GradeCalc
https://gradecalc.app

COLLEGE GRADES MANAGER AND SPECULATOR PROGRESSIVE WEB APP

By Maurici Abad Gutierrez
https://mauriciabad.com

Bachelor Degree in Informatics Engineering (Software Engineering)

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)
Facultat d'Informatica de Barcelona (FIB)

2nd July, 2020

Director: Cristina Gómez Seoane

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

COLLEGE GRADES MANAGER AND SPECULATOR

PROGRESSIVE WEB APP

Abstract

by Maurici Abad Gutierrez, GEI
Universitat Politècnica de Catalunya (UPC)
2nd July, 2020

Director: Cristina Gómez Seoane

English

GradeCalc is an open-source Progressive Web App to help students manage their current grades and calculate the necessary mark in the upcoming exams to pass a subject. This document showcases the different functionalities offered by this application, the pre-development study, the application's architecture, and the validation strategies used.

Español

GradeCalc es una aplicación web progresiva de código abierto que ayuda a los estudiantes a administrar sus notas actuales y calcular las necesarias en próximos exámenes para aprobar alguna asignatura. Este documento presenta las diferentes funcionalidades ofrecidas por esta aplicación, el estudio previo al desarrollo, la arquitectura de la aplicación y las estrategias de validación usadas.

Català

GradeCalc és una aplicació web progressiva de codi obert que ajuda als estudiants a administrar les seves notes actuals i calcular les necessàries en pròxims exàmens per aprovar alguna assignatura. Aquest document presenta les diferents funcionalitats oferides per aquesta aplicació, l'estudi previ al desenvolupament, l'arquitectura de l'aplicació i les estratègies de validació utilitzades.

Contents

Page

ABSTRACT III

BODY TEXT

1 INTRODUCTION 1
1.1 The problem 1
1.2 The solutions 2
1.2.1 The expected solution 2
1.2.2 The most used solution 2
1.2.3 The proposed solution 3
1.3 State of the Art 5
1.3.1 Generic grade calculators 5
1.3.2 Atenea - UPC's virtual campus 6
1.3.3 Spreadsheet document 6
1.4 Stakeholders 7

2 PROJECT MANAGEMENT 9
2.1 Scope 9
2.1.1 Objectives 9
2.1.2 Risks and obstacles 10
2.1.3 Changes in the objectives 11
2.2 Workflow 12
2.2.1 Methodology 12
2.2.2 Tools 14
2.2.3 Changes in the workflow 15
2.3 Schedule 15
2.3.1 Estimates 15

2.3.2 Description of tasks . . . 17
2.3.3 Gantt diagram . . . 19
2.3.4 Required resources . . . 20
2.3.5 Risk management: alternative plans and obstacles . . . 20
2.3.6 Changes in the schedule . . . 21
2.4 Budget . . . 23
2.4.1 Human resources costs . . . 23
2.4.2 Hardware costs . . . 25
2.4.3 Software costs . . . 27
2.4.4 Other costs . . . 28
2.4.5 Cost contingency . . . 29
2.4.6 Unforeseen and extraordinary costs . . . 29
2.4.7 Total costs . . . 30
2.4.8 Management control . . . 31
2.4.9 Changes in the budget . . . 31
2.5 Sustainability . . . 32
2.5.1 Self-assessment . . . 32
2.5.2 Economic Dimension . . . 32
2.5.3 Environmental Dimension . . . 33
2.5.4 Social Dimension . . . 34
2.6 Laws and regulations . . . 35

3 REQUIREMENTS AND SPECIFICATION . . . 37
3.1 Functional requirements . . . 37
3.2 Non-Functional requirements . . . 43
3.3 Conceptual model . . . 45
3.4 Tasks . . . 47

4 SOFTWARE DESIGN . . . 61
4.1 Software architecture . . . 61
4.1.1 Logical architecture . . . 61
4.1.2 Physical architecture . . . 64
4.2 Presentation layer's design . . . 67
4.2.1 User Interface . . . 67
4.2.2 Navigational map . . . 86
4.2.3 Flowcharts . . . 87

4.3 Domain layer's design . . . 90
4.4 Data layer's design . . . 91
4.4.1 Data model . . . 92
4.4.2 Example objects . . . 94

5 IMPLEMENTATION . . . 96
5.1 Tech stack . . . 96
5.1.1 Analysis of alternatives . . . 98
5.2 Brand identity . . . 101
5.2.1 Color Palette . . . 101
5.2.2 Typography . . . 103
5.2.3 Logo and icons . . . 104
5.2.4 Voice . . . 105

6 TESTING . . . 106
6.1 Manual tests . . . 106
6.2 Automated tests . . . 113
6.3 Satisfaction survey . . . 118
6.3.1 Acquisition . . . 118
6.3.2 Questions . . . 120
6.3.3 Answers . . . 121
6.3.4 Conclusions . . . 127

7 DEVOPS . . . 131
7.1 Continuous integration . . . 131
7.2 Continuous deployment . . . 134

8 POST-DEVELOPMENT . . . 135
8.1 Maintenance tasks . . . 135
8.1.1 Fix critical bugs . . . 135
8.1.2 Git repository administration . . . 135
8.1.3 Tidy up subjects in the database . . . 136
8.1.4 Promote the app . . . 136
8.1.5 Note down sporadic feedback . . . 136
8.2 Web analytics . . . 137
8.2.1 Algolia's search reports . . . 137
8.2.2 Google Analytics . . . 139

8.2.3 Google Search Console 141

9 CONCLUSIONS 143

9.1 Encountered issues 143

9.2 Accomplished objectives 145

9.3 Technical competences' accomplishment 146

9.4 Integration of knowledge 149

9.5 Future improvements 150

9.6 Personal takeaways 151

REFERENCES 154

LIST OF TABLES 155

LIST OF FIGURES 156

APPENDIX

A Satisfaction survey answers 161

Chapter One

INTRODUCTION

1.1 The problem

As a student, I've always had a way of **keeping track of my grades, and making speculations** about what grades I need in upcoming exams to pass my courses. For my own experience, speculating has been a very successful way of focusing my efforts on the right assignments and passing the courses.

One day, talking to my colleagues, I realized that I wasn't the only student doing this. The reality is that many students chose to aim for the minimum grade instead of performing their best. This may be due to a bunch of reasons, ranging from being lazy to finding the assignments fruitless.

1.2 The solutions

1.2.1 The expected solution

The straight forward solution would be that the university provided an official way of seeing the grades of each assignment. But in many cases, it doesn't solve the problem because:

- Some teachers don't publish the grades.
- The interface is confusing and difficult to understand.
- There's no speculating implemented.
- Some universities don't even have this kind of app.

Because **this solution is unreliable**, students that can't benefit from this solution have to come up with their ones.

1.2.2 The most used solution

Each student managed his grades differently. This is the typical procedure I saw students from my faculty following:

1. Save the grades in:

   - Notes app
   - Agenda
   - School portal
   - Memory
   - Paper
   - Spreadsheet

2. Speculate the necessary grades. Generally filling the undone exams with made up values, in a test-error way, until getting the desired final grade. Using:

   - Phone basic calculator
   - Advanced calculator
   - Spreadsheet

The most used method is writing over, and over, the calculation in the phone calculator until the right value is found.

In conclusion, **the whole process is rudimentary and tedious**, it may lead to mistakes in the calculations causing failed courses, and the students not having their grades and speculations easily accessible.

1.2.3 The proposed solution

I suggest making a specific application for this use case. It would be a productivity app that lets students store their grades and also lets them speculate to find the best combination of grades to pass, considering their current ones. All this with excellent ease-of-use.

One year ago, I already identified this issue so **I created a very basic app for myself** that helped me calculating the necessary grades in the upcoming exams to pass my courses. I published it in a domain and shared it with my friends. This started a chain reaction, students started recommending it to their friends because they liked it a lot. Against all odds, the users spread the app through the faculty, to a point that the app had 1,000 views/day during December 2019 (Fig. 1.1). Due to this unexpected success, I'd like to improve the app and also make it suitable for other subjects.

As of February 2020, this app doesn't have a database, and subjects are hardcoded by myself, also it is not always working. I developed this project whenever I had free time and I used it as a playground for learning web development. It was a personal project for myself, not thought to be scaled up. The fact that people used it and gave me positive feedback motivated me to improve it. I want to make this app usable by many more students, and let them create and edit subjects, between other things.

