Carbon Footprint

Project title: Carbon Footprint
Developers: Eros Marucchi and Edoardo Antonini
Company: ProfileTree
Manager: Ciaran Connolly
Year: 2024

1 Introduction

1.1 Project Information

- Developers: Eros Marucchi and Edoardo Antonini
- Manager: Ciaran Connolly
- Company: ProfileTree
- Type of project: Web
- Start date: 10.06.2024
- End date: 17.06.2024

1.2 Scope of application

This project has two main objectives. The first is to design and implement a website that calculates carbon emissions generated by monthly visits to other websites. Users will be able to enter the URL of the page for which they wish to calculate emissions and our system will provide a detailed report on the associated carbon emissions. The web application will also provide estimates with practical examples, so that the calculated emission amounts can be better understood.
This objective aims to make users aware of the environmental impact of web surfing and provide useful tools to reduce the digital carbon footprint.

The second objective of the project is educational. This project will serve to consolidate our digital knowledge through practical application. It will be an opportunity to deepen our skills in web development, including HTML, CSS, JavaScript and the use of advanced libraries and frameworks.

2 Analysis

2.1 Domain analysis

The product will come used to mainly by the clients of ProfileTree but everybody can used, so this means that includes all the age groups because it's simple easy to utilize

---

2.1.1 Explanation of the items in the requirements table

ID: unique identifier of the requirement

Name: brief description of the requirement

Priority: indicates the importance of a requirement within the overall project, defined together with the client. For example, having reports with columns of different colors has a lower priority compared to having a database with the elements inside it. Typically, a maximum of 2-3 priority levels are defined.

Version: indicates the version of the requirement. Each modification of the requirement will have an updated version. Only the latest version will appear in the documentation, while the old ones should be included in the logs.

Notes: any important observations or references to other requirements.

Sub-requirements: elements that make up the requirement. List and describe the means available for the realization of the project. Remember to always describe in detail the versions and the reference model.

2.2 Requirements Analysis and Specification

ID: REQ-01	
Name	Creating the interface for the application
Priority	1
Version	1.0
Notes	A simple and intuitive graphical interface is required for all types of users.
Sub-Requirements	
001	requires an input to enter the URL of the page whose emissions you want to calculate.
002	requires a button to display information other than emissions.
003	requires an input to change the number of monthly views for the emission examples.
004	requires a button to print or download the page as a PDF

ID: REQ-02	
Name	Multi-language site
Priority	1
Version	1.0
Notes	-
Sub-Requirements	
001	There must be the possibility of translating the site into several languages.

ID: REQ-03	
Name	Page saving
Priority	1
Version	1.0
Notes	-
Sub-Requirements	
001	It must be possible to print the new image directly from the application.
002	It must be possible to download the content of the page in PDF format directly from the application.

ID: REQ-04	
Name	Responsive website
Priority	1
Version	1.0
Notes	A simple and intuitive graphical interface is required for all types of users.
Sub-Requirements	
001	the website must be able to be used on any device

2.2.1 Means analysis

To do this project I was provided with a PC from the company ProfileTree

---

2.2.2 Software

- Visual Studio Code 1.78.2, text editor to create the web application.
- Google Chrome 116.0.5845.188, browser to test the application.
- FireFox 110.0, browser to test the application.
- Microsoft Edge116.0.1938.81, browser to test the application.

2.2.3 Hardware

- Processor: 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz 3.50 GHz
- Installed RAM: 64.0 GB
- Device ID: E04FD891-668E-43B8-A507-B0FB5F4A2D79
- Product ID: 00355-60674-96746-AAOEM
- System type: 64-bit operating system, x64-based processor
- Windows version: Windows 11 Pro, 23H2

