive without needing extra CSS.

Example Prompt 3: *“Can I use @media to adjust screen sizes (@media <type> <feature> { ... })?’”*

Output: You don't necessarily have to write your own `@media` rules because Bootstrap already includes them in its framework.

For example, when you write:

```
<div class="col-md-6">
```

Bootstrap translates it into something like:

```
@media (min-width: 768px) {
  .col-md-6 {
    flex: 0 0 50%;
    max-width: 50%;
  }
}
```

That means it automatically takes up 100% width on small screens and 50% on medium or large screens.

You only need to use your own `@media` queries if you want to customize specific details.

Edits: I added a few `@media` rules for specific elements like titles and images to improve their appearance.

Reflection: I learned that Bootstrap already includes responsive features, but I can still use `@media` for small custom adjustments.

Investigation

Some specific doubts I Googled to make my research faster and keep coding.

Prompt: *“What size should a favicon image be?’”*

Output: A favicon should be a square image. The most common size is 16x16 pixels, but it's recommended to use larger sizes like 48x48 or 512x512 pixels for high-resolution devices.

Edits: I used a 300x300 image for my favicon.

Reflection: I learned that favicon sizes can vary, but it's best to use a high-resolution image so it looks good on all devices.

What I Learned

I learned that Bootstrap automatically adapts the layout for different screen sizes. You only need to use custom `@media` rules for small visual adjustments. I also learned the importance of checking how each element behaves when resizing the browser window.

Challenges

The biggest challenge today was understanding how to make my page look good in different screen sizes without breaking the design.

Resources used

- https://getbootstrap.com/docs/4.5/components/alerts/

Next Steps

- Identify which elements need extra adjustments with `@media`.
- Keep practicing responsive design to make sure my websites look good on all devices.

Personal Reflection

Today I finally understood how Bootstrap makes a website responsive automatically. At first, it was confusing to see how the layout changed, but now I know it's part of Bootstrap's grid system.

Date: 11/10/2025

Project/Module: Portafolio - 05 JS

Goals for the Documented Session

- Learn how to program with JavaScript for web pages

- Use basic JavaScript functions to create interactivity in a webpage.

Task Completed

I worked on Portafolio exercise 05. The main objective was to learn JavaScript and understand how this language allows us to make our webpages interactive.

Use of Gen AI

I used GitHub Copilot for small details that I found difficult to implement, especially because I had never worked with JS before.

Mostly, I used AI to help me understand the exercises. For example, in the Pig Latin exercise, I didn't understand what I was supposed to do with the words. AI helped me break down the problem and understand the logic behind it. I also asked AI to explain some of the code given by the teacher. For instance, in the clock exercise, I didn't understand most of the code at first, but AI explained it step by step, which helped me complete the exercise successfully.

I also used it for specific bugs that I didn't understand.

Example Prompt 01:
"This is how my folders are organized. Why can't my HTML file find my JS file?."
Output:

- Because your HTML is inside the `html` folder and your JS file is in a sibling folder called `js`, you need to go up one level before entering `js`.
- `..` means "go up one folder."

Edits: I corrected the `src` of my `<script>` and added `..` to go up a folder.

Reflection: I learned how important it is to organize folders and file paths correctly to avoid compilation or linking problems.

Example Prompt 02:
"What does this error mean: `fibonacci.js:38 Uncaught ReferenceError: Cannot access 'value' before initialization`*?"*
Output:

- It means you’re using the variable `value` before declaring it.

Edits: I moved my variable declaration to the top of the code.

Reflection: I learned the importance of declaring variables before using them in JavaScript.

Example Prompt 03:
“Why don't the clock hands move? They stay static.’”
Output:

- You have two script tags in your HTML. The problem is that when you run the code, the `drawClock` function isn’t defined yet because the `clock.js` file loads afterward.
- So `setInterval(drawClock, 1000)` can’t run correctly.

Edits: I moved `<script src="../js/clock.js" charset="UTF-8"></script>` to the beginning.