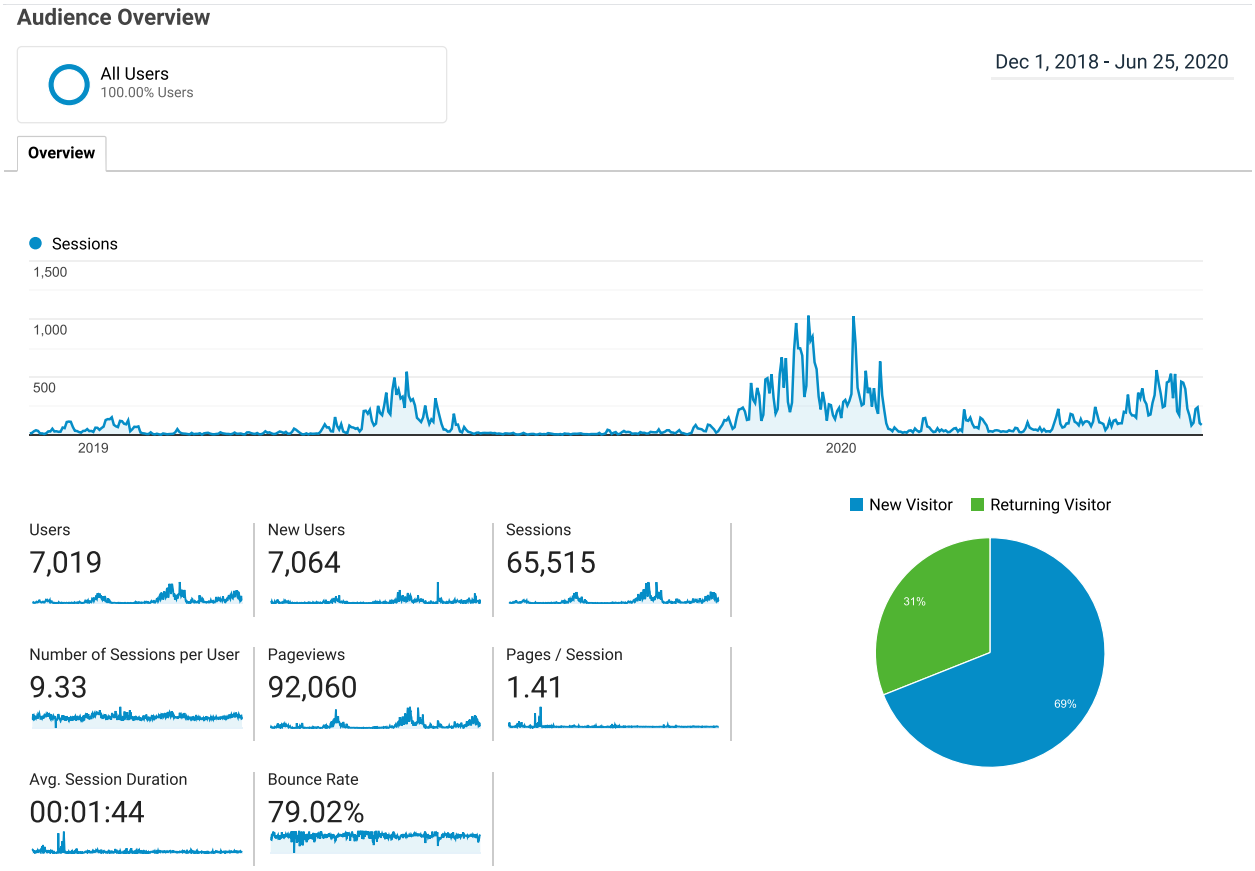


Figure 1.1 GradeCalc analytics - Sessions from Dec 2018 to June 2020

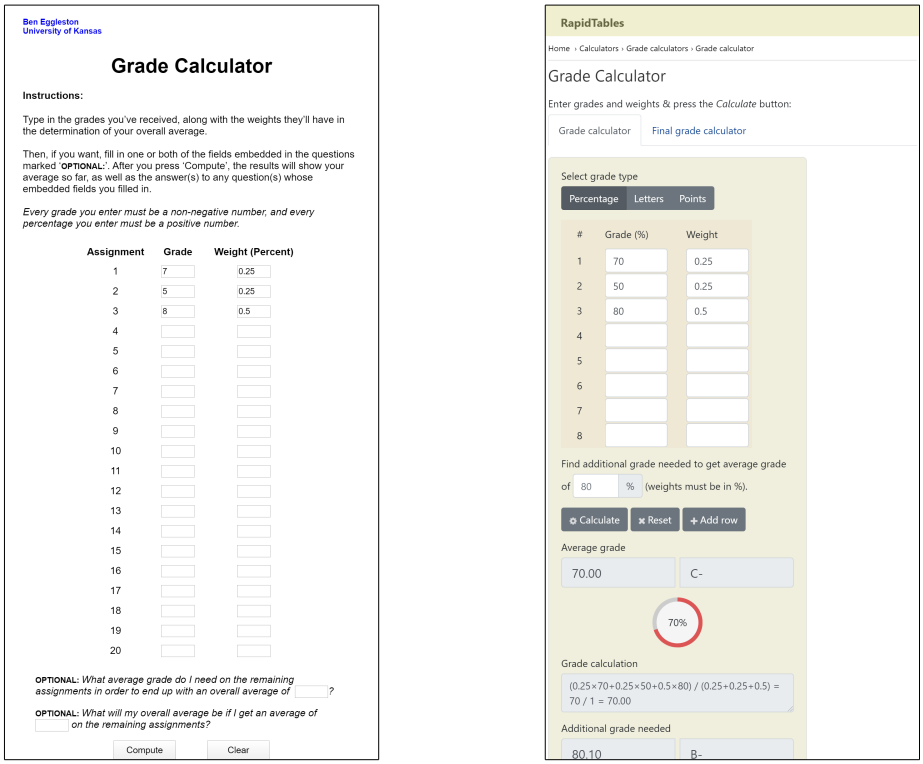
1.3 State of the Art

1.3.1 Generic grade calculators

There are plenty of generic grade calculators, but they don't fulfill the needs of UPC students because they lack the following key features:

- Can't save grades nor evaluation formulas.
- Doesn't estimate the necessary grade to pass.
- You have to rewrite all the parameters every time you refresh.
- They have a very bad user experience.
- Most of them use the American F-A system instead of the Spanish 0-10 system.

In fact, using the phone calculator is faster than this kind of apps.



(a) Ben Eggleston [13] **(b)** RapidTables [34]

Figure 1.2 Generic grade calculators

1.3.2 Atenea - UPC's virtual campus

UPC University has an online campus platform that includes a qualification table, where students can see their current grade.

- Not available for all courses.
- Not available outside UPC.
- Doesn't estimate the necessary grade to pass.

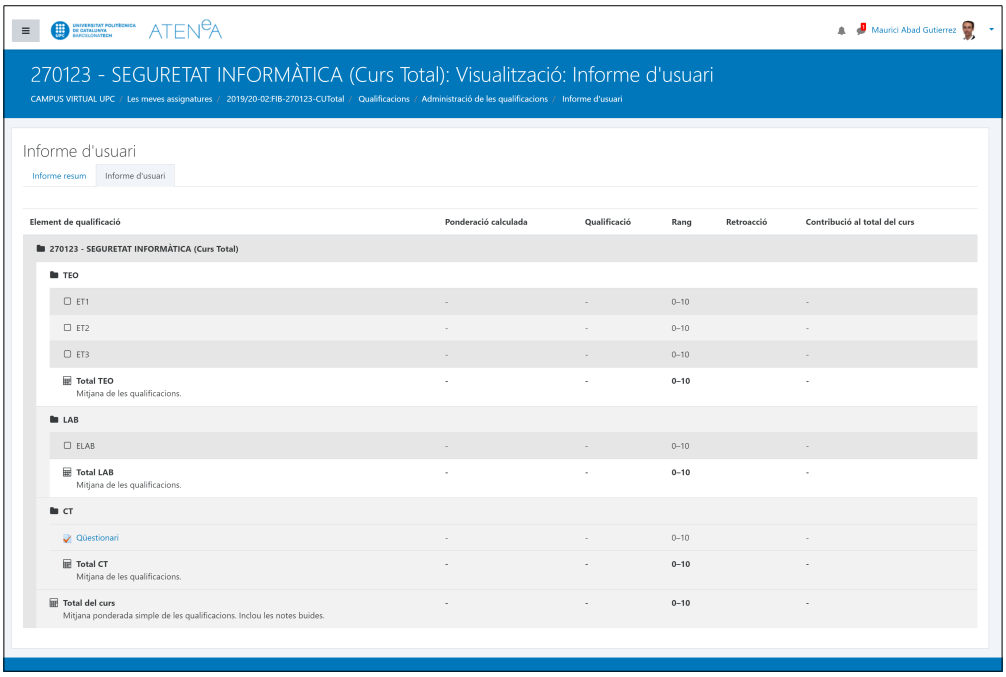


Figure 1.3 Atenea UPC [41]

1.3.3 Spreadsheet document

Some students use a spreadsheet to calculate their grades, but it has many disadvantages:

- Not accessible from the phone.
- You have to enter all evaluation formulas.
- Not specific to the use case.
- Doesn't estimate the necessary grade to pass.

1.4 Stakeholders

The main users are going to be Students of Informatics Engineering in UPC, so they are the most important stakeholder. Then there are students on the same campus, and then from all Barcelona faculties.

Another really important stakeholders are the Developers from the community, that are going to voluntarily help in the development of the app, they need to be satisfied to keep developing or the app won't be able to get updates.

Finally, there are the users who use the app. They must be really satisfied to share the app with their friends if the app becomes popular it will help more students.

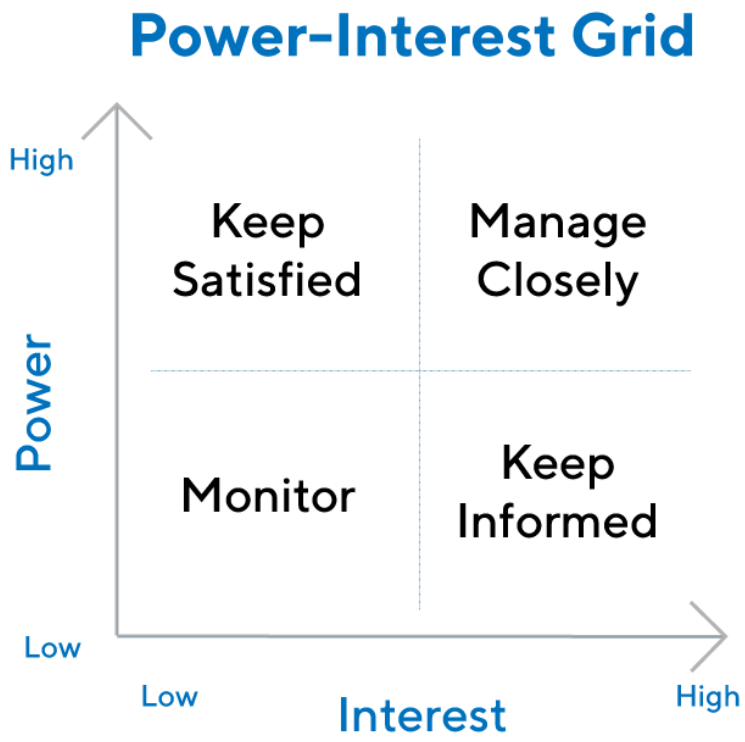
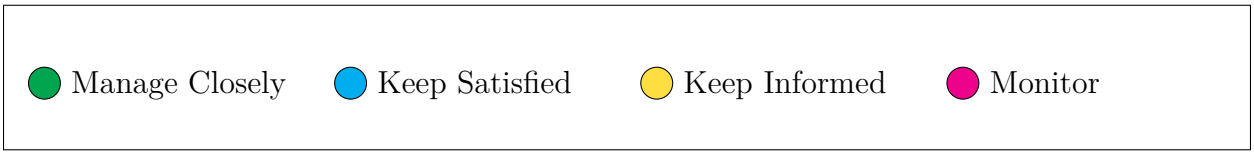