---

3 Designing

3.1 Interface design

3.1.1 Index.html

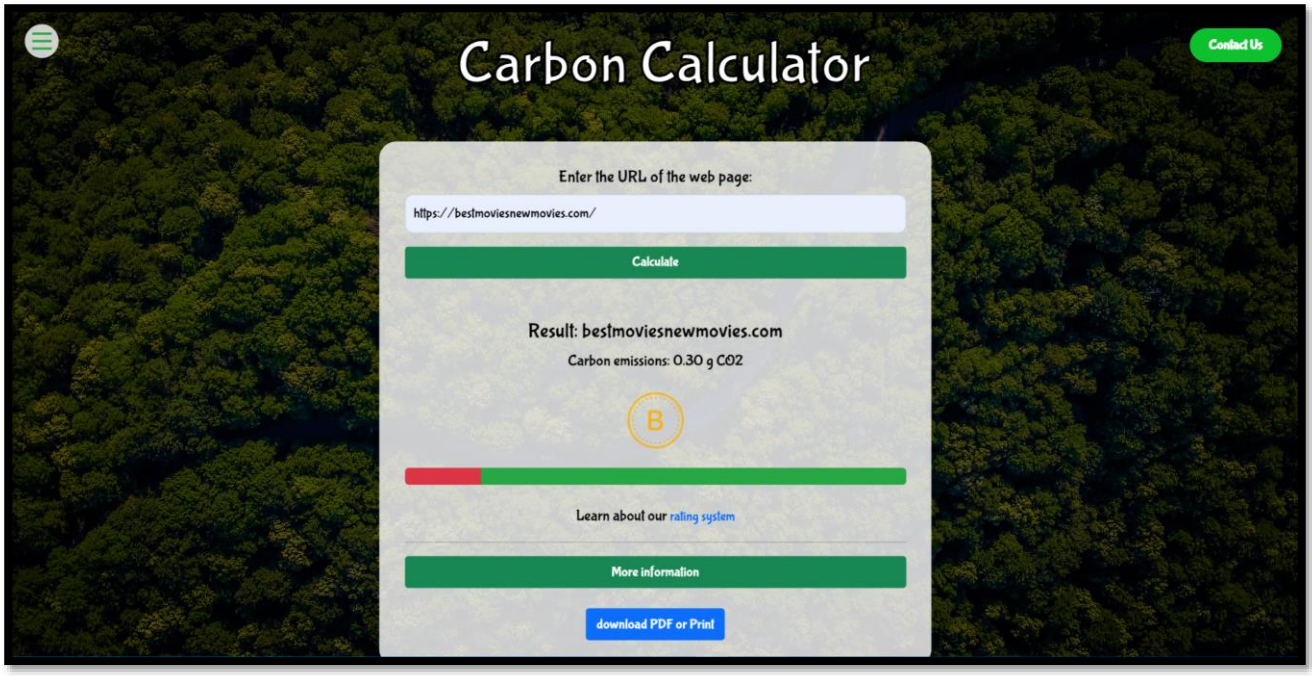


Image 1 - index.html

This is the principal page, here we can find in center a box area, inside this area we have a text area who will come insert the URL of the page whose we will calculate the carbon emissions. Under we find the result of the calculation, with the possibility of see how they are calculated the various page, we have also the possibility of read more information concerning the calculation and print or download a file with the information