Reflection: I learned the importance of script order in HTML to make sure functions are available when the page loads.

Investigation

Some specific doubts I Googled to make my research faster and keep coding.

Prompt: *“What is the Fibonacci sequence?’”*

I remembered learning about it in my advanced programming class, so I also checked my old notes and then looked it up online.

Output: I found this answer on this website:

https://www.mathsisfun.com/numbers/fibonacci-sequence.html

Edits: I used the formula I found to build my Fibonacci exercise.

Reflection: I refreshed my memory of the Fibonacci sequence and how to implement it in code.

Prompt: *“Fibonacci sequence calculator”*

I asked this to check if my results were correct.

Output:

I found this website:

https://www.calculatorsoup.com/calculators/discretemathematics/fibonacci-calculator.php

Edits: I didn’t edit my code, just used this to verify my outputs.

Reflection: I learned to validate my solutions with external tools.

Prompt: *“how to convert an input to a number in JS?”*

Output:

Use `parseInt()` for integers, `parseFloat()` for decimals, or the unary `+` or `Number()` function.

Edits: I used the `Number()` function in my code.

Reflection: This helped me a lot when I was still getting used to JS and learning about useful functions like `isNaN()` and `isFinite()` from https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/isNaN

Prompt: *“symbols < > in HTML”*

I asked this because I was getting errors when using the symbols directly.

Output:

I found this website page:

https://ascii.cl/es/codigos-html.htm

Edits: I replaced the symbols with the proper HTML entity codes.

Reflection: I learned that some characters must be escaped to display correctly in HTML.

Prompt: *“Clock formulas”*

I asked this to better understand how a clock works mathematically.

Output:

I found this webpage:

https://www.splashlearn.com/math-vocabulary/clock-angle-formula

Edits: I applied these formulas to my JavaScript code for the clock exercise.

Reflection: I learned how math concepts can be used directly in programming.

What I Learned

I learn many new things about JavaScript:

- How to correctly link JS files to HTML.
- How to declare and use variables.
- How to use basic functions and interact with the DOM.
- How math logic can be implemented in JavaScript to build dynamic elements like a clock or Fibonacci sequence.

Challenges

The biggest challenge was that I didn't know any JavaScript at the beginning, so everything felt difficult. I spent a lot of time understanding the code the teacher gave us and figuring out how to adapt it to make the exercises work properly.

Resources used

- https://www.mathsisfun.com/numbers/fibonacci-sequence.html
- https://www.calculatorsoup.com/calculators/discretemathematics/fibonacci-calculator.php

- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/isNaN
- https://ascii.cl/es/codigos-html.htm
- https://www.splashlearn.com/math-vocabulary/clock-angle-formula

Next Steps

- Improve my JavaScript coding speed.
- Optimize my logic and learn to solve problems faster.
- Keep practicing to feel more confident working with functions and events in JS.

Personal Reflection

Even though this was one of the hardest modules so far, I learned a lot about JavaScript. At first, it was frustrating not understanding the code, but little by little I started to make sense of it and see how everything connects.

Date: 11/10/2025

Project/Module: Portafolio - 06 Personal Website (JS), 07 Basic Node

Goals for the Documented Session

- Understand JavaScript more deeply.

- Learn Node.js — what it is and how to use it.

Task Completed

I first worked on Exercise 06 of the portafolio, which was to create a function that adds the information from the form fields into the schedule table I already had. This exercise helped me understand how to make simple modifications to the DOM tree using JavaScript to see how classic JS works before moving on to Node.js.

Then I continued with Exercise 07, where I installed Node.js and completed the first exercises to learn how to create a simple Node script, execute it, and use libraries inside de node.

Use of Gen AI

I used GitHub Copilot only for small details that I found difficult to implement, specifically for Exercise 06 of the portafolio.

Example Prompt 01:

"schedule.js:44 The specified value "" does not conform to the required format. The format is "#rrggbb" where rr, gg, bb are two-digit hexadecimal numbers.

It shows this warning in the console — should I worry about it?'"

