Marcelo Gaspar drmpgaspar@gmail.com

MyPortfolio

27th January 2020 - 3th February 2020

Overview

Nowadays having a portfolio in web format is a good way to make yourself known and so that everyone can have access to information about you in an easy and dynamic way.In this project I have created a personal portfolio through NodeJS and ExpressJS.

Table of Contents

Phase I - Project Initiation

Project Requirements

1. General Analysis

The main concepts / entities that will be worked on this site are:

- About
- Soft skills
- Technologies
- Work experiences

1.1. User Interface design

At least the interface should have the following sections:

- About:
    - Summary about you.
        - At least a degree, a description, and a summary of technologies, soft skills and experience as a CV.
- Soft skills:
    - Soft Skills List
- Technologies:
    - List of technologies with their score according to your experience
- Experience:
    - List / Block showing the main information about each experience
        - It will have at least one title and one main block of text
- Contact:
    - Contact form that when sent enter the contact in the database and send by email.

You will also develop a control panel to manage the previous sections that will cover the following needs:

- Login to identify yourself in the control panel
- Logout

- Modify summary information about yourself (About)
- CRUD of the technologies
- CRUD of experiences
- CRUD of soft skills
- List contact requests that have been made

1.3. Implementation

To carry out this project it will be necessary to take into account two main roles:

- The backend system that is responsible for obtaining the information from the database
- The frontend system that will be responsible for consuming the data provided by your own backend system.

There are two “Fronts”:

- The public portfolio
- The control panel that will be used to manage the portfolio information

1.4. Architecture

It is important that the application logic (the domain) is completely isolated from the rest of the application and that you use the MVC architecture pattern. This pattern will force you to organize your code cleanly and clearly.

Project Specifications

- You must use GIT from the beginning of the project
- You must develop the project with NodeJS and ExpressJS
- You must use MySql to manage the database.
- All code must be properly documented
- You must include a small user guide to understand how to interact with the tool
- The interface must be fully responsive
- All comments included in the code must be written in English
- Use the camelCase code style
- In the case of using HTML, never use online styles
- In the case of using different programming languages always define the implementation in separate terms
- It is recommended to divide the tasks into several subtasks so that in this way you can associate each particular step of the construction with a specific commit
- For the project documentation a PDF version is required within the repository
- You should try as much as possible that the commits and the planned tasks are the same
- Delete unused files

Quality Control Measurements

Even though the project should adhere to all of the *Project Requirements* and *Specifications*, there are some conditions that, if properly met, add a sense of quality and robustness to the project itself. Those conditions are:

- The HTML and CSS code must be validated by W3C.
- The JavaScript code must be lint-free, as stated by a trusted JavaScript linter, like ESLint
- The web application must be fully responsive.
- The web application must be compatible with the main browsers in the market:
    - Internet Explorer 11 or higher.
    - Safari in one of its latest versions.
    - Firefox in one of its latest versions.
    - Chrome in one of its latest versions.

Phase II - Project Planning

Verbal Reasoning

Based on the *Project Overview*, the end result will be a portfolio app. This service will offer two interfaces: one is the public page visible to logged-in and non logged-in users, while the other is like a control panel for the logged-in users. The capabilities of each interface are listed in the *Project Requirements*.

These two interfaces will be served by the same server: it is enough to differentiate between them by the route the client is using to perform the different queries. In the other hand, just panel control interface will be handling entities and do queries for models, like a administrator.

That calls for the use of the ***M**odel-**V**iew-**C**ontroller* (***MVC***) architectural pattern: the different ***models*** will speak to the Database Service to gather and store the necessary information, the ***controllers*** will define which actions will be performed when the client requests specific information; and the ***views*** will represent visually the data being handled.

Now, the Database Service will consist of an instance of *MySQL Server 8.0* running in the background, without any special configuration. The connection will be made through the ***root*** user, via ***localhost*** and the default port ***3306***.