3.1.2 rating.html

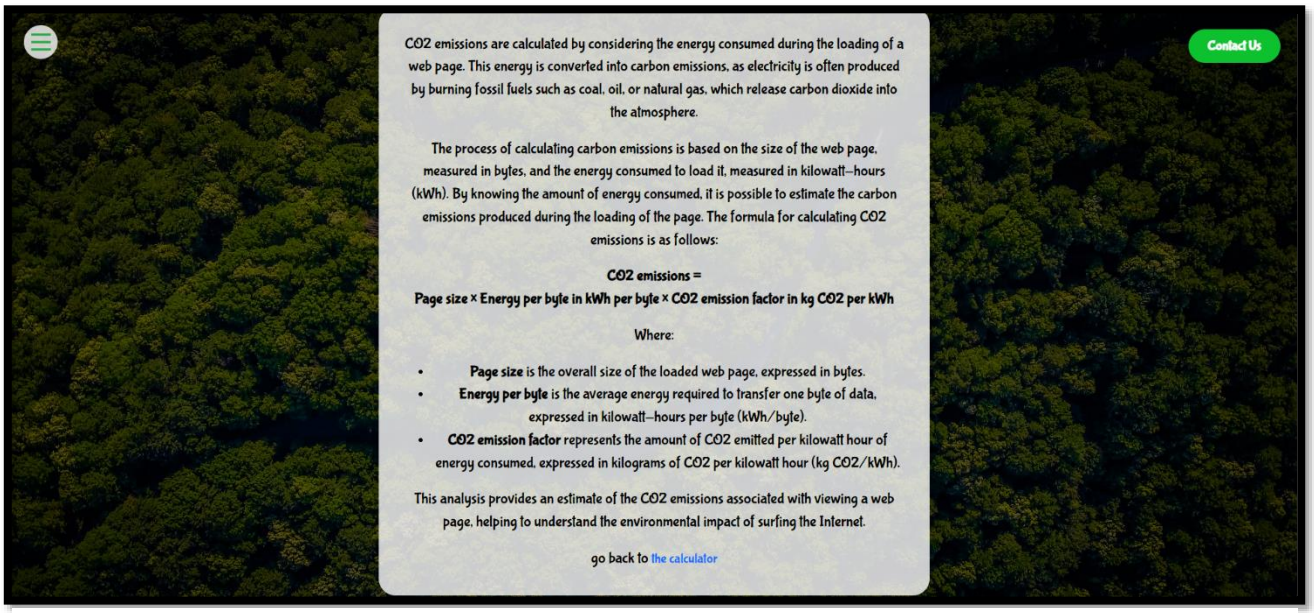


Image 2 - rating.html

In this page we find a box area with the explanation about the calculation, a box area with a table on how the websites are ranked and a box area with a graphic always how the websites are ranked.

In the upper table, you can see the rating ranges we wanted to use.
While the graph, only shows the trend of the ratings in a visual way to make it easier to understand.

3.1.3 aboutUs.html

This page represents where the ProfileTree have headquarters, under this we find all the social network that the ProfileTree have with a link