Output:

- The AI explained that this warning appears because the color input expects a valid color format (like `#ff0000`), but when it's empty, its value is `""`.
  So, when the browser tries to read it, it says it's not a valid color — but this is only a **warning**, not an error.
- It doesn't affect the code, the form, or its behavior.

Edits: I left my code as it was, since it didn't affect anything.

Reflection: It's important to pay attention to the error and warning shown in the console to understand if they are critical or not.

Example Prompt 02:

"I read in the documentation about let and var, but I didn't understand their differences.'"

Output:

- The AI explained that `let` is safer, more modern, and prevents overwriting errors.
- It also mentioned `const`, which is used when a variable's value should not change.

Edits: I replaced all my `var` declarations with `let` to make my code cleaner and more consistent.

Reflection: I learned that using `let` and `const` helps prevent bugs and makes the code easier to understand and maintain.

Investigation

To speed up my progress, I googled a few specific questions:

Prompt: *"How to add a new row in a table using JS?'"*

Output: I found helpful examples on Stack Overflow and MDN:

https://es.stackoverflow.com/questions/192336/agregar-filas-a-tabla-con-javascript-dom

https://developer.mozilla.org/es/docs/Web/API/HTMLTableElement/insertRow

Edits: I tested both methods and chose the one that worked best for my table.

Reflection: I understood better how the DOM works and how each element can be dynamically created or modified.

Prompt: *"How to insert a cell and fill it in JS?'"*

Output: I found this documentation:

https://developer.mozilla.org/en-US/docs/Web/API/HTMLTableRowElement/insertCell

Edits: Based on that, I successfully implemented the logic to insert each form field's data into its corresponding table cell.

Reflection: I learned that understanding how each DOM element works helps a lot when creating interactive and dynamic web content.

Prompt: *"How to close a Bootstrap modal with JavaScript?"*

Output: From Stack Overflow:

https://stackoverflow.com/questions/69170870/bootstrap-5-close-modal-with-vanilla-javascript

It mentioned using this snippet:$('#nameModal').modal('hide');

Edits: I adapted that line using my own modal's ID, and it worked correctly.

Reflection: I learned that Bootstrap modals can be controlled not only with data attributes but also directly through JS for more flexibility.

What I Learned

I improved my understanding of JavaScript and started to feel more confident using it. The most valuable part of today´s session was learning Node.js. I didn't fully understand it during class, but my dad helped explain it to me and guided me through the first exercises.

After that, I was able to complete the rest on my own and understand how Node executes scripts and manages modules.

Challenges

At first, I didn't understand Node.js at all, and it made me nervous about the exercise. Also, in Exercise 6, the logic behind adding a new row to the table was a bit confusing. However, after reading documentation and testing different examples, I finally understood it and made it work successfully.

Resources used

- https://es.stackoverflow.com/questions/192336/agregar-filas-a-tabla-con-javascript-dom
- https://developer.mozilla.org/es/docs/Web/API/HTMLTableElement/insertRow
- https://developer.mozilla.org/en-US/docs/Web/API/HTMLTableRowElement/insertCell
- https://stackoverflow.com/questions/69170870/bootstrap-5-close-modal-with-vanilla-javascript

Next Steps

- Keep improving my JavaScript logic and syntax

- Practice with more complex Node.js exercises to feel more confident
- Optimize my time and try to complete exercises faster.

Personal Reflection

This session helped me realize how much I’ve improved since the beginning of the portfolio. I’m still learning, but now I feel more confident experimenting with JavaScript and Node.js. Even though I found some parts difficult, I was proud that I managed to solve them with research and practice.

Date: 12/10/2025

Project/Module: Portafolio - 08 Basic Express

Goals for the Documented Session

- Learn what Express is and how to use it.
- Understand the importance of an Express server.
- Learn about the body-parser library and its role in handling data.

Task Completed

I worked on Exercise 8 of the portfolio, which focused on learning how to serve multiple static pages and handle basic interaction through form parameters.

The task was to:

- Create an **Express server**.