Figure 1.4 Power-Interest grid [33]



This is the stakeholder list with its Power-Interest category:

- Developers
- UPC University
- Students
  - By degree:
    - **Students of Informatics Engineering in UPC**
    - Students of Telecommunications Engineering in UPC
    - Students of other degrees
  - By location:
    - Students in Barcelona
    - Students in Spain
    - Students from everywhere
  - By app experience:
    - Students that are enjoying the app
    - Students that know and use the app
    - Students that don't know the app

Chapter Two

PROJECT MANAGEMENT

2.1 Scope

2.1.1 Objectives

The main objective of this project is to add as many requirements as possible and improve the app in general. To be more specific, the following are the most important aspects to treat:

- **Add functionalities.**
- **Migrate code to a JavaScript task runner or bundler.**
- Improve UI design.
- Brand design.
- Bug fixing.
- Add tests to the code.
- Implement Continuous Integration and Deployment.
- Implement users accounts.

2.1.2 Risks and obstacles

During the development of the project, there may be risks and obstacles. These are the ones I detected at the beginning:

- **No decent free option**: I don't want to spend money on the project if it doesn't pay back, by an economical income, or gained knowledge. So a highly probable risk is not finding any free service (like Hosting, Database, or Instant search engine) for a specific thing. This would mean using an alternative that provides a worse user experience or not having the feature at all.

- **Not enough time**: Not having time to add a substantial amount of new features or releasing them too slowly may harm users' engagement. A cause of this problem may be having to learn many technologies new to me, although extra learning time is considered in the schedule.

- **App becoming forgotten**: I fear that once I finish my bachelor no one is going to publicize the app among students, this is why marketing and making the app engaging is an important topic in this project. I want this project to last several student generations.

- **Competence appearing**: If the UPC releases a better Atenea version, many students may stop using the app. In part, this is one of the reasons why this app should be used for students from other degrees.

- **Fitting everyone's needs**: Every course has it's own rules, especially from different degrees. Also, every student likes doing it in his way. It's really difficult to find one solution that fits everyone, so defining the right requirements is critical.

2.1.3 Changes in the objectives

In the end, some objectives changed from the beginning. The main objective didn't change. This is the adjustments that took place:

- **Migrate code to a JavaScript task runner or bundler** was *Migrate code to a JavaScript framework*. I set up a bundler instead of migrating to a framework so less refactoring was required and I could spend more time in other tasks while achieving similar results.
- **Brand design** was *Marketing of the app*. I trimmed the scope into designing the brand specifically.

2.2 Workflow

2.2.1 Methodology

The project is going to us the **Kanban** agile methodology, and **Jira** to manage the project. I chose agile against waterfall because it makes it easier to continue the project once ended and it's also easier to share the tasks with the open-source community. I also considered using Scrum, but because I'm going to be the only active developer it would add a lot of overhead to the management, so with Kanban, I'll be able to develop faster.

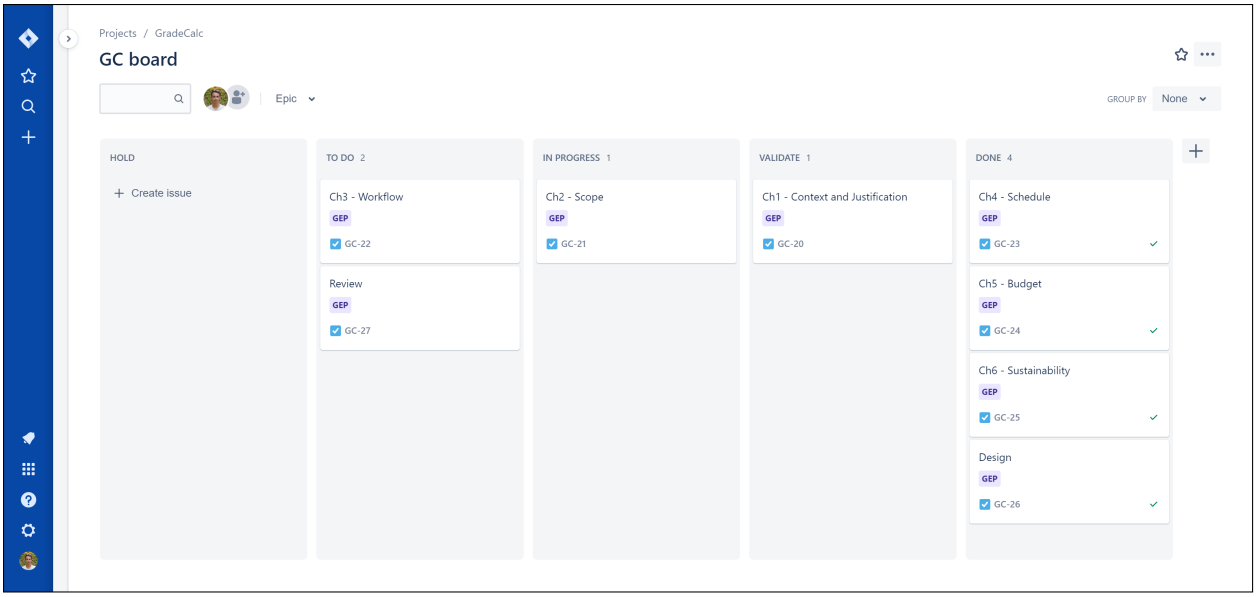


Figure 2.1 Kanban board in Jira (https://mauriciabad.atlassian.net/browse/GC)

The Kanban board is going to have the following columns:

- **HOLD**: Defined tasks that don't have to be done yet. Usually due to dependencies or external factors.
- **TO DO**: Tasks to do once there's room in the IN PROGRESS column.
- **IN PROGRESS**: Tasks being performed at the moment.
- **VALIDATE**: Apparently finished tasks that need to be tested and validated. If it's not right, it will be moved back to IN PROGRESS.
- **DONE**: Finished and validated tasks.

The DONE tasks are going to be removed from the Kanban to the backlog at the end of each sprint, and tasks from the backlog moved to the Kanban HOLD column.

The project is going to use GitHub Flow [21], a lightweight branch-based workflow. It's specially crafted for Continuous Integration apps and reduces overhead to the minimum. In addition, most open-source projects adopt it.

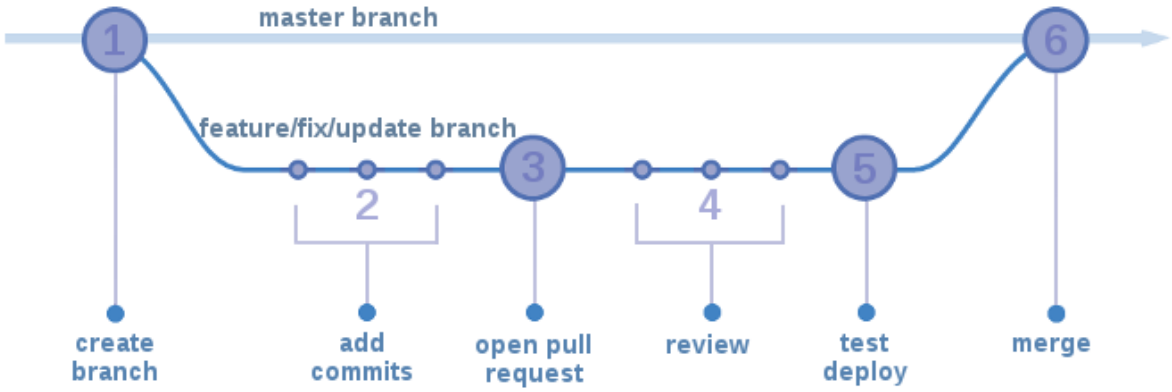


Figure 2.2 GitHub Flow [21]

2.2.2 Tools

To manage the project and validate that tasks are indeed completed, the following tools are going to be used:

- **GitHub**: Version control system and issue tracker.
- **Jira**: Project management tool.
- **Netlify**: Continuous deployment.
- **Travis CI**: Continuous integration it runs all tests.
  - **Jest**: Testing framework.
  - **Lighthouse**: Audits performance, accessibility, PWA[1], SEO[2] and more.

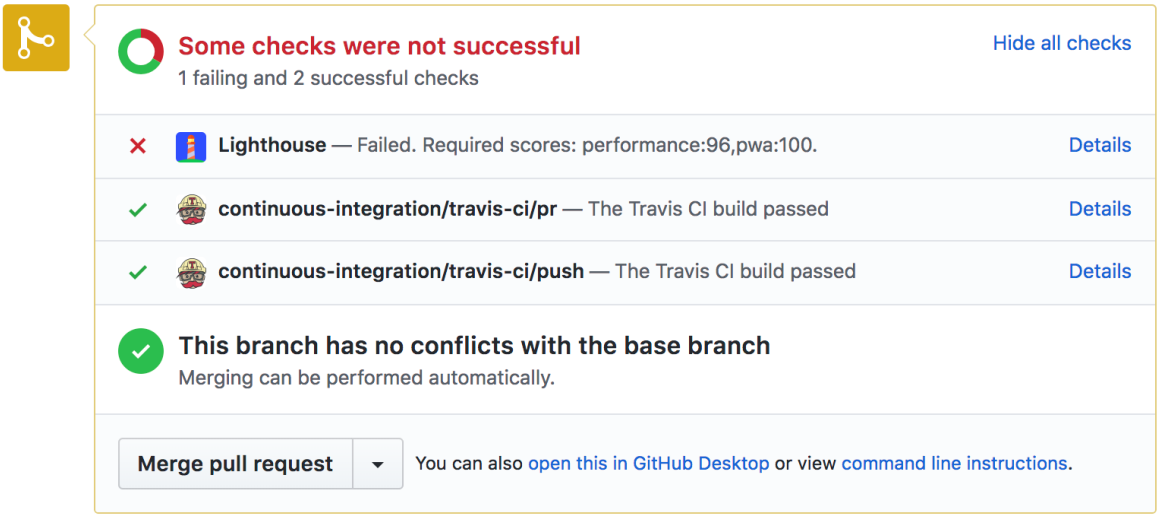


Figure 2.3 GitHub CI - Pull request UI

[1]Progressive Web App
[2]Search Engine Optimization

2.2.3 Changes in the workflow

The workflow established at the beginning of the project (chapter 2.2) has been followed but not in a strict way. No main changes have been made.