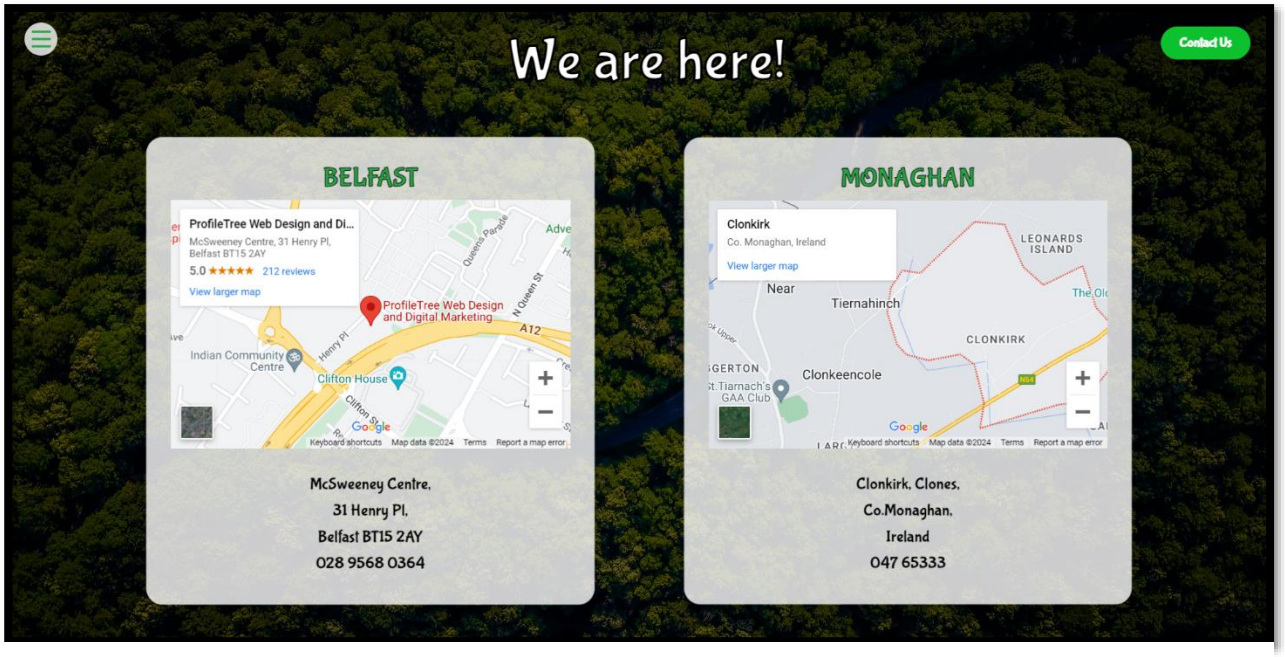


Image 3 - aboutUs.html

3.1.4 policy.html

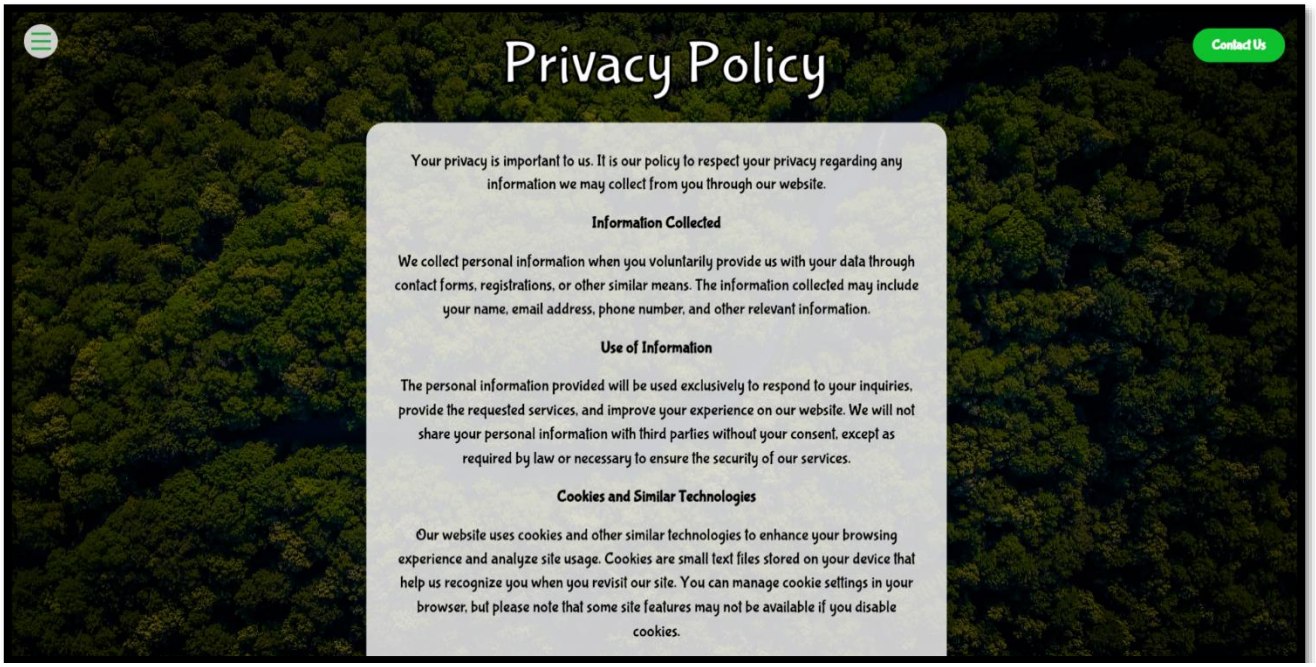


Image 4 - policy.html

This is the page about the Privacy Policy of the company.

4 Implementation

The project is structured in 15 separate files for better readability and order.
The architecture comprises 4 files which contain all the elements necessary for the user to interact with the application.
There are also 3 JavaScript files, which contain the code to implement all the functionality of the application.
Finally, there are also 8 CSS files, which contain all the visual styles of the application.