- Make sure it was listening on port `3000`.

- Use the `body-parser` library to get the values submitted through the form.

This was my first time working with Express, so the goal was to understand how everything connects — from the server setup to receiving data from the frontend.

Use of Gen AI

I used GitHub Copilot only for small details that I found difficult to implement, especially for the folder and file structure in the project.

After finishing the exercise, I also asked the AI to help me create a `README.md` file because I didn't really know what to include, and my dad told me these files are very important once you start working with Node.js projects.

Example Prompt 01:
"I get this error in the terminal when I run `node server.js`: Error: Cannot find module.'"

Output:

- The AI told me to make sure I was in the correct folder — the one where `server.js` is actually located.

Edits: I moved everything into a single, organized folder.

Reflection: It is important to have very organized the files in your project.

Example Prompt 02:
"Why is my BMI calculation wrong? Can you explain, please?'"
I had written `Math.round(bmi)`, andI was not getting the exact result of the other bmi.

Output:

The AI explained that:

- `Math.round(bmi)` only rounds to the nearest whole number.
- If I want **two decimals**, I need to multiply by 100 before rounding, and then divide by 100 afterward.

Example:

```
const bmiRounded = Math.round(bmi * 100) / 100;
```

Edits: I replaced my formula with the correct version so the BMI calculation would display with two decimal places.

Reflection: I understood better how rounding works in JavaScript and why small details like this can affect precision.

Example Prompt 03:
“Does my code meet the requirements?’”

I wasn’t sure if I was doing the exercise correctly. The AI suggested adding:

```
app.use(express.json());
```

I asked what it was for.

Output:

The AI explained that:

- `express.json()` allows Express to understand JSON data sent in the request body.
- Without it, Express wouldn’t know how to read or parse JSON.
- This makes the content available in `req.body`.

Edits: I added the function to my code.

Reflection: I understood why this line was important, but then I was confused about why we still needed `body-parser`.

So I asked about the difference between `express.json()` and `body-parser`, and the AI explained that:

- `body-parser` is an older library but still works.
- Newer versions of Express already include the same functionality.

In this exercise, since the form was sending data as `x-www-form-urlencoded`, we needed:

```
app.use(express.urlencoded({ extended: false }));
```

- JSON parsing is for different types of requests, like those from JavaScript fetch calls.

This explanation helped me understand the difference between the two and update my code accordingly.

Investigation

To speed up my progress, I googled a few specific questions:

Prompt: *"How to round a number in JS?""*

Output: I found helpful examples on MDN:

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Math/round

Edits: I used `Math.round()` with the multiply-and-divide method for decimals.

Prompt: *"What is body-parser Express""*

Output: I found documentation on:

https://expressjs.com/en/resources/middleware/body-parser.html

https://dev.to/eclecticcoding/express-and-body-parser-khf

Reflection: This helped me understand better why body-parser is needed when handling form data.

What I Learned

I learned how to:

- Set up an **Express server** and understand why it runs on `localhost:3000`.

  This allows us to create a local environment where the server listens to requests and responds accordingly.

- Use `body-parser` and `express.urlencoded` to **retrieve and process data** from HTML forms.
- Understand the difference between `express.json()` and `body-parser`.
- Organize my folders and files properly to avoid “module not found” errors.

Challenges

I struggled with **folder and file hierarchy**. I started the project the day before, and when I reopened it, the terminal kept giving me errors because my file paths weren’t correct. I learned how important it is to keep a clear project structure.

At first, I didn’t understand how Express worked or how to use `body-parser`.

Debugging my BMI calculation took longer than expected because of a small detail (`Math.round`).

Resources used

- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Math/round
- https://expressjs.com/en/resources/middleware/body-parser.html
- https://dev.to/eclecticcoding/express-and-body-parser-khf

Next Steps

- Keep practicing with **Express** to understand routing and middleware better.
- Build more projects locally using `localhost` to get more comfortable with server-side logic.
- Practice debugging and organizing file structures to avoid confusion later.

