Design Decisions

The website has been designed with the following focus areas and stated requirements in mind:

1. The website must serve static pages
2. The layout must have a responsive design (to work with phones and tablets)
    - That implies the usage of javascript (universally accepted in all browsers)
3. The content must be simple to edit/add
    - It has to use a content language to avoid working directly over HTML
        - Markdown has been implicitly selected as per requirements and suggested solution. I have been investigating about it and seems a very good option as content language.
    - It needs to be in a separate files to avoid confusion with HTML code
    - The tools to edit such files must be simple and widely available for non-technical people
4. It must be possible to be deployed in any web server or websites platform
5. The code must be stored in a central location for easy access to different users

Having all these factors into account, the proposed solution fulfills the above points by:

1. The design is created in very simple HTML static files
    - The average page of the website only contains 80 lines of code
    - I am not a web developer or programmer, so the code is really basic
2. The website uses bootstrap css and javascript library to provide the responsive design
3. The content is stored in separate text files containing markdown content
    - Any text editor (gedit in linux, TextEdit in Mac OS, notepad in Windows, etc...) is the only tool needed to edit the markdown content
    - Showndownjs javascript library is used to convert the markdown content to html on run time
        - Is very fast
        - Is very powerful, accepting markdown but with focus on HTML conversion, allowing to alter the design if desired
        - Does not require any other unnecessary extra software to work with a website (Jekyll, Ruby, etc...)
4. Basic HTML and javascript files have the advantage of meeting the requirements with all hosting solutions
    - Being only basic files, it can be deployed in any web server and maintained by any individual (non-technical), not tying the architecture to any specific technology (like Jekyll)
5. Can work with any repository (or even a simple FTP) for the code
    - Because it was implicitly stated in the requirements, I have been working with git and GitHub as repository and used GitHub Pages for accessing the website. Not my preferred option, but of course it will work without problems.

Challenging the suggested solution

A solution was suggested for building the website using the Jekyll static generator technology. Comparing it to the above presented solution, I see some disadvantges in the initial proposal:

- Editing the content with Jekyll is NOT simple:
    - You have to install a Ruby environment:
        - "Simple" instructions: source
        - Is it really necessary to use so big *programming* language to be able to edit text of a small website?

        - As a real-life statistic: how many non-technical co-workers in our team have a Ruby environment installed? Aren't they supposed to add content to the website too?
    - You have to install Jekyll software and learn how to use it:
        - "Simple" instructions to use Jekyll: source
        - Reality check: how many co-workers in our team (even technical ones) will prefer these commands over using a simple text editor?
    - As far as I saw Jekyll mixes a lot markdown notation with tricky HTML code in it, making far more harder to debug and fix HTML problems in production. You end up not learning proper markdown documentation that is compatible with any markdown reader application and for sure not learning proper web development to be used in non-Jekyll websites.
- Although Jekyll has allegedly became popular the last year, truth is that less than 0.03% of the world websites use this technology (source). Actually, Hugo seems to be even more used, but still with a 0.05% of penetration. Learning a technology that is used so little in the real world does not feel appealing to me, and risks its usefulness if one day Jekyll dissapears (like many other web technologies have done in the past). But learning basic HTML and javascript is a more solid foundation for my knowledge because that is what the web (and Jekyll too!) is really using in the end.