4.1 URL cleaning

```
/* This function is for cleaning URLs */
function cleanUrl(url) {
    try {
        const parsedUrl = new URL(url);
        return `${parsedUrl.protocol}//${parsedUrl.hostname}`;
    } catch (error) {
        console.error('Invalid URL:', error);
        return null;
    }
}
```

Image 5 - URL cleaning

The cleanUrl function has the task of cleaning a URL, returning only the protocol (such as "http" or "https") and the domain name (hostname). When the function receives a URL as input, it attempts to create a URL object with the supplied URL. If the URL is valid, the function returns a string consisting of the protocol and the domain name. However, if the URL is invalid, an error is caught, an error message is printed in the console, and the function returns null.

4.2 Calculation of energy consumption

```
/* this function performs the calculation for the energy consumption of a website */
function calculateEnergyConsumption(pageSizeBytes) {
    const energyPerByte = 0.000015;
    const totalEnergyConsumption = pageSizeBytes * energyPerByte;
    return totalEnergyConsumption;
}
```

Image 6 - Calculation of energy consumption

The function calculateEnergyConsumption calculates the energy consumption of a website based on the page size. When the function receives as input the page size in bytes (pageSizeBytes), it multiplies this value by an energy consumption factor per byte (energyPerByte), which is equal to 0.000015. The result of this multiplication is the total energy consumption of the web page, which is then returned by the function.

4.3 Calculation of carbon emissions

```
/* this function is used to calculate carbon emissions */
function calculateCarbonEmissions(energyConsumption) {
    const conversionFactor = 0.12;
    const carbonEmissions = energyConsumption * conversionFactor;
    return carbonEmissions;
}
```

Image 7 - Calculation of Carbon emissions

The function calculateCarbonEmissions calculates carbon emissions based on energy consumption. When the function receives energy consumption (energyConsumption) as input, it multiplies this value by a conversion factor (conversionFactor), which is 0.12. The result of this multiplication is the amount of carbon emissions produced, which is then returned by the function.

4.4 Download PDF

```
// Download the contents of the page
function downloadPDF() {
    const doc = new jsPDF();
    const element = document.getElementById('pdf-content');
    const options = {
        margin: { top: 10, left: 10, right: 10, bottom: 10 },
        html2canvas: { scale: 0.8 },
        filename: 'carbon_footprint_report.pdf',
        jsPDF: { unit: 'pt', format: 'a4', orientation: 'portrait' },
        pagebreak: { mode: 'avoid-all' }
    };

    html2pdf().from(element).set(options).save();
}
```

Image 8 - Download PDF

The downloadPDF function allows the contents of a specific HTML element of the web page to be downloaded as a PDF file. It uses the jsPDF and html2pdf libraries, applying predefined formatting options such as margins, rendering scale, format and orientation of the PDF.
It converts the element with the pdf-content ID into a PDF with the name carbon_footprint_report.pdf, ensuring that the content is well formatted and without inappropriate interruptions.

4.5 Custom Print

```
// Prints the contents of the page
function customPrint() {
    window.scrollTo(0, 0);
    setTimeout(function () {
        window.print();
        document.body.removeChild(printContainer);
    }, 300);
}
```

Image 9 - Custom Print

The customPrint function is designed to improve the process of printing a web page. When activated, the function first scrolls the browser window to the top of the page, ensuring that printing starts at the top rather than at a scrolled position. It then introduces a short delay of 300 milliseconds to ensure that the scrolling is completed, before actually starting the browser print window with window.print();.
Once printing is complete, the function removes a specific element called printContainer from the document, making sure to clean up any temporary or supporting elements used during the printing process.

