UNIVERSIDAD
Panamericana

Beta version: Development log

Andrea Carolina Hinojosa Trujillo

Aemi Rosas Sánchez

José Eduardo Moreno Paredes

Jetzuvely Del Carmen González Gutiérrez

Universidad Panamericana

Web Application Development

Gabriel Castillo Cortes and Bernardo Moya de la Mora

October 21st, 2025

Development Log

Team

Day 1: Distribution of tasks

We started by distributing the tasks to every team member. Eduardo was going to make one map window, Aemi the other map window, Andrea the recommendations window and Jetzuvely the main window.

30/09/2025 Investigation about the best map tool

Andrea and Aemi

We started by exploring different mapping tools. Our first idea was to use Google Maps because it is widely known and offers many features. However, after researching its documentation, we found out that it requires payment once a certain usage limit is reached. Since we needed a completely free option for our project, we decided to discard that idea.

Next, we discovered OpenLayers, an open-source mapping library. We tested it, and we successfully displayed the map on the page. Although we still need to test how it works with an PAI, this was a big step forward in the development process.

Challenges: It was finding a mapping tool that was both functional and free. Many of the most popular options required a paid plan, which limited our choices.

Solution: We decided to use OpenLayer, but we needed to test it with an API.

02/10/2025 API Research

Andrea and Aemi

We spent the session researching APIs that provide information about endangered marine species. At first, we realized that there weren´t many available options, and we also did not fully understand how to access or obtain an API. To keep track of our research, we wrote down several possible options that could be useful to our project.

Since we hadn't learned how to properly get and use APIs yet, we asked for help. Aemi´s dad explained to us how the process works and guided us on where to look. Thanks to that, we were able to choose the IUCN Red List API as the main data source for our endangered species web page.

Challenges: It was not knowing how to find or access APIs. This limited our progress at the beginning of today's research.

Solution: We decided to ask someone with more experience in the topic to guide us. With that help, we understood the process better and successfully selected a suitable API.

09/10/2025 - 16/10/2025 Implementation of endangered map

Andrea and Aemi

Goals for the documented session
The goal of this session was to improve our map integration by making it more efficient and user-friendly. We also aimed to filter the API data to make it more manageable and relevant to our project´s focus.

Task Completed
We decided to switch the map, since Leaflet is simpler to use and works directly with JavaScript. This change allowed us to handle map functionality more easily and with better performance.

While testing the API, we noticed that it contained a massive amount of information, but the data wasn't well-organized. Because of this, the map took more than five minutes to load when displaying all endangered animals worldwide. To solve this, we decided to limit our map to filter the data to include only species that are in critical conservation status and only in Mexico. This reduced the dataset to just 53 animals, which improved the loading time significantly.

At first, we built the map in a separate project to make testing easier. The challenge came when we needed to integrate it with our Endangered Page, but after some adjustments, we managed to connect everything correctly.

Finally, we used to help extract and assign the correct latitude and longitude values for each species, allowing us to accurately plot them on the map.

Challenges

- The original API contained too much unorganized information.
- The map took several minutes to load due to the large dataset.
- Integrating the map project into the main page required additional effort.

Solutions

We reduced the dataset by focusing on critically endangered species in Mexico and switched to Leaflet for simpler and faster map rendering. We also used AI tools to efficiently locate the geographic coordinates for each species.

Use of AI

Prompt: “Give me the latitude and longitude coordinates to each endangered animal”
We needed to find a way to assign accurate latitude and longitude coordinates to each endangered animal so they could be correctly displayed on the map. Doing it manually would have been too time-consuming.

Output: The tool gave us the coordinates.

Edits: After generating the data with AI, we organized it to fit the correct format for our JS map. We then verified that each point matched the right species before adding them to the map.
Reflection: Using AI saved us a lot of time for the coordinates and made the process more efficient.

Resources used
Leaflet
IUCN Red List API
The help of Aemi´s dad for specific doubts of API implementation

Next Steps
Improve the visual design and interactivity.