Jira ended up not being used. At the beginning I used it, but I ended up having a to-do list in a markdown text file and just checking the checkboxes once done.

This change was made because Jira was slowing down productivity. I had to fill in many fields when creating and editing the issues. In contrast, a simple checklist has no overhead at all.

The typical problem of using a to-do list is that when there are many tasks there's a lack of order and teamwork becomes hard. Because this project doesn't have many tasks and there are almost no dependencies between them using a to-do didn't introduce any problem, simplifying a lot the project management.

And, of course, if the project scales in the future and a to-do list becomes an obstacle, a tool like Jira will become again the standard to manage it.

2.3 Schedule

2.3.1 Estimates

The estimated project duration is **19 weeks** spending **25 hours/week**, a total of **475 hours**. From **February 17 to June 29**, working an average of **5 hours/day** excluding weekends.

Sprint plan

A sprint lasts 2 weeks and is composed of the following elements (Fig. 2.4), distributed in as shown in table 2.1:

- **Sprint Planning**: At the beginning of the sprint I will:

  - Analyse how the entire project is going and decide if any changes are needed.
  - Plan how the working hours will be spread through the sprint.
  - Define the tasks that will be done (approximately).

- **Sprint Review**: At the end of the sprint I will:
  - Analyse how the sprint went and decide if any changes are needed.
  - Compare the expected tasks to finish with actually finished tasks.

- **Decide and learn approach**: 5 hours throughout the sprint, at the beginning of every task, I will:
  - Analyse the requirements of the task and research the best approach.
  - Learn how to perform the approach if I don't.

- **Code**: 20 hours through the sprint I will:
  - Implement the features.
  - Other code-related actions like refactoring and reading code.
  - It also includes tasks like designing the UI, analyzing user analytics, designing the brand, advertising ...

- **Write tests**: 5 hours throughout the sprint, at the end of each task.

- **Manage Project**: 5 hours throughout the sprint I will:
  - Administrate repository: create/close merge requests, fix CI, close issues...
  - Administrate Kanban: create/edit tasks, move tasks, fill task details...
  - Miscellaneous small project-related tasks: configure firebase, buy the domain...

- **Write Memory**: 15 hours throughout the sprint.

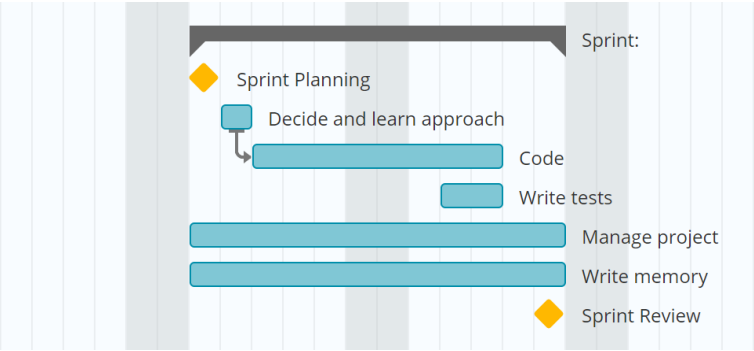


Figure 2.4 Sprint Gantt diagram

Sprint	**50h**
Decide and learn approach	5h
Code	20h
Write tests	5h
Manage project	5h
Write memory	15h

Table 2.1 Description of sprint tasks and estimations

2.3.2 Description of tasks

The project is split up into 4 blocks. See table 2.2.

- **Project Management**: Bound scope, create the planning, calculate the budget...
- **Development**: The app is developed.
- **Report**: Write the missing texts and review the entire document.
- **Defense**: Prepare the oral defense.

TASK	TIME
Project Management	**100h**
Document format	10h
Write Context	10h
Write Justification	10h
Write Scope	10h
Write Workflow	10h
Write Schedule	10h
Write Budget	10h
Write Sustainability	10h
Review and finish document	20h
Development	**275h**
Sprint 1	50h
Sprint 2	50h
Sprint 3	50h
Sprint 4	50h
Sprint 5	50h
Sprint 6	25h
Report	**50h**
Write report	25h
Review report	25h
Defense	**50h**
Write script	10h
Make slides	10h
Design demo	5h
Practice	25h
TOTAL	**475h**

Table 2.2 Description of tasks and estimations

2.3.3 Gantt diagram

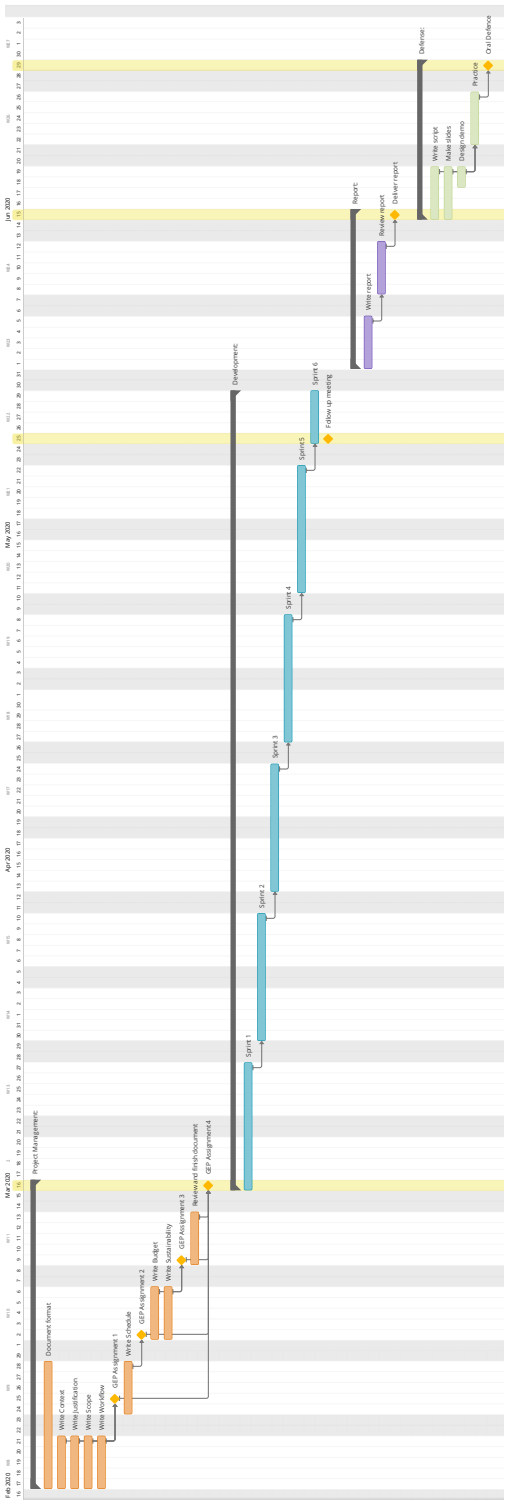


Figure 2.5 Gantt diagram

Dependencies

- **Project Management**: Each GEP assignment doesn't require the previous ones, except the last one that requires all of them.
- **Development**: Each sprint starts when the previous one finishes, they can't overlap and the order doesn't matter.
- **Memory**: To start reviewing the memory, it must be finished.
- **Defense**: To start practicing the oral defense, it must be finished.

2.3.4 Required resources

All this project is thought to be developed by one person, me. The only required material resource is a computer.

2.3.5 Risk management: alternative plans and obstacles

The main risks are explained in section 2.1.2. These are the mitigation of the problems:

Risk	Probability	Impact	Mitigations
No decent free option.	High	Medium	A. Check is the free tier is enough. B. Find alternatives. C. Pay if it's really important.
Not enough time.	Medium	High	A. Quit other activities. B. Reduce scope. C. Ask additional enrolment.
Too much to learn.	Medium	Medium	A. Do a workaround. B. Use something else. C. Skip the task.

Table 2.3 Risks and mitigations

2.3.6 Changes in the schedule

The main changes in the planning (chapter 2.3) have been in the Development phase. The other blocks (section 2.3.2) didn't have significant changes.

I followed the Kanban methodology, but I didn't do the formal sprint reviews I planned to. So because there were no sprint reviews there weren't sprints neither.

Instead of grouping the tasks in sprints, the project evolved into a more fluid methodology where I was completing tasks faster than I expected in general.

Not framing the tasks inside sprints had a positive impact on my performance because when a task was getting too complicated I could leave it aside, do other tasks and come back to it once some time passed. When retrying a task I had a fresher perspective and better ideas because I had time to come up with alternative solutions.

Another deviation is that I didn't write the memory while developing, so the last sprint (the 6th) was devoted to writing the follow-up report.

The follow-up meeting was scheduled on the Friday 29th of May.

The memory delivery date was scheduled on Thursday 25th of June, and the defense date on Thursday 2nd of July.

The writing time of the memory was stretched until the deadline, and the presentation was done in one week.