Task Planning

The following set of tasks, each comprised of a numeric hierarchical identifier, a short title, a priority value (from 1 to 5), a difficulty value (from 1 to 5), an estimation of the duration, and its dependencies, are an effort to synthesize and streamline the development of the project: by dividing the project into a task tree, where the deeper the level the higher granularity of the task, development efforts can be focused into completing one branch after the other methodically, with the final result being a project successfully completed on time.

1.	Document the project during its life-cycle			
	1.1.	Phase I - Project Initiation	[4 - 2 - 30min]	
	1.2.	Phase II - Project Planning	[4 - 4 - 4 h]	[1.1]
	1.3.	Phase III - Project Execution	[4 - 4 - 1 h]	[1.2]
	1.4.	Phase IV - Project Closure	[4 - 4 - 2 h]	[1.3]
2.	Set up the project infrastructure		[5 - 3 - 1 h]	[1.2]
	2.1.	Set up the GitHub repo	[5 - 1 - 15 min]	
	2.2.	Set up the MySQL Server	[5 - 3 - 15 min]	
	2.3.	Set up the node/express local server	[5 - 3 - 15 min]	
	2.4.	Set up the folder structure	[5 - 2 - 15 min]	

3.	Develop the backend of the project	[4 - 4 - 5 h]	[2.1]
3.1.	Test sample MVC architecture	[2 - 2 - 30 min]	[2.3]
3.2.	Test connection to MySQL Server	[4 - 2 - 15 min]	[2.2]
3.3.	Implement MVC architecture	[4 - 4 - 4 h]	[2]
3.3.1.	Implement the authController	[3 - 4 - 45 min]	
3.3.2.	Implement the aboutController	[3 - 4 - 30 min]	
3.3.3.	Implement the skillsController	[3 - 4 - 30 min]	
3.3.4.	Implement the techsController	[3 - 4 - 45 min]	
3.3.5.	Implement the experienceController	[3 - 4 - 45 min]	
3.3.6.	Implement the contactController	[4 - 4 - 45 min]	
4.	Develop the frontend of the project	[4 - 4 - 5 h]	
4.1.	Layout creation	[2 - 2 - 30 min]	
4.2.	Html structuring (with handlebars)	[3 - 2 - 30 min]	
4.3.	Implementation of Views	[4 - 3 - 4 h]	
4.3.1.	Implement the loginView	[3 - 3 - 30 min]	
4.3.2.	Implement the aboutView	[4 - 3 - 45 min]	
4.3.3.	Implement the skillsView	[3 - 3 - 30 min]	
4.3.4.	Implement the techsView	[4 - 3 - 45 min]	
4.3.5.	Implement the experienceView	[3 - 3 - 30 min]	
4.3.6.	Implement the contactView	[3 - 3 - 1 h]	
5.	Integrate backend with frontend	[5 - 4 - 1 h]	[3 4]
5.1.	Resolve conflicts	[5 - 5 - 30 min]	
5.2.	Test integration	[5 - 5 - 30 min]	

Chronogram of the Project Execution Phase

Based on the estimated completion times of the previous tasks, the following chronogram tries to position in time the completion of each set of tasks, finishing with the completion of the phase.

ID	Blocks of 15min																								
2	X	X	X	X																					
2.1	X																								
2.2		X																							
2.3			X																						
2.4				X																					
3					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
3.1					X	X																			
3.2							X																		
3.3								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
3.3.1								X	X	X															
3.3.2											X	X													
3.3.3													X	X											
3.3.4															X	X	X								
3.3.5																		X	X	X					
3.3.6																					X	X	X		
4					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
4.1					X	X																			
4.2							X	X																	
4.3									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
4.3.1									X	X															
4.3.2											X	X	X												
4.3.3														X	X										
4.3.4																X	X	X							
4.3.5																			X	X					
4.3.6																					X	X	X	X	