What I learned
I learned how to use Leaflet to create dynamic maps with JavaScript and how to filter large API datasets to improve performance. I also learned how to integrate a separate project into a larger web page and how AI can be used to automate data organization.

10/10/25 – Research on Different Methods for Designing Maps

Jetzuvely Gonzalez

During this day, I spent time researching different ways to design maps for fishing activity. I explored several tools and platforms to understand their capabilities. One of the tools I found most visually appealing was Figma, which allows creating map designs using plugins like Map Maker. I realized that Figma is mainly a design tool and doesn't offer full interactivity, but it is very useful for planning the layout, colors, and overall style of maps.

I also looked into more interactive approaches, like using Leaflet and JSON (GeoJSON) data. Eduardo explained how JSON can be used to structure map data and make it interactive. This approach seemed more practical for real applications because it allows the map to display actual data dynamically without relying on platforms like Google Earth, which can be more complex to use.

Throughout this research, I compared the advantages and disadvantages of each method. Figma is great for aesthetics and visual planning, while JSON + Leaflet provides functionality and real data interaction. I wrote notes about how each tool could be used depending on whether the goal is visualization or interactivity and the use of API, I think is more difficult.
Later, I reviewed all the information I collected and reflected on how these methods could influence the final design of our maps. I focused on aspects such as title placement, layout, and responsiveness for mobile devices. I noted that some designs look good on a computer but overlap or lose clarity on smaller screens.

Although I did not implement any maps myself, this research helped me understand the workflow of designing maps, from aesthetic planning in Figma to functional implementation using JSON and Leaflet. I also considered how structured data can improve map usability and help communicate important information clearly. Overall, this process improved my knowledge of map design methods and allowed me to evaluate different options for future projects.

18/10/25 Fishing Activity Page Enhancement with Leaflet and EJS

José Eduardo Moreno Paredes
Project / Module: Catch the Change - Web Development

Goals for the documented session
  Complete map integration and refactor common components
  transform header and footer into reusable EJS components
  transform htmls into EJS templates
  add map functionality to fishing activity page
  add necessary routes in app.js
  add required dependencies if any
  update of proyect structure
  creation of comandsRedquired.md file
  add devLog.md file for future documentation of development process
  add geoJson files for
  map functionality
  removal of incesessary files (map.js)

List what you planned to accomplish. (e.g)
  [:)]Complete map integration and refactor common components
  [:)]transform header and footer into reusable EJS components
  [:)]transform htmls into EJS templates
  [:)]add map functionality to fishing activity page
  [:)]add necessary routes in app.js
  [:)]add required dependencies if any
  [:)]update of proyect structure
  [:)]creation of comandsRedquired.md file
  [:)]add devLog.md file for future documentation of development process
  [:)]add geoJson files for
  [:)]map functionality
  [:)]removal of incesessary files (map.js)

Describe what you actually worked on. (e.g)

In this development session, the project was refactored to improve maintainability and functionality.
The header and footer were transformed into reusable EJS components,nd all static HTML files were converted into EJS templates.
Full map functionality was added to the Fishing Activity page using GeoJSON data, with corresponding routes updated in app.js
and unnecessary files like map.js removed.The project structure was reorganized, required dependencies were added,
and documentation files comandsRequired.md and devLog.md were created to support future development.

Document how you used Gen AI (e.g., Copilot, ChatGPT, etc.)