Personal Reflection

At first, working with Express felt confusing and overwhelming — especially since it was my first time setting up a server. But after experimenting, asking questions, and reading documentation, I started to understand how everything connects.

I still have a lot to learn, but I'm proud of how much progress I've made in just one session.

Date: 13/10/2025

Project/Module: Portafolio - 09 Express Personal API

Goals for the Documented Session

- Practice Express more deeply
- Learn to test different responses
- Understand how to handle and return different error types in Express.

Task Completed

I developed Exercise 9 from the portfolio, which tests various Express server responses and basic error handling. The project uses HTML templates for dynamic pages and supports multiple routes and methods (GET, POST, PUT, DELETE) to manage greetings and a simple TODO list. It also serves static files, handles task updates and deletions, redirects URLs, and returns JSON responses for Postman testing.

Use of Gen AI

I used GitHub Copilot for parts I found difficult or unclear. I also used AI to generate a `README.md` file after completing the project, since I didn't know exactly what to include. My dad mentioned that READMEs are very important when working with Node.js projects.

Example Prompt 01:

“porque esto no funciona: `if (name !== "undefined" && name.trim() !== "")`*”*

Output:

- Because you're comparing `name` to the string `"undefined"`, not the actual `undefined` value in JavaScript. If `name` is not defined, calling `name.trim()` will throw an error.

Edits: I changed it to: if (typeof name !== 'undefined')

Example Prompt 02: *“What this means: err.status .’”*

I didn't understand the error handling section and the AI suggested that I use error.status 500 but I didn't understand so I started looking at the documentation and understood that section better.

Output:

- `err.status` is a property often used in frameworks like Express.js to indicate the HTTP status code of an error. It’s not part of the native `Error` object but can be added manually to specify how the server should respond.

Edits: res.status(err.status || 500);

Investigation

To speed up my progress, I googled a few specific questions:

Prompt: *“como quitar los espacios en blanco en una entrada js’”*

Output: I found helpful examples on MDN:

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String/trim

Edits: I used `trim()`.

Prompt: *“errors in js to create a new handler that outputs the error message in the index page..”*”

Output: I found this documentation on MDN and Stack Overflow:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling

https://stackoverflow.com/questions/6970475/get-all-javascript-errors-on-page-javascript-error-handling

Prompt: *“error status”*”

Output: I found documentation on HTTP status codes:

https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Status/500

What I Learned

aprendí usar …. (usar las instrucciones)

Challenges

I learned how to use different Express methods like GET, POST, PUT, and DELETE, and how to manage responses, handle basic errors, and test endpoints using Postman. I also understood better how error handling works in Express.

Resources used

- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String/trim
- https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Status/500
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling
- https://stackoverflow.com/questions/6970475/get-all-javascript-errors-on-page-javascript-error-handling

Next Steps

- Keep practicing Express and localhost
- Improve error handling
- Work on time management

Personal Reflection

This exercise helped me understand Express more deeply. Even though error handling was tricky, I learned how to fix issues step by step. I'm starting to feel more confident with Express.

Date: 14/10/2025

Project/Module: Portafolio - 10 Express EJS

Goals for the Documented Session

- Learn how to use EJS to create dynamic content with Express
- Understand how to organize project structure.
- Practice from handling with GET/POST methods.

Task Completed

I worked on the portfolio exercise 10, which focused on integrating EJS with Express to create dynamic pages. I set up the server to handle static content, templates, JSON, and forms. Then I implemented basic interactions with a login form, created reusable partials for the header and footer, and built different templates (`test.ejs`, `home.ejs`, `post.ejs`) to display greetings, show blog posts, and allow users to create, view, and edit posts. I also ensured that logged-in user data persisted between pages.

Use of Gen AI

I used GitHub Copilot to clarify instructions I didn’t understand and to help me fix file structure issues.

Example Prompt 01:
“Please, explain me what is wrong with my structure'"

I had issues again with the organization of my project files. The AI recommended creating a **“views”** folder, which is typically used to store `.ejs` files, and a **“public”** folder, where all the publicly accessible files are usually placed.