5																					X	X	X	X
5.1																					X	X		
5.2																							X	X

Risk Management Plan

In every project there are risks that should be taken into account. These risks may pause or even halt the project, so it should be a priority to have plans in case any of the risks becomes a reality. The following table lists the possible risks that may obstruct the development of the project. It's important to note that the probability follows a scale from 1 (unlikely) to 5 (highly likely), as well as the impact, which also follows a scale from 1 (low) to 5 (high). Priority is calculated via the product of the probability and the impact:

ID	Risk	Consequence	Prob. (1-5)	Imp. (1-5)	Pri. (1-25)	Mitigation approach
1	Losing the laptop	Wouldn't be able to work	1	5	5	Keep the repo updated + find another workstation
2	Getting sick	Wouldn't be as productive	2	3	6	Follow healthy schedules
3	Low performance and productivity	Deadlines wouldn't be met	2	5	10	Rethink the deadlines + filter out some nice-to-have's + work on personal diligence
4	Unrealistic deadlines	Deadlines wouldn't be met + development shortcuts would have been taken affecting the robustness of the code	2	3	6	Rethink the deadlines + split the conflicting tasks into subtasks and set deadlines to each subtask
5	Divergence from established tasks	Loss of control over the development flow of the project	4	5	20	Stop the current task and follow the established list of tasks

Git Workflow

All commits are going to be pushed to the ***master*** branch, following a personal criteria of only uploading snapshots that are functional and that work properly, not counting minor bugs. There are no other branches, as it would slow down the development process.

On another note, commit messages start with its main objective stated in square brackets: for example, the commits related mainly to CSS changes start with ***[styling]***; those related mainly to the design of the page, ***[layout]***; those related mainly to the documentation, ***[documentation]***.

Tools

Different tools were used in the development of the project. They are the following:

- ***git:*** a powerful version control system that helps keep track of the changes in the working tree.
- ***Visual Studio Code:*** a code editor optimized for building and debugging modern web applications.
- ***Google Chrome Developer Tools:*** used for debugging the JavaScript code and for testing layout adjustments.
- ***Google Docs:*** used for writing the project documentation.
- ***W3C Validator:*** used for validating the HTML and CSS code.
- ***ESLint:*** used for validating the JavaScript code.
- **Bootstrap:** a modern responsive front-end framework.
- **MySQL**: for database.

Phase III - Project Execution

Folder Structure

Use the MVC architecture pattern to organize my code cleanly and clearly.

Incidents and Lessons

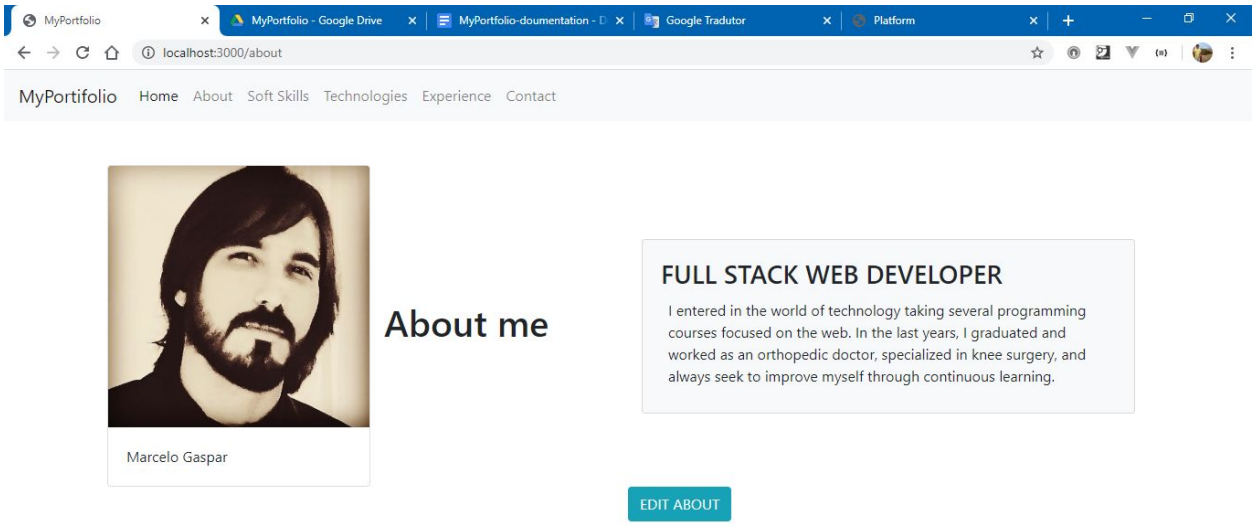
- Improved my programming knowledge
- Improved my knowledge in the implementation of MVC
- Improved my knowledge of NodeJS
- Improved my knowledge of the ExpressJS framework
- Improved my knowledge of CSS & HTML & Javascript
- Improved my knowledge in database design and creation
- Improved my knowledge in project management and teamwork