Prompt: "I want to connect my GEOJSONs into my ejs element called map. The geojsons are files that contain the information
for the commercial zone in mexico and two geojson about the protected areas"

Output: AI provided two implementation suggestion options:
1. Frontend approach using fetch() to load GeoJSON files directly
2. Backend approach using fs.readFileSync() to pass data through EJS templating
AI recommended Option 2 for better performance and provided complete code examples
for both app.js route handler and map.ejs integration using Leaflet.js

Edits:
- Essential css for map height added to map container
- GeoJSON data loaded server-side in app.js using fs.readFileSync
- Added inline styles and CSS classes to legend color indicators for visual distinction
- Moved console.log statements earlier for better debugging

Reflection: The back-and-forth helped me understand:
- How EJS templating variables are passed from Express routes (res.render with data object)
- The <%- %> syntax for injecting unescaped content (necessary for JSON)
- Why Leaflet maps require explicit height styling to render
- The trade-off between server-side file reading vs client-side fetching

What I Learned:
- Server-side data loading pattern: read files → parse JSON → stringify → pass to template → parse in frontend
- Leaflet.js GeoJSON layer integration with styling, popups, and controls
- CSS requirement: map containers need explicit height or they render at 0px
- EJS variable scope: variables must be passed through res.render() object to be accessible in templates
- Debugging approach: using console.log to verify data loading before rendering

19/10/25 – Page integration and hosting

Aemi Rosas Sánchez

During this day, I integrated our Endangered Page into the main project. Once the pages were connected, I created the Host Page to display everything in one place. I use Render to make the live link of our project.

However, while testing the site, I encountered an error in the Recommendations Page. After debugging, I discovered that the issue was caused by a capitalization error. One of the EJS files had a capital letter in its filename, while the references in the code used lower case letters. Once I fixed the file name to match, the page loaded correctly, and everything worked.

Challenges: I faced a technical issue caused by inconsistent file naming (uppercase vs lowercase). This small detail caused the page not to render properly, and it took some time to identify the source of the problem.

Solutions: We fixed the capitalization error in the EJS file name to match the references in the project, which resolved the issue and allowed the page to load successfully.

20/10/25 – Final Design and Adjustments

Jetzuvely Gonzalez

On this day, I focused on making the final adjustments to the design of the main page, the maps, and the titles. I reviewed the layout carefully and identified small details that could improve the overall look and user experience.

One of the main tasks was ensuring that the design works well on mobile devices. I adjusted some sections so that text, images, and map elements do not overlap or appear too small on smaller screens. This involved checking spacing, resizing certain elements, and improving alignment.

Although the core design was already defined, these small changes were important to make the page feel polished and professional. This process helped me understand how small adjustments can greatly impact usability and visual balance. In the end, the main page, maps, and titles are more coherent, readable, and responsive for different devices.

20/10/2025 Setting Up The Contact Form

Andrea Hinojosa

Goals for the documented session

[ ] Search for solutions to send emails through the forms so that people can contact us.
[ ] Allow the forms to send emails
[] Send a nice message of confirmation

Tasks Completed

[ ] Search for solutions
[ ] Allow the forms to send emails
[] Send a nice message of confirmation

Use of Gen AI

PROMPT 1

Prompt: *“How to set a forms in a webpage to send to gmail.”*
Output: ✅ **Option 3: Backend with Node.js + Nodemailer (For Full Control)**
Edits: I modified some of the given colors to make them clearer and more legible, and modified the font so it didn't look so out of place..
Reflection: It helped me get a general idea of which design suited him best.

PROMPT 2

Prompt: *“Can you explain it to me step by step?”*
Output: Great — your server is tidy and nearly ready. I’ll walk you through, step-by-step, exactly what to change/add so your Render app accepts form submissions and sends them by Gmail using **nodemailer**. I’ll keep everything in CommonJS to match your code.

```
// --- Add near top of file, after other requires ---
const nodemailer = require("nodemailer");
require('dotenv').config(); // only if using .env locally (safe to keep)

// --- Add body parsers if not already added ---
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

// --- New route to handle form submissions ---
app.post("/send", async (req, res) => {
  try {
    const { name, email, message } = req.body;

    if (!name || !email || !message) {
      return res.status(400).send("All fields (name, email, message) are required.");
    }

    // Configure transporter using Gmail and App Password
    const transporter = nodemailer.createTransport({
      service: "gmail",
      auth: {
        user: process.env.GMAIL_USER,
        pass: process.env.GMAIL_PASS
      }
    });

    // Build email
    const mailOptions = {
      from: `"Website Contact" <${process.env.GMAIL_USER}>`, // shows as from your Gmail
      replyTo: email, // reply goes to the user who filled form
      to: process.env.GMAIL_USER, // you receive the message
      subject: `New contact form message from ${name}`,
      text: `Name: ${name}\nEmail: ${email}\n\nMessage:\n${message}`,
      html: `<p><strong>Name:</strong> ${name}</p>
             <p><strong>Email:</strong> ${email}</p>
             <p><strong>Message:</strong><br/>${message.replace(/\n/g, "<br/>")}</p>`
    };

    // Send mail
    await transporter.sendMail(mailOptions);

    return res.status(200).send("✅ Email sent successfully.");
  } catch (err) {
    console.error("Error in /send:", err);
    return res.status(500).send("❌ Failed to send email.");
  }
});
```