Updated Gantt diagram

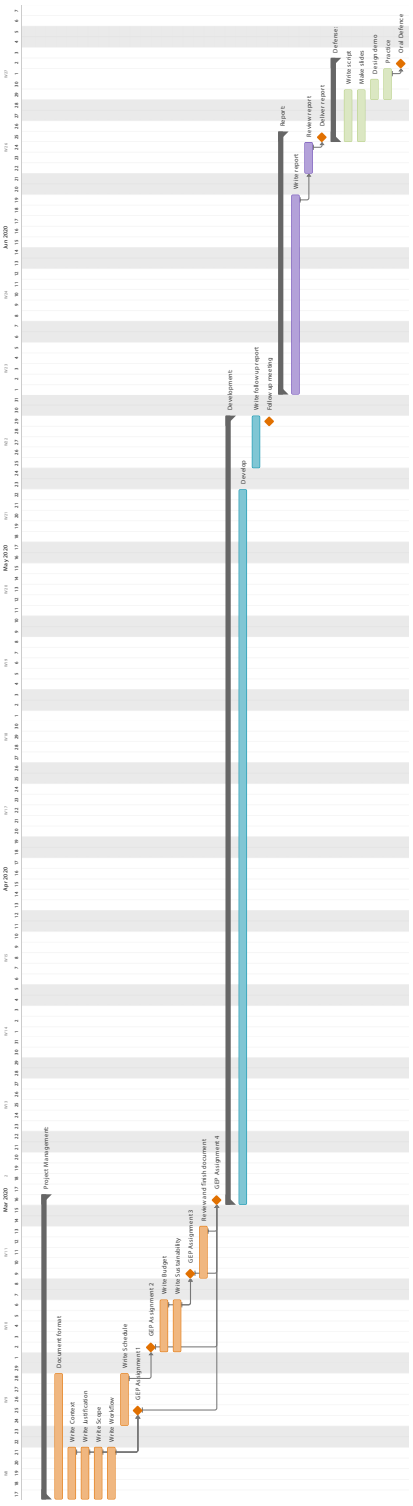


Figure 2.6 Updated Gantt diagram

2.4 Budget

2.4.1 Human resources costs

This is an open-source project driven by free contributions from the community. The project will start with only one contributor, me, but many people may get involved at some point. Because no one will get paid, the human resources costs will be **0€**. Nonetheless, I can estimate how much would this cost if people got hired to develop it, but I won't consider them in the total project sum.

Rol	**Weekly hours**	**Pay**	**Gross salary**	**Expense**
Project Manager	10h	25€/h	1000€/month	1300€/month
Front-end developer	20h	20€/h	1300€/month	2080€/month
Full-stack developer	20h	20€/h	1300€/month	2080€/month
Total	**50h**		**4200€/month**	**5460€/month**

Table 2.4 Ideal team

I estimated that a 30% of the gross salary is paid to Social Security.

$$Expense = GrossSalary \times 1.3$$

According to the Gantt diagram 2.5, this team would be working for 6 sprints. This translates to 3 months, meaning that the total human resources costs would be 16380€.

It's important to mention that they would do many more tasks than the ones I'll be able to do alone, so the planning would end up looking very different and the estimations I made inapplicable. Also, the Memory Writing, Report, and Defense blocks wouldn't make sense to have, so they would have even more sprints, increasing this cost.

Salaries

The total compensation per year of a **Project Manager** in Barcelona ranges between **22K€ and 58K€** [37]. I chose 25€/h (**52K€**) because it's in that range.

The total compensation per year of a **Front End Developer** in Barcelona ranges between **19K€ and 40K€** [35]. I chose 20€/h (**41.6K€**) that is slightly above that range because the developer will also take extra responsibilities.

The total compensation per year of a **Full Stack Developer** in Barcelona ranges between **19K€ and 42K€** [36]. I chose 20€/h (**41.6K€**) because it's in that range. I also thought that having similar salaries would be fairer for everyone on the team.

Why that team?

The Project Manager would be in charge of things like optimizing team performance, defining requirements, unblocking the others, having a larger picture of the project.

The Front-end developer would be an expert on the javascript framework chosen and answer questions that the other developer may need.

Because most of the work will be in the front-end and the back-end is very little, a full-stack would do the little back-end available first.

2.4.2 Hardware costs

Regarding hardware a similar thing happens, no specific hardware will be bought for this project and everything is going to be reused so hardware costs will be **0€** also. If the developers don't have it they just don't collaborate or someone else tests the app. The hardware I'm actually going to use is:

- My laptop, a Surface Pro 2018
- My phone, a Google Pixel 2 XL.
- Sporadically, iPhones from friends.
- Sporadically, laptops from friends.

But, again I can estimate the costs in case all the hardware had to be bought.

Item	Model	Amount	Cost	Useful life	Expense
Laptop	Apple MacBook Pro 2019	3	4500€	5 years	75€/month
Mobile	Google Pixel 4 XL	1	760€	3 years	21.11€/month
Mobile	Apple iPhone 11	1	630€	4 years	13.13€/month
Total			**3890€**		**108.24€/month**

Table 2.5 Ideal hardware

The VAT of the hardware could be saved if they are bought by a company. And if this is too much to pay, older or second-hand devices can be bought instead. Once the project finishes all hardware has to be sold to be afforded.

Why those devices?

To test the web app developers would need different operating systems and browsers. To test all browsers with a >2% usage and 81% global coverage, the web app needs to run in these browsers (Table 2.6).

All those browsers can be installed in macOS, iOS, and Android. So with a MacBook, an iPhone, and a Pixel, we can perform tests in all of them. Those devices are the newest ones when buying them, so they will last longer and get outdated latter.

BROWSER	USAGE
Mobile Browsers	**48.17%**
Chrome for Android 78	34.26%
UC Browser for Android 12.12	2.20%
iOS Safari 13.3	9.29%
Samsung Internet 10.1	2.42%
Desktop Browsers	**32.42%**
Chrome 79	16.86%
Chrome 80	10.66%
Firefox 70	2.05%
Safari 13	2.85%
Total	**80.79%**

Table 2.6 Browsers with more than >2% usage [4]

Developers will use the same laptop to code the app. They could do this with any laptop. All tasks can be done with these devices.

2.4.3 Software costs

This project free software as much as possible: Git, GitHub, Jira, Netlify, Firebase, Algolia, Travis CI, Figma, Google Analytics, VS Code, Overleaf, Linux... The only unavoidable cost is the domain registration. GradeCalc uses a .app domain which costs **15€/year**, slightly more than a .com or .net domain. This is the only software expense by now, but if the app gets a lot of traffic it may get extra charges.

Most of the services that this project uses are free until a certain point. From that point the price increases drastically. The following software needs to be looked after to avoid unexpected charges:

- **Firebase**: The free tear is enough, if the app doesn't waste reads/writes to firestore. https://firebase.google.com/pricing
- **Algolia**: May not be enough. https://www.algolia.com/pricing/
- **Jira**: The free tear is enough. https://www.atlassian.com/software/jira/pricing
- **Netlify**: The free tear is enough. https://www.netlify.com/pricing/

Service	**Plan**	**Cost**
Domain	Domain Renewal	15€/year
Firebase	Spark Plan	Free
Algolia	Community	Free
Jira	Free	Free
Netlify	Starter	Free
Total		**1.25€/month**

Table 2.7 Software expenses

2.4.4 Other costs

The project is going to be done with my current resources. But, again, simulating that a team would be hired, they would need a place to work. Barcelona offers coworking spaces for small companies or temporal projects, which fit perfectly this project. BarcelonaNavigator.com has an up-to-date list of recommended coworking spaces in Barcelona, from there I found SOWO that offers the Mini Flex rate [38] that fits our needs:

- **Flexible Desk**
- **1h Meeting rooms**
- **Access from 8:30-13:30h**
  or from 14:00-19:00h
- Electricity and water included
- Internet connection
- Common areas access
- Coffee, water and snacks free
- 50 B/W prints
- Central location

Service	**Rate**	**Company**	**Cost**
Co-working space	Mini Flex [38]	SOWO	140€/month + VAT
Total			**140€/month + VAT**

Table 2.8 Indirect expenses

2.4.5 Cost contingency

10% of the estimated costs would be saved to pay for unexpected expenses.

Category	Cost	Percentage	Contingency
Human resources	5460.00€/month	10%	546.00€/month
Hardware	58.24€/month	10%	5.83€/month
Software	1.25€/month	10%	0.13€/month
Other	140.00€/month	10%	14.00€/month
Total			**586€/month**

Table 2.9 Contingencies expenses

2.4.6 Unforeseen and extraordinary costs

The main risk is spending too much time learning instead of coding, to mitigate that, the planning sets on each sprint a reasonable time to learn. But maybe it's not enough.

In terms of the project, if this happened the less important tasks would be skipped. This solution wouldn't add any extra cost at the expense of a less complete app.

If the project was developed by a team and they go too slow to finish, they would work for one extra month or add a new member.

$$MonthCost = HumanResources + Hardware + Software + Other + Contingencies$$

Event	Event cost	Odds	Cost
Extra month	6295.49€	50%	3147.75€
Total			**3147.75€**

Table 2.10 Unforeseen expenses

2.4.7 Total costs

Category	Category cost	Amount	Total cost
Human resources	5460€/month	3months	16380.00€
Hardware	3890€	1	3890.00€
Software	15€/year	5year	75.00€
Other	140€/month	3months	420.00€
Contingencies	586€/month	3months	1758.00€
Unforeseen	3147.75€	1	3147.75€
Total			**25670.75€**

Table 2.11 Total expenses

2.4.8 Management control

To ensure the compliance of the initial budget, during the sprint review (2.3.1) at the end of each sprint I will calculate:

- Human resources costs deviation.
- General costs deviation.
- Unforeseen costs deviation.
- Total costs until this point.

The deviations are going to be calculated with the following formulas:

$$Budget = HumanResources + Hardware + Software + Others + Contingencies + Unforessn$$

$$Estimated = Budget - Contingencies - Unforeseen$$

$$Deviation = Estimated - Real$$

$$UsedBudget = \frac{CurrentlySpent}{Estimated} \times 100\%$$

The *UsedBudget* should grow suddenly at the beginning, due to the hardware and software acquisition, and then grow linearly until 100%. If we observe a different growing rate more money than estimated is being spent.