Output:

- EJS projects usually have a `views` folder for `.ejs` files and a `public` folder for static files.

Edits: Reorganized my folders accordingly.

Reflection: It is important to have very organized files.

Example Prompt 02:
“Why don't we use the header tag in the ejs file?’”

When I combined the **header** and **footer** EJS files, I encountered an error, and they didn’t merge correctly, so I asked about this issue.

Output:

- The idea of EJS is to avoid repeating code. Instead of writing the same `<header>` in every file, we use `<%- include("partials/header") %>` to import a shared partial.

Edits: Created and included `header.ejs` and `footer.ejs`.

Example Prompt 03: *“I got an error in the file path...’”*

Output:

- EJS automatically looks inside the `views` folder when including partials. You don't need to write `/views/` or the file extension.

Edits: Fixed the path in my includes.

Investigation

To speed up my progress, I googled a few specific questions:

Prompt: *"Structure of EJS Nodejs?'"*

Output: Found structure examples on Dev.to:

https://dev.to/victrexx2002/introduction-to-ejs-a-guide-to-building-dynamic-web-applications-2737

Edits: Used it to correctly apply `<% %>` syntax.

Reflection: Is very similar to HTML but using this syntax in every line.

Challenges

The hardest part was understanding how partials and includes worked and how to properly set up the folder structure.

Resources used

- https://dev.to/victrexx2002/introduction-to-ejs-a-guide-to-building-dynamic-web-applications-2737

Next Steps

- Keep practicing Express and EJS
- Improve file structure organization and partial usage

Personal Reflection

At first, EJS was confusing, especially the structure and includes. But after understanding how partials and routing work, I felt much more confident with dynamic rendering.

Date: 15/10/2025

Project/Module: Portafolio - 11 Express & 3rd-party API

Goals for the Documented Session

- Learn how to use third-party APIs
- Practice API calls and responses in Express
- Handle errors properly when working with APIs

Task Completed

I worked on portfolio exercise 11, which focused on integrating an external weather API into an Express project. I connected the front end to the server, built the API call using `https.get`, and displayed the temperature and weather description dynamically with the corresponding icon. I also handled possible API errors and made sure users could return to the home page easily.

Use of Gen AI

I used GitHub Copilot only for the parts I found difficult to implement and to help me understand the instructions. I also asked it to generate the **README.md** file after finishing my exercise, since my dad told me that it’s important to include this file when working with an API.

Example Prompt 01:
“How do I put my API key in my project?’”

Output:

- Avoid uploading your API key to Git and store it locally.
- Create a **.env** file and write your key inside it.

Edits: I started doing it that way, but I ran into many problems and errors on my page. For example, it kept saying that my key didn't exist or wasn't being read, even though it worked before. The AI helped me understand these errors and how to fix them, but the issue persisted. Since it was taking me too long to get the code running, I decided to leave the key in my **server.js** file. I know that's bad practice, but I couldn't solve the error in time.

Reflection: I have to investigate more with more time how to implement in my server the env file.

Example Prompt 02:
"When I enter the city, the temperature appears like this: Â°C ."'

Output:

- This happens due to encoding issues. Adding `res.setHeader("Content-Type", "text/html; charset=utf-8");` fixes it.

Edits: Added the header.

Investigation

To speed up my progress, I googled a few specific questions:

Prompt: *"apis"'*

Read API documentation on OpenWeather.

Output: documentation:

https://openweathermap.org/current

Edits: Used it to understand the structure of the response.

Challenges

The biggest difficulty was dealing with the `.env` file and API key errors. I also needed to understand how to parse and display API data properly.

Resources used

- https://openweathermap.org/current

Next Steps

- Keep practicing APIs and error handling

- Learn to manage API keys more securely

Personal Reflection

This exercise gave me a better understanding of how APIs work with Express. Although the `.env` setup was challenging, I learned a lot about debugging and how to handle external data effectively.