Edits: I stopped at the middle of this step because I realized the app somehow would have access to my account and I don't want my information to be exposed.
Reflection: Understanding what chatGPT says to you is important because imagine what would happen if I had simply done what the AI said without being aware that my data could be exposed.

.

PROMPT 3

Prompt: I am concerned that my data may be exposed with this method.
Output:We can use **Ethereal Email**, a **fake SMTP testing service**. It's designed exactly for development/academic projects:

```
const nodemailer = require('nodemailer');

app.post('/send', async (req, res) => {
  const { name, email, message } = req.body;

  // Validate form fields
  if (!name || !email || !message) {
    return res.status(400).send('All fields are required.');
  }

  try {
    // Create a test account on Ethereal
    let testAccount = await nodemailer.createTestAccount();

    // Create a transporter using the test account
    let transporter = nodemailer.createTransport({
      host: 'smtp.ethereal.email',
      port: 587,
      auth: {
        user: testAccount.user,
        pass: testAccount.pass
      }
    });

    // Define the email content
    let info = await transporter.sendMail({
      from: `"Academic Project" <${email}>`, // from the form
      to: testAccount.user,               // you “receive” it in the test account
      subject: `New message from ${name}`,
      text: `Name: ${name}\nEmail: ${email}\nMessage:\n${message}`
    });
    // Send the preview URL back to the browser
    console.log("Preview URL:", nodemailer.getTestMessageUrl(info));
    res.status(200).send(`✅ Email sent! Preview it here: ${nodemailer.getTestMessageUrl(info)}`);

  } catch (err) {
    console.error('Error sending email:', err);
    res.status(500).send('❌ Failed to send email.');
  }
});
```

Edits: I added this part to the code in the app and adapted it to the real variables that the forms have.

Reflection: I didn’t imagine there was a way to do it just as a test.

Prompt: I want the data sent from the font to be saved so that I can see it later because Ethereal only displays it temporarily.

Output:

```
//1️⃣Save submission locally
    const submissions = JSON.parse(fs.readFileSync(submissionsPath, 'utf8'));
    submissions.push({ name, email, subject, message, date: new Date() });
    fs.writeFileSync(submissionsPath, JSON.stringify(submissions, null, 2));
```

Edits: I added this part to the code in the app and added the URL also to the JSON file.

Reflection: I thought it would be harder to send the email, but I’m happy that it would not.

Prompt: I want to open a popup window if the mail is sent, so I have to send the response to the front end, right?

Output:

```
const formData = new FormData(this);
const data = Object.fromEntries(formData.entries());
try { //converts the info to json file and sends it to the server
const response = await fetch('/send', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify(data) });
 if (result.success) {
      alert(result.message); // simple popup
      this.reset(); // optional: clear the form
    } else {
      alert("Error: " + result.message);
    }
  } catch (err) {
    alert("Error sending request");
    console.error(err);
  }
});
```

Edits: I added this part to the eventlistener part in our code.

Reflection: This change makes the code cleaner and allows the data to be sent to the server in a structured way, without having to create a variable for each field in the form.