4.6 Letter from Emissions

```
/* this function decrees the carbon emission letter for each website */
function getLetterFromEmissions(emissions) {
    if (emissions >= 1.5) {
        return 'F';
    } else if (emissions >= 1.2) {
        return 'E';
    } else if (emissions >= 0.9) {
        return 'D';
    } else if (emissions >= 0.6) {
        return 'C';
    } else if (emissions >= 0.3) {
        return 'B';
    } else {
        return 'A';
    }
}
```

Image 10 - Letter from Emissions

The getLetterFromEmissions function evaluates the carbon emission level of a website based on a numerical value provided as a parameter. Depending on the range in which this value falls, the function returns a letter from 'A' to 'F', representing the lowest to the highest emission ratings respectively.
This allows users to get a quick indication of the level of environmental impact of a website based on its carbon emissions.

4.7 Update Examples

```
/* Function to update CO2 emission examples based on monthly visits */
function updateExamples() {
    const carbonEmissionsText = document.getElementById('carbon-emissions').innerText;
    const carbonEmissions = parseFloat(carbonEmissionsText.split(' ')[2]);

    const monthlyVisits = parseInt(document.getElementById('monthlyVisits').value);
    if (isNaN(monthlyVisits) || monthlyVisits < 1) {
        alert('Please enter a valid number of monthly visits.');
        return;
    }

    const annualVisits = monthlyVisits * 12;
    const annualCO2 = (carbonEmissions / 1000) * annualVisits;

    if (isNaN(annualCO2)) {
        alert('Error calculating annual CO2 emissions.');
        return;
    }

    const teaCups = (annualCO2 / 0.00736).toFixed(0);
    const smartphoneCharges = (annualCO2 / 0.0053).toFixed(0);
    const kWhEnergy = (annualCO2 / 0.128).toFixed(2);

    if (isNaN(teaCups) || isNaN(smartphoneCharges) || isNaN(kWhEnergy)) {
        monthlyVisits = 10000;
        alert('Error converting CO2 emissions.');
        return;
    }

    document.getElementById('carbon-emission-examples').innerHTML = `
        <p>With this visits per month, this page emits approximately ${annualCO2.toFixed(2)}
        kg of CO2, which is equivalent to:</p>
        <ul>
            <li>Boiling water for ${teaCups} cups of tea</li>
            <li>Charging an average smartphone ${smartphoneCharges} times</li>
            <li>Consuming ${kWhEnergy} kWh of energy</li>
        </ul>
    `;
}
```

Image 11 - Update Examples

The updateExamples function updates the carbon-emissions examples based on the number of monthly visits to a website. It starts by retrieving the current carbon-emissions text from the HTML element with the ID carbon-emissions and converts it into a numerical value (in kg of CO2). Then, it obtains the number of monthly visits from the HTML element with the ID monthlyVisits and converts it into an integer number. If the number of monthly visits is invalid, it displays a warning message and terminates. It then calculates the number of annual visits by multiplying the monthly visits by 12 and determines the annual CO2 emissions by multiplying the CO2 emissions per visit (in kg) by the number of annual visits. If an error occurs in the calculation of annual CO2 emissions, it displays an error message and terminates. The function then converts the annual CO2 emissions into three equivalences: cups of tea (using a conversion factor of 0.00736 kg CO2 per cup), smartphone charges (using a factor of 0.0053 kg CO2 per charge) and kWh of energy consumed (using a factor of 0.128 kg CO2 per kWh).
If there are errors in the conversion, it displays an error message and terminates.