2.4.9 Changes in the budget

Although the planning changed a bit, the hours spent, the material used, and the services used are the same, so the costs don't vary.

2.5 Sustainability

2.5.1 Self-assessment

After answering the sustainability analysis survey, I've realized that in any project there is a sustainability component divided into three different dimensions: economic, environmental, and social. It has also made me think about the causes, consequences, and possible solutions about social, environmental, and economical problems that affect all kinds of IT projects. I also contemplated the importance of suggesting ideas and applying solutions to this project to make it more sustainable.

Not only I chewed over the importance of sustainability in a project like this one, but also I realized the importance I was giving to this topic was not enough in my past projects, as result of a little knowledge about topics like the environmental costs that IT products have over their lifespan. I neither knew how to measure the environmental impact of Information and communications technology in the world. And I also didn't know the existence of sustainable technologies applicable to a software project. I already knew the importance of introducing social justice, equity, diversity, and transparency in IT projects.

To conclude, this project has made me reflect on the the importance of taking into account sustainability when developing a project from all perspectives; economically, environmentally, and socially.

2.5.2 Economic Dimension

Regarding PPP: Reflection on the cost you have estimated for the completion of the project

The estimated cost of the project only includes essential resources, in this case, 0. I'm going to reuse all the hardware I currently own and sporadically use my friends' devices to test the app.

Regarding Useful Life: How are currently solved economic issues (costs...) related to the problem that you want to address (state of the art)?

All current solutions have a similar architecture, all of them are websites or apps without a database or a really simple one. So, their costs must be the domain registration and hosting.

Regarding Useful Life: How will your solution improve economic issues (costs ...) with respect to other existing solutions?

I'm going to use many free services at the beginning, and scale the project with also free or cheap services.

2.5.3 Environmental Dimension

Regarding PPP: Have you estimated the environmental impact of the project?

This project won't cause any environmental impact directly, it's an app to solve a quotidian problem more efficiently. It may have an indirect impact due to the services it uses, none of them has a bad reputation on managing its resources.

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources?

Yes, as explained earlier in section 2.4.2, all hardware is reused.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)? and how will your solution improve the environment with respect to other existing solutions?

Any of these projects produce any waste, and my solution is no different.

2.5.4 Social Dimension

Regarding PPP: What do you think you will achieve -in terms of personal growth- from doing this project?

I'll learn new technologies, tools, and techniques. Having real users using the app makes me set higher standards, giving me the motivation to do everything as good as possible.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?, and how will your solution improve the quality of life (social dimension) with respect other existing solutions?

This project is about making life easier for its users. The current solutions are very inconvenient and that's why this project was born. This topic is explained in-depth in section 1.

Regarding Useful Life: Is there a real need for the project?

Yes, there is. Although the problem that solves isn't critical.

2.6 Laws and regulations

The most important law that this project needs to follow is the GDPR[3][7]. It's the core of Europe's digital privacy legislation. Another similar regulation like the CCPA[4] can be considered, but the GradeCalc core users reside in Europe, where it doesn't apply.

To understand the GDPR it's really important to comprehend these two definitions[6]:

> (1) '**personal data**' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;
>
> (2) '**processing**' means any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction;

The only information stored in the database that could be considered as personal data by the GRPD[7] are the grades and the subject's creator id and name, explained in section 4.4.1. Other information stored such as subject evaluation criteria and general information like name, color, and date is not personal, thus it's not regulated by this law.

In order to be GDPR compliant the app needs to have:

- A message above the login button that says something similar to "by clicking the Sign in button, I accept the terms and conditions".

[3]General Data Protection Regulation
[4]California Consumer Privacy Act

- Have a page that explains how the personal data is processed (Privacy Policy).
- Have a page that explains how users can perform their rights, regarding GDPR, like erasure, portability, rectification, access, and restriction of the processing.

GradeCalc doesn't need a cookie banner because although using Google Analytics, the information that it collects is anatomized and basic.

If a user wants to perform any of the GDPR rights he/she just has to send an email to the provided address in the Privacy Policy page, and the administrator (in this case myself) will fulfill the request.

The third parties involved in this project are all GDPR compliant, so they can be used legally without any problem. Firebase is the most important external entity because it's the one processing the personal data, I had to verify that it is GDPR compliant[18] before using it.

Chapter Three

REQUIREMENTS AND SPECIFICATION

This section defines and explains the requirements of the application. They are obtained by analyzing how students manage their grades and coming up with solutions that would help them be more productive. They are also obtained by directly asking what potential users would want in the app.

3.1 Functional requirements

This section presents the requirements of the app in the form of user stories (US).

US 1 - Calculate needed grade

As a user, **I want to** calculate needed grade to pass a course, **so I can** decide how much to study.

US 2 - Calculate final grade

As a user, **I want to** calculate my final grade assuming a 0 in the remaining exams, **so I can** know if I already passed.

US 3 - Current grades

As a user, **I want to** see my current grades, **so I can** view my performance.

US 4 - Sign-up

As a user, **I want to** sign-up with an external authentication provider, **so I can** register faster and more conveniently..

US 5 - Log-in and Log-out

As a user, **I want to** log-in and Log-out, **so I can** decide when to use my account.

US 6 - Save grades in my account

As a user, **I want to** save grades in my account, **so I can** share them between my desktop and mobile.

US 7 - Save grades in the device

As a user, **I want to** save grades in the device, **so I can** use the app without an account or offline.

US 8 - Create subject

As a user, **I want to** create subject, **so I can** use the app with any of my courses.

US 9 - Edit subject

As a user, **I want to** edit subject, **so I can** correct mistakes or tweak values.

US 10 - View subject

As a user, **I want to** view subject details, **so I can** see clearly all it's information.

US 11 - Delete subject

As a user, **I want to** delete subject, **so I can** focus on other subjects.

US 12 - Search subjects

As a user, **I want to** search subjects, **so I can** avoid creating them and save time.

US 13 - Multiple evaluations

As a user, **I want to** be able to set have multiple evaluation formulas on subjects, **so I can** represent my subject's evaluation precisely.

US 14 - Dashboard

As a user, **I want to** have a dashboard with my subjects, **so I can** handily input my grades.

US 15 - Auto-evaluate

As a user, **I want to** automatically reevaluate the calculations, **so I can** see the right values all the time.

US 16 - Auto-save

As a user, **I want to** automatically save my changes, **so I can** save time saving and don't forget to save.

US 17 - Auto-sync

As a user, **I want to** automatically syncs my changes with my account, **so I can** have a consistent state always.

US 18 - Responsive UI

As a user, **I want to** have UI designed for mobile and desktop, **so I can** use the app in any screen size.

US 19 - Use from browser

As a user, **I want to** use the app from the browser, **so I can** use it without downloading it.

US 20 - Lunch icon

As a user, **I want to** have a lunch icon on the home screen, **so I can** quickly open the app.

US 21 - Offline

As a user, **I want to** use the app offline, **so I can** use it without depending on my internet connection.

US 22 - SEO

As a user, **I want to** find the app in Google, **so I can** open it without knowing the domain.

US 23 - Tutorial

As a user, **I want to** have a tutorial, **so I can** learn to use the app the first time I visit it.

US 24 - Undo

As a user, **I want to** be able to undo some actions, **so I can** fix mistakes.

US 25 - Notifications

As a user, **I want to** be notified with tips and suggestions inside the app, **so I can** take advantage the app, and use it correctly.

US 26 - Accessible

As a user, **I want to** still be able to use the app if I have a disability, **so I can** use the app without complications.

US 27 - Updates

As a user, **I want to** receive frequent updates, **so I can** enjoy the latest features.

US 28 - Analytics

As a user, **I want to** give data of my usage up (anonymized), **so I can** improve the app.

US 29 - Open-source

As a user, **I want to** have the main code publicly available (open-source), **so I can** validate that the app does what it states.

US 30 - Background activity

As a user, **I want to** be aware of background activity, **so I can** better predict the behavior of the app.

US 31 - In Spanish

As a user, **I want to** have the app entirely in Spanish, **so I can** understand it.

US 32 - Legal page

As a user, **I want to** have a page with legal information, **so I can** solve any doubts.

3.2 Non-Functional requirements

This section highlights the 4 non-functional requirements (NFR) that the application must meet. These requirements shape the app so much that all features aim to satisfy them.

NFR 1 - Reliable

- **Description**: The functionalities have to work consistently and predictably, in other words, the app has to work as the user expects. For example, it doesn't lose data, it doesn't miscalculate, it works on all the user's devices, it works offline...
- **Justification**: The app has to work properly to be useful, otherwise there's no point in using it.
- **Acceptance criteria**: Ask a sample of users to evaluate, from 1 to 5, how much they agree to the following statements. A mean greater than 2.5 needs to be achieved.
  - The app works as I expect.
  - The app does the calculations without mistakes.
  - I can use the app from any of my devices.
  - The app has never lost any of my data.
  - I can use the app with the subjects I course.

NFR 2 - Easy to use

- **Description**: The app has to be intuitive and accessible, in other words, easy to learn and easy to use.
- **Justification**: If the app is easy to use the users will save time and overall be more satisfied.

- **Acceptance criteria**: Ask a sample of users to evaluate, from 1 to 5, how much they agree to the following statements. A mean greater than 2.5 needs to be achieved.
  - The app is easy to learn
  - The app is easy to use
  - The app responds quickly to my actions.
  - I know how to use the app in general.
  - I know how to create a subject
  - I understand all the information in the dashboard.

NFR 3 - Engaging

- **Description**: The app must engage users and ensure that they are enjoying it and will recommend it.
- **Justification**: Happy users will attract new users helping the app grow. If the users are engaged, that means that the app is useful to them.
- **Acceptance criteria**: Ask a sample of users to evaluate, from 1 to 5, how much they agree to the following statements. A mean greater than 2.5 needs to be achieved.
  - I enjoy using the app.
  - I would recommend the app.
  - There's people using the app thanks to me.
  - The app helps me to achieve my goals.
  - The app helps me to pass the subjects.
  - I open the app frequently.
  - I open the app when I receive a new grade.
  - GradeCalc's logo, name, and style are easy to remember.
  - I like GradeCalc's logo, name, and style.