Prompt: how can I activate a modal from bootstrap?
Output:**using JavaScript manually** (to control it programmatically).

```
 const modal = new bootstrap.Modal(document.getElementById('exampleModal'));
  document.getElementById('openModalBtn').addEventListener('click', () => {
    modal.show(); // opens the modal
  });
```

Edits: I combined the last prompt with this one. Instead of showing an alert, I showed the modal with moda.show().
Reflection: Sending a modal instead of an alert makes the UI prettier and I learned how Bootstrap can be activated: by Javascript (coding) or html (using `data-bs-toggle` and `data-bs-target`..). The best option was javascript because I don't want to press a link or target to activate it.

Prompt: "[Code] The client prints "Error sending request", but the server says "Received data".
Output: Perfect, that means that the server is receiving and processing the info, but the client is failing to read the response… [] Check in the browser console if the request reaches the browser correctly and what the server replies.
Edits: I created a new modal named modal.
Reflection: The browser is very literal in what the problem is and that is very helpful. I should stop to see it before freaking out because it didn't work.

Resources Used

Ethereal Email
Nodemailer

What I Learned

-There are many ways in which you can send emails through a website
-The order in which the modules are set is important for the app to start correctly
-Sometimes there are errors that you have until they are magically solved. I had an error "submissions variable is not defined" and there was not a variable with that name. Suddenly, without any changes in the code, when I started the server, it decided to work. I mean, of course it was not magic, probably the file had not been updated or something like that.
-You have to be very careful with the variables, I usually have mistakes in having different names so they are not recognized by the server and I don't realize.
-I learnt about the node's module that is used to send emails form web pages in an easier way.
-I didn't know anything about SMTP which is Simple Mail Transfer Protocol and allows emails to be sent by the internet. I didn't go that deep into that world, just enough to understand the code, but I had never thought about how emails are sent and now I know it's through this protocol.
-There is a better way to send all the information from the form fields
-There's a better way to send all the information from the form fields, because with FormData(this); everything is sent at once.
-Object.fromEntries converts data into a hashtable.

-application/json is the format in which the data is sent to the server. I already knew how to send it as html format, but didn't know how to do it as a json.

Challenges & Bugs

❌ I didn't want to expose my gmail.
✅ Used a fake SPTM that is used for tests.
❌ When modifying the code to add the futures there were errors I didnt understand.
✅ Debug the code correctly.
❌ The modal wasn't showing. The client printed an error sending request, but the server printed "received data".
✅ I asked chat to help me and told me that the problem was not the server, but the client. Since inside the try was more steps than just sending the request, if something else failed, it also sent the error, which was happening. Chat told me to check the browser console and it said "modal undefined", which meant that I didn't have a variable named modal, so I added it.
❌ The modal wasn't showing in the public website, but it was working well in the localhost.
✅ It was because sending the fake email interfered with the functionality, so we decided to look for more options to send the email. For now, the info that the user sends to us is saved into a json.

Resources Used

Ethereal Email

21/10/2025 Solving Design Problems

Andrea Hinojosa

Goals for the documented session

[ ] Maintain fishing activity map into the bootstrap container
[ ] Center the map into the div
[ ] Copy the changes of the color of the nav-bar in the main page to the other ones
[ ] Make the recommendation page responsive

Tasks Completed

[ ] Maintain fishing activity map into the bootstrap container
[ ] Center the map into the div
[ ] Maintain fishing activity map into the bootstrap container
[ ] Center the map into the div
[ ] Copy the changes of the color of the nav-bar in the main page to the other ones
[ ] Make the recommendation page responsive

Use of Gen AI

PROMPT 1

Prompt: *"How to make this container responsive so that the content doesn't go out of it in smaller windows?"*