5 Test

5.1 Test Protocol

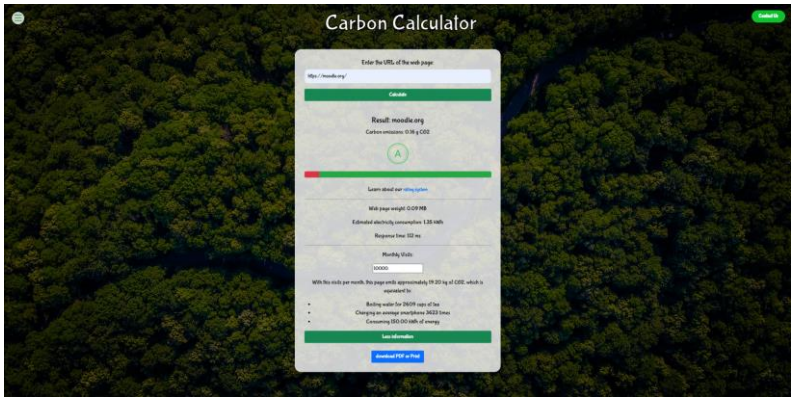
Test Case: **Reference**:	**TC-001** REQ-01	**Name:**	**GUI**
Description:	- The page must have an input to enter the URL, it must have a button to show other information, it must have an input number to change the monthly visits and it must have a button to print o download		
Prerequisites:	- Have the text area and a button		
Procedure:	1. Enter the Url 1. Press the button “Calculate”		
Expected Results:	- The page have all the Sub-requirements		

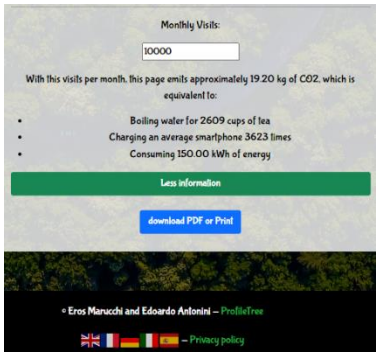
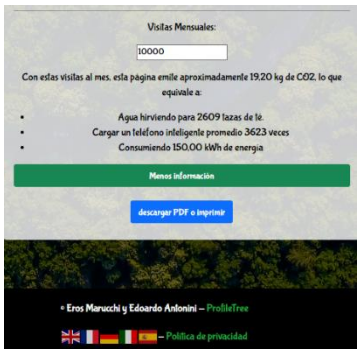
Test Case: **Reference**:	**TC-002** REQ-02	**Name:**	**Multi-language**
Description:	- The pages must be able to be translated		
Prerequisites:	- Have the function in the code for translate		
Procedure:	1. Go in the footer of any page 2. Find the national flag 3. Press one of these		
Expected Results:	- The pages are translated		

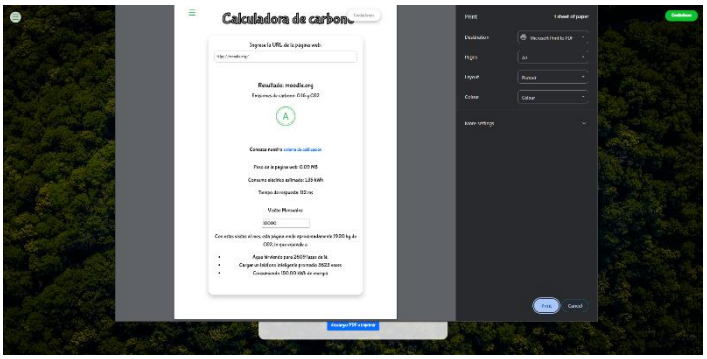
Test Case: **Reference**:	**TC-003** REQ-03	**Name:**	**Page saving**
Description:	- The pages must be able to be saving		
Prerequisites:	- Have the button and the function for translate		
Procedure:	1. Enter the Url 1. Press the button “Calculate” 2. Press the button “download PDF or Print” 3. One time for download 4. One time for print		
Expected Results:	- The page is downloaded and printed		

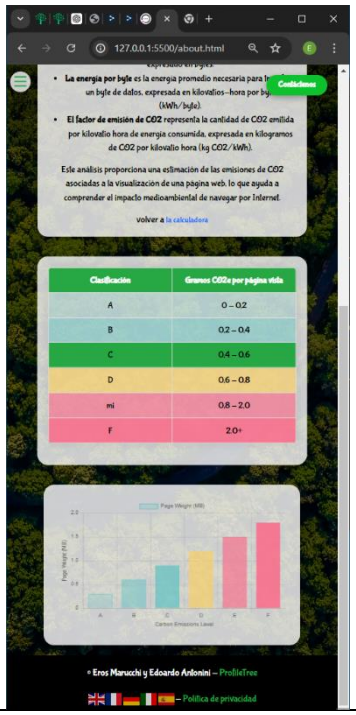
Test Case: Reference:	TC-004 REQ-04	Name:	Responsive
Description:	- The pages must be responsive		
Prerequisites:	- nothing		
Procedure:	1. Take a corner and shrink and enlarge		
Expected Results:	- The pages don't lose the beauty		