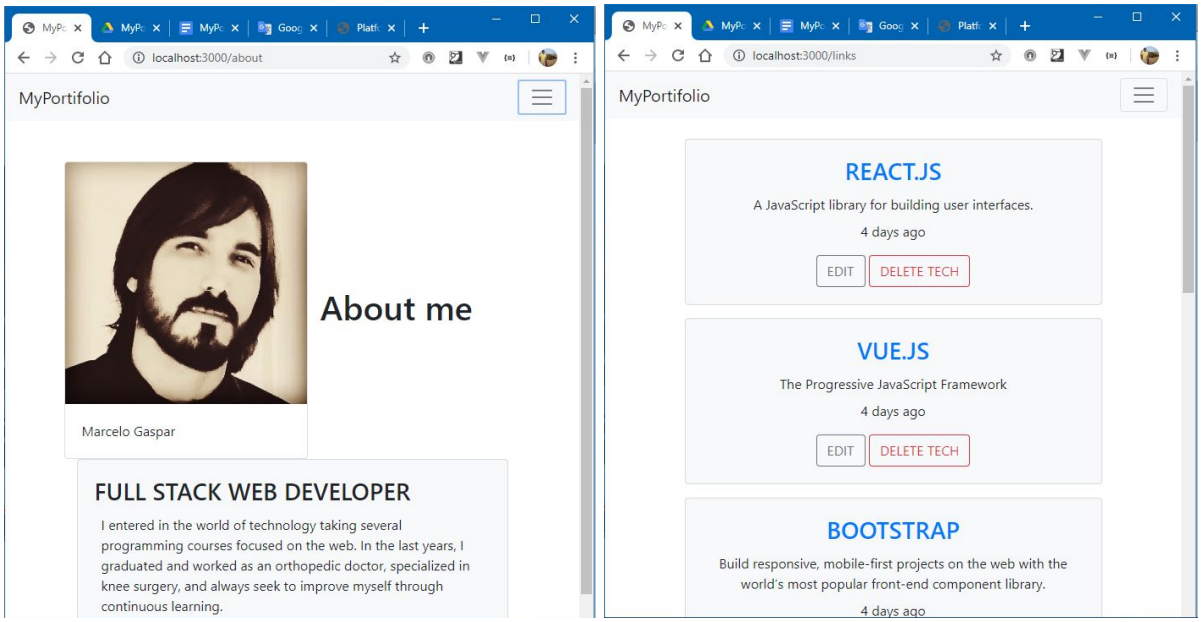
Phase IV - Project closure

Quality Metrics

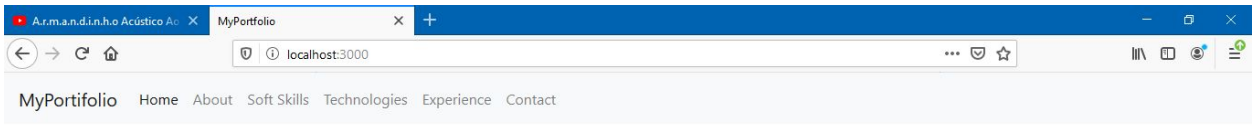
Browser Compatibility

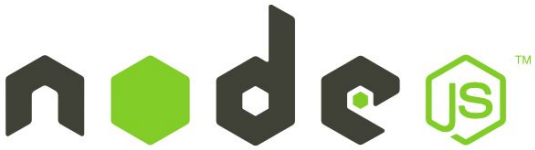
Google Chrome

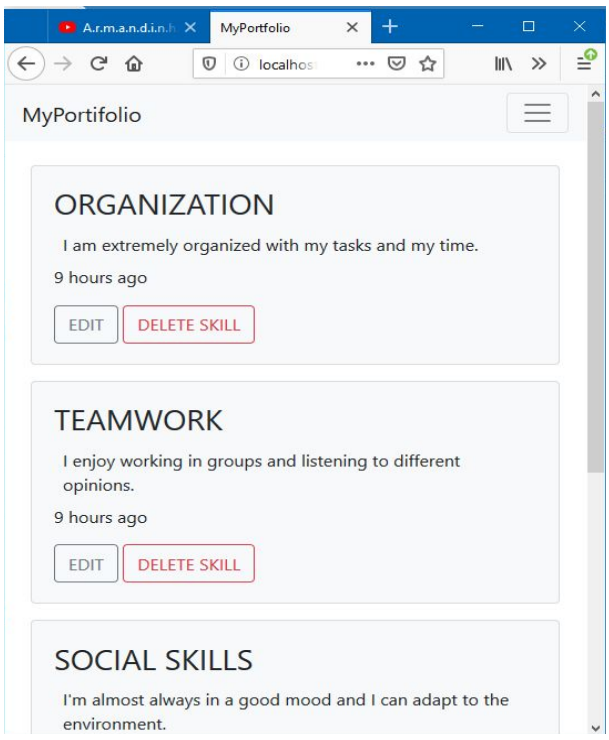


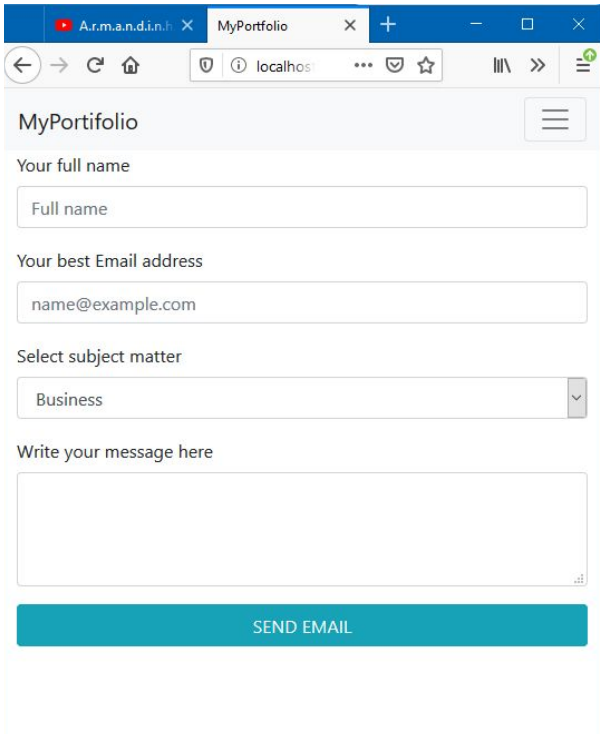


Mozilla Firefox









Opera

Safari

Edge

Internet Explorer 11

General comments

Overall, the project is interesting, fulfilling and gratifying, but its extension doesn't fit in our timelines. Once the MVC pattern is internalized the whole structure can be thought out in one go, but it has not been the case for the team. Many conceptual roadblocks prevented the team from developing properly and had to constantly review and rewrite the code.