Output: 🧩 1. Wrap everything in a responsive container
2. Use Bootstrap's responsive grid correctly
3. Avoid fixed widths or heights in your CSS
Prompt: This is my code, I'm avoiding fixed widths as you said, why is the div not working yet? .rectangle{ background: linear-gradient(white,#b9eff9); border-radius: 10px; padding: 10px; width: 70%; box-shadow: 0px 4px 10px rgba(0,0,0,0.1); margin-top: 10px; margin-bottom: 20px; display: flex; } .contact, .forms{ align-items: center; padding: 20px; width: 50%; }
Output: Perfect 👍 That CSS explains exactly why your layout overflows on smaller screens — and fixing it is super simple. Let's look at what's happening and how to fix it without changing your visual design 👇So inside `.rectangle` (which is 70% of the page), you have two elements of 50% each side by side. That works fine on wide screens,
but on narrow screens (like phones), 50% + 50% still try to sit horizontally → they overflow out of the container.

```
.rectangle {
  width: 100%;              /* use full width of Bootstrap col */
  max-width: 900px;           /* optional max width to limit it */
  flex-wrap: wrap;           /* ✅ allows wrapping on small screens */
  justify-content: center;    /* keeps content centered */
}

.contact, .forms {
  min-width: 300px;          /* prevents shrinking too small */
}

/* ✅ Stack the two blocks vertically on small screens */
@media (max-width: 768px) {
  .contact, .forms {
    width: 100%;
  }
}
```

Edits: I added lines to the code so that the content doesn't flow out. Regarding the rectangle, its width changed to 100% so that it covers all the bootstrap containers, I added flex-wrap so that the content goes down in smaller windows, and justify the content center to keep the content centered and added @media.
Reflection: It is important to add

What I Learned

-@media is to set different conditions to the behaviour of that element depending on the screen size.
-flex wrap is for the content to move down when there is not enough space for them to be.

-The prompt that helped me to understand the CSS code of the rectangle that contains the forms helped me to adjust by myself the title in the hero, which was coming out of the blue rectangle.

Individual

Andrea

With each exercise I've done, I can see that everything actually works based on requests and responses. Even if you're doing something unfamiliar and seemingly out of this world, you always start from the basics, and it's not difficult to understand how it works. Also, I'm really impressed with all the things that you can do online. Before this class I didn't picture how everything online was done (meaning how the emails travel through the internet) and I think it is amazing that there are many possibilities that are easy to implement. I'm happy because I understand and implement the topic better and better. Although we couldn't implement the sending and receiving emails for now, I learned a lot and I think that that is the most important thing.

Team work reflection

I'm satisfied with my teammates' work. I think we maintain excellent communication and distribution of tasks. Everyone has done an excellent job with their parts and we have delivered everything on time and with high quality.

Aemi

Andrea and I developed the map of endangered marine animals in Mexico. At first, we explored different map tools and APIs, switching from Google Maps to OpenLayers, and finally to Leaflet for easier use with JavaScript. We used the IUCN Red List API, but since it had too much data, we filtered it to show only critically endangered species in Mexico.

Then I integrated the map into the Endangered Page, later merging it with the main project. Along the way, I fixed errors like a capitalization issue in an EJS file and improved the site´s performance and organization.

Team work reflection

I think our team worked well together and maintained good communication throughout the project. We supported each other, shared ideas, and divided tasks.

Eduardo

Personal work reflection

I discovered that the data required for the project is available; however, there is no accessible API to retrieve it. Therefore, I believe we should prepare to generate a dedicated database containing this information, allowing users to access it without having the data stored directly within the project folder.

Team work reflection

Collaborating as a team introduced diverse priorities and perspectives, enabling us to support one another in tackling challenging tasks and ensuring collective progress throughout the project.

Jetzuvely

Personal work reflection

During this project, I focused on improving my CSS skills and ensuring the layout and responsiveness of the website were functional across devices. I also explored different ways to integrate maps and GeoJSON data. Overall, I feel satisfied with my contributions and progress.

Team work reflection

Working with the team was a positive experience. Communication and collaboration were smooth, and everyone contributed their knowledge effectively. I learned from my peers, shared ideas, and we managed to implement the project goals efficiently. Everything went super well, and the teamwork was excellent.