5.2 Test results

Test Case	Result	GUI	Date Test
001	Passed	**Expected result:** The page has all the requirements **Effective result:** The page has all the requirements. **Foto:** 	13.06.24

Test Case	Result	Multi-language	Date Test
002	Passed	**Expected result:** The pages change language **Effective result:** The pages change language when press the flag **Foto:**   	13.06.24

Test Case	Result	Responsive	Date Test
003	Passed	**Expected result:** the page fits **Effective result:** The page is responsive **Foto:**  	14.06.24

Test Case	Result	Page saving	Date Test
004	Passed	**Expected result:** It's possible save the page **Effective result:** The page is saved or printed **Foto:** 	17.06.24

5.2.1 Test Summary

ID Test-Case	Risultato	Commento / Note	Data
TC-001	Passed		13.06.2024
TC-002	Passed		13.06.2024
TC-003	Passed		14.06.2024
TC-004	Passed		17.06.2024

6 Conclusions

The proposed application could be extremely useful for users interested in calculating the carbon emissions of specific websites. It can be used both for educational purposes, to teach pollution-related concepts, and for pure entertainment, to find out the emissions of sites of personal interest. The impact could be significant as the application addresses an important topic such as pollution, potentially raising awareness of this environmental issue among users. Furthermore, the ability to download and print the data obtained allows users to share information and promote debates on pollution, contributing to greater public awareness and discussion on the subject.

6.1 Future Developments

To improve and further develop an application for calculating the carbon emissions of websites, several future developments could be considered.
First, one idea could be the expansion of the application to include historical data and emission trends, allowing users to monitor the evolution of the environmental impact of websites over time. Introducing a rating or certification system based on environmental performance could incentivise site owners to improve the energy efficiency of their platforms.
Furthermore, integrating the application with educational or environmental platforms could expand the use of the application as an educational tool for schools, universities and non-profit organisations, contributing to greater awareness and action against digital pollution.
Finally, considering the creation of an online community or forum within the application could foster the sharing of ideas, experiences and best practices to reduce the environmental impact of the web, promoting constructive debate and a collective commitment to a more sustainable future.

6.2 Personal considerations

Developing such an application was a surprise for both of us. From the very beginning, we found the project extremely interesting and challenging. As we continued working on it, our curiosity grew and in the end it was extremely satisfying to complete it. We both appreciated the use of JavaScript for this type of application, as the language proved to be versatile and suitable for handling the complexities of the web operations required.

The project was successfully completed according to our expectations, and during this process we considerably strengthened our skills in HTML, CSS and JavaScript. We were particularly impressed by the idea behind the project, which aims to make users aware of the carbon emissions generated by even just viewing a web page, due to the energy consumption required for its operation.

From a personal point of view, we firmly believe that projects of this kind are crucial to educate and promote greater awareness of the environmental impact of digital technologies. We hope that in the future more developers and users will strive to design and use web applications in a more sustainable way, thus contributing to a more ecologically responsible digital world.

7 Index of image

8 References

8.1 Sitography

- https://chatgpt.com/ - 10.06.2024
- https://allorigins.win/ - 11.06.2024
- https://gtranslate.io/ - 11.06.2024
- https://elfsight.com/ - 12.06.2024
- https://www.w3schools.com/js/ - 12.06.2024
- https://remixicon.com/ - 13.06.2024
- https://getbootstrap.com/ - 13.06.2024
- https://www.cssportal.com/ - 14.06.2024

9 Attachments

- Project description (1_Description\Project_Description.txt).
- Application (3_Application).
- Explanation of emission calculation (4_Attachments\emission_calculation.txt).
- Demo Project (4_Attachments\Project_demo.mp4).