NFR 4 - Cheap to produce

- **Description**: The app has to be as cheap as possible to develop and maintain.
- **Justification**: This is a small hobby project developed by a student that is not willing to spend money on it unless it's imperceptible or would make a big difference.
- **Acceptance criteria**: the app must have a profit greater than -50€/year and cost less than 100€ to develop.

3.3 Conceptual model

The app has two main entities, subjects, and students, they are marked with thicker borders.

- Students are the users of the app and they can create and course subjects.
- Subjects are what students course that lead to an examination or qualification. A subject is taught in a faculty, and a faculty belongs to a university. Also, subjects are done in a course (usually a semester).
- Faculty, university, and course are not relevant for the app's requirements, they are just related concepts, so in the database, they'll be just attributes of a subject, not standalone classes.
- A subject contains one or more evaluations and one or more exams. An evaluation contains exams with weights.

The main takeaway from the conceptual model (Fig. 3.1) is that students create and course subjects, and subjects have evaluations with exams and basic information to be searched like short name, full name, university or course.

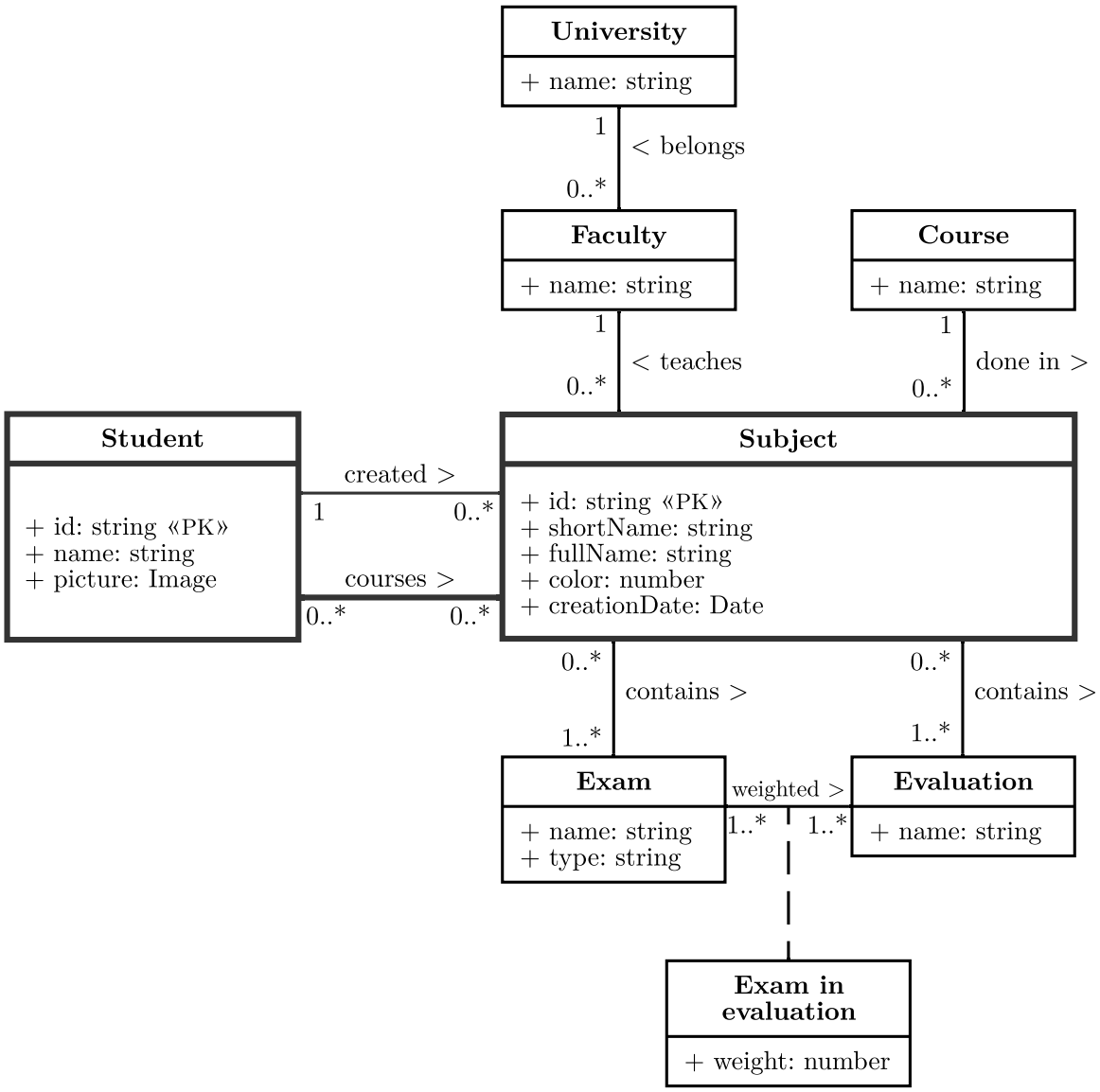


Figure 3.1 Conceptual model's UML Diagram

3.4 Tasks

This section defines all the tasks to do in order to fulfill all the requirements that the application must meet. These tasks are very specific and can be directly placed into the Kanban board. They are organized by screens and components of the app. This section will help understand all the features implemented and it's level of detail.

Dashboard

A user wants to see and interact with his subjects and grades.

(Task 1) It is the homescreen of the app.

(Task 2) It contains subject cards.

(Task 3) When the app is opened, the subjects are retrieved from the device's local storage. And a non blocking spinner is shown while the subjects are being loaded.

(Task 4) When the app is opened and the user is logged-in, subjects are synced with his account. While this happens a non blocking spinner appears.

(Task 5) The subject cards are sorted alphabetically by *shortName*.

(Task 6) When there are no subject cards, the add button starts a looping animation, similar to the rubberBand from the Animate.css[11] library, to draw attention.

(Task 7) When there are no subject cards, the empty space is replaced with an explanatory text about the app. At the end of the text, there are instructions to add the first subject. This text serves two purposes, inform the user, and optimize the search engine indexing of the app.

(Task 8) It has a header bar on top of the subject cards and in the top of the screen.

(Task 9) The header bar has, in the left, an icon that when clicked goes to the login screen.

(Task 10) The header bar has, in the center, the app name.

(Task 11) The header bar has, in the right, an icon that when clicked goes to the search subject screen.

(Task 12) It is responsive and the subjects are rearranged depending on the screen width.

(Task 13) When all subject cards displayed are passed, a present appears at the bottom of the dashboard along with a congratulating text.

(Task 14) When the gift is clicked a funny congratulating video appears at the bottom of the screen and plays automatically in a loop.

(Task 15) If the app meets the PWA installation criteria[30], a notification shows up with the install action.

Subject card

A user wants to work with a specific subject in the dashboard.

(Task 16) It can be in collapsed or expanded state.

(Task 17) A subject card is collapsed by default.

(Task 18) When a subject card is expanded it pushes the other ones instead of overlapping them.

(Task 19) When a subject card is clicked where there's no intractable elements, it toggles its state from collapsed or expanded. The intractable elements are the subject bar, exam input, evaluation selector, edit icon and delete button.

(Task 20) There's an animation for when a subject card is collapsed or expanded.

(Task 21) A collapsed card doesn't show the exams' inputs nor the evaluation selector. And a expanded one does.

(Task 22) It has a graphical representation of the selected evaluation, called *subject bar*. The representation is a bar divided in sections where each section represents an exam and it's length is proportional to the exam weight.

(Task 23) In the subject bar, hovering a section shows a tooltip displaying the exam weight in percentage.

(Task 24) In the subject bar, each section contains the exam name and the grade. If the exam is undone, it contains the necessary grade instead.

(Task 25) In the subject bar, the done exams have the subject's color as background color.

(Task 26) In the subject bar, the undone exams have a light background color.

(Task 27) In the subject bar, the exams are sorted by same order that they are in the exam array of the evaluation. Not alphabetically nor per weight.

(Task 28) In the subject bar, when a section is clicked it focuses the exam's input. If the card is collapsed it's also expanded.

(Task 29) In the subject bar, when a section is to narrow to see the full grade or name, the first characters are visible. Instead of the characters in the middle.

(Task 30) It has the final grade in big text.

(Task 31) The final grade is calculated by multiplying each exam grade by its weight and summing all the values. If an exam is undone a 0 is used in the calculation.

(Task 32) The final grade is green if it's greater or equal than five and red otherwise.

(Task 33) The final grade is rounded to two decimals.

(Task 34) When the final grade passes from a number lower than five to one greater or equal to five a confetti animation is triggered. The animation shoots confetti over the subject card to congratulate the user.

(Task 35) It has an evaluation selector if there are more than one.

(Task 36) When the evaluation is changed, the card updates instantly it's information (exam inputs, subject bar, final grade...).

(Task 37) If an exam with the same name is in more than one evaluations, the grade is shared among them. It means that the same exam can be used in different evaluations.

(Task 38) The application remembers what is the selected evaluation. If the user closes and opens the app again, the selected evaluation doesn't change.

(Task 39) It has the subject's *shortName* in big text.

(Task 40) It has an input field for each exam in the selected evaluation.

(Task 41) The inputs are grouped by their type attribute.

(Task 42) The type is styled like a subtitle.

(Task 43) Next to the type, there's the sum of the weights of the exams with that type in percentage and rounded.

(Task 44) The types are sorted by the same order that they are in the exam array of the evaluation. Not alphabetically nor per weight.

(Task 45) The exam inputs inside the type are sorted by same order that they are in the exam array of the evaluation. Not alphabetically nor per weight.

(Task 46) The exam input has different styles depending on it's vaule.

(Task 47) When the exam input has a valid value, it's considered a *done exam*. It has the subject's color as background.

(Task 48) When the exam input has an empty value, it's considered an *undone exam*. It has a gray background and it's placeholder is the necessary grade to pass.

(Task 49) When the exam input is focused it has a black border.

(Task 50) When the exam input's value isn't a number nor empty, it has a gray background with red borders. And it's considered an *undone exam*.

(Task 51) When the exam input's value is a number but lower than 0 or greater than 10, it has the subject's color as background with red borders. And it's considered a *done exam*.

(Task 52) If the exam has weight 0, it's hidden.

(Task 53) At the left of the input there's a label with it's name.

(Task 54) The exam input only accepts numbers (digits, periods, commas, plus, minus and e for exponential). And in digital keyboards shows a numeric keyboard.

(Task 55) The necessary grade is the necessary grade to receive in all the remaining undone exams to get a 5 in the final grade.

(Task 56) Every time a grade is changed other information is updated instantly.

(Task 57) The necessary grade is updated.

(Task 58) The final grade is updated.

(Task 59) If the user is logged-in, the new grade is sent to his account.

(Task 60) The exam inputs take a reasonable space for their expected value length, usually less than 6 characters.

(Task 61) It has a edit icon.

(Task 62) The edit icon has a hover animation.

(Task 63) When clicked opens the edit subject screen of the current subject. With the title “Edit subject” and the action button “Save”.

(Task 64) If the user finally edits the subject, the subject card’s information is updated instantly (name, color, exam inputs...).

(Task 65) If the user finally edits the subject, and he’s logged-in, the subject is also edited in his account.

(Task 66) It has a delete button.

(Task 67) Hovering the card displays the delete button.

(Task 68) The delete button has tree animations. On show it fades in and scales up to normal. On hover it shakes. On click it scales down.

(Task 69) When the delete button is clicked the card is deleted.

(Task 70) When a card is deleted, the other cards move to their new locations and sizes in a smooth animation.

(Task 71) When a card is deleted, a notification appears saying "You deleted *subject’s shortName*" and a action button says "undo". When the action is clicked the subject is restored, like nothing happened. If the user is logged-in, undoing the action re-adds the subject to his account.

(Task 72) When a card is deleted, if the user is logged-in, the subject is also deleted from his account.

Register, Log-in, and log-out

A user wants to register, log-in, or log-out.

(Task 73) When the user is logged-in, the user icon in the dashboard changes to his profile picture.

(Task 74) When the user is logged-in, and a non blocking spinner is shown while the dashboard's subjects are being synced with the ones in the user's account.

(Task 75) When the user is not logged-in, a notification shows up with the log-in action. Clicking the action does the same as clicking the log-in button in the log-in screen.

(Task 76) The log-in screen is a popup in desktop and mobile.

(Task 77) It has the user's profile picture. If the user is not logged in, the picture is a default one.

(Task 78) It has the user's profile name. If the user is not logged in, the name is a default one (Anonymous).

(Task 79) It has a short text listing the benefits of logging-in.

(Task 80) It has a log-in or log-out button, green and red respectively, depending on weather the user is logged in or not.

(Task 81) The log-in button registers the user, without extra steps, if he is not registered yet.

(Task 82) The login system uses Google's log-in.

(Task 83) When the app is opened, if the user was logged-in, he's logged-in automatically.

Edit subject

A user wants to edit a subject

(Task 84) The screen title can be changed.

(Task 85) The action button label can be changed.

(Task 86) When the action button is clicked the all inputs are read and a new subject object is crated. This object is returned to the previous screen, delegating the responsibility.

(Task 87) It has a grid to edit the evaluation.

- (Task 88) In the grid, each row represents an exam, and each extra column an evaluation.
- (Task 89) The fixed columns are the following: exam name, exam category and exam grade.
- (Task 90) There are extra columns, one per each evaluation. The first row of the column is the evaluation name, and each row the weight of the corresponding exam.
- (Task 91) The weight are in percentage without the % symbol.
- (Task 92) If a weight is empty, it's considered 0.
- (Task 93) Below each evaluation column there's a label indicating the sum of all the weights of the evaluation. This way the users can check quickly that it adds to 100%.
- (Task 94) In the grid, rows and columns can be added and removed.
    - (Task 95) There's a faded extra row at the bottom, and a faded extra column at the left. If the user types something in there, it

becomes clear and part of the grid. A new faded row or column is added to let the user repeat the process.

(Task 96) If the user deletes all the values of a row or column, it disappears.

(Task 97) The grid can be easily navigable with the keyboard.

(Task 98) If the grid is to big to fit the screen, it can be scrolled.

(Task 99) If the user clicks the action button and there isn't at least 1 exam a notification appears and the action is ignored.

(Task 100) If the user clicks the action button and there isn't at least 1 evaluation a notification appears and the action is ignored.

(Task 101) It has a section with the subject information fields.

(Task 102) It has a field for: *shortName*, *longName*, *course*, *faculty* and *university*.

(Task 103) It has a field *color* where the allowed colors are displayed and the user checks the one he wants. By default a random one is selected.

(Task 104) Each field has it's label always visible.

(Task 105) When the label is clicked the input is focused.

(Task 106) The inputs have their sizes proportionate to the expected content length.

(Task 107) The fields *shortName*, *longName*, *course*, *faculty* and *university* can't be empty.

(Task 108) The fields have the same style as the exam inputs.

(Task 109) The field *color* must have always be chosen.

(Task 110) It has the *creationDate* and *creatorName* but they are not editable. They look like a text instead of a disabled input. The *creationDate* is

in format `d/m/yyyy`.

(Task 111) When a field is invalid it has a red border.

(Task 112) If the user clicks the action button and there are invalid fields a notification with appears with the name of the invalid field in the message and the action is ignored.

(Task 113) It looks like a popup in desktop and like a full page in mobile. It can be scrolled in both styles.

(Task 114) If the user clicks the back button of the browser, the popup blurred background or the back arrow in the header bar, the changes are discarded and the app navigates to the last screen.

Search subject

(Task 115) It has a big search input.

(Task 116) The search input has an example as placeholder.

(Task 117) Below the the search input there's a short text explaining what field are searchable.

(Task 118) The search input is focused automatically when this screen is opened.

(Task 119) When the search input has text, a cross icon appears at the right and when it's clicked it removes the input's content.

(Task 120) When the user types in the search input, the results appear in the screen.

(Task 121) When the user types a character, a search is performed automatically and the results displayed and updated instantly as the user writes.

(Task 122) If there are no results, a text appears saying that there are no results.

(Task 123) When there are results, the create button disappears.

(Task 124) The search engine accepts typos.

(Task 125) The results are sorted by relevance.

(Task 126) If there are results they are displayed in a list.

(Task 127) It contains a checkbox

(Task 128) The checkboxes are disabled by default.

(Task 129) When the subject of the result is already added to the dashboard, the checkbox is checked but disabled.

(Task 130) Clicking the checkbox checks or unchecks it with an animation.

(Task 131) When the search query changes, the checked subjects stay checked although they are not visible. If they appear in another query they'll still be checked.

(Task 132) It contains the *shortName*, *longName*, *course*, *faculty* and *university* attributes as text.

(Task 133) The matching characters from the query are highlighted.

(Task 134) It contains a edit icon that when clicked opens the edit subject screen of the current subject. With the title "Edit and add subject" and the action button "Add" that adds the subject directly with the changes. When clicked also checks the result checkbox.

(Task 135) When the action button is clicked the checked subjects are added to the dashboard.

(Task 136) If the user is logged-in, when the action button is clicked the checked subjects are also added to the user's account.

(Task 137) When the action button is clicked, while the subjects are being added, a non blocking spinner informs that the subjects are being added in the background.

(Task 138) It has a create button.

(Task 139) There's a short explanatory text of what creating a new subject means.

(Task 140) When the create button is clicked it opens the edit subject screen of an empty subject.

(Task 141) The edit screen has as title "New subject" and as action button "Create" that creates and adds the subject to the dashboard.

(Task 142) If the user is logged-in, the subject is also created in the database.

(Task 143) While the subjects are being added, a blocking spinner informs that the subjects are being created.

(Task 144) The create button doesn't draw a lot of attention. To prevent unnecessary clicks.

(Task 145) It looks like a popup in desktop and like a full page in mobile. It can be scrolled in both styles.

(Task 146) If the user clicks the back button of the browser, the popup blurred background or the back arrow in the header bar, the changes are discarded and the app navigates to the last screen.

Notifications

A user wants to be notified of certain things and be able to respond with an action.

(Task 147) It has a message that can be changed.

(Task 148) It has an action name and callback function that can be changed.

(Task 149) It appear for 8 seconds by default and can be changed.

(Task 150) It has a slide up animation when appears.

(Task 151) It has a slow fade out animation to inform the user how much time is remaining until it disappears, instead of disappearing instantly.

(Task 152) If it has a action, after disappearing it can still be clicked for 0.5 seconds, although it's not visible. It's a grace second, in case the user clicked it too late.

(Task 153) It's placed on top of everything at the bottom center of the screen.

Spinners

A user wants to be notified of background processes.

(Task 154) The non blocking spinners appear at the bottom of the dashboard.

- (Task 155) They are discrete
- (Task 156) They have a message.

(Task 157) The blocking spinners, are placed on top of everything (in the z dimension).

- (Task 158) They have a message.
- (Task 159) They have a long and entertaining animation. To make the waiting more pleasant.
- (Task 160) If they take more than the expected time, a message shows up telling that something went wrong.

General

(Task 161) The app's UI is entirely in Spanish.

(Task 162) The app is a Progressive Web App (PWA).

(Task 163) The app is published in a domain.

(Task 164) The app is published in the Google Play Store.

(Task 165) The app can work offline (exept logging-in, and searching subjects).

(Task 166) The subjects are stored in the device's local storage and if logged-in in his account.

(Task 167) The app is the first google result when searching "GradeCalc" and "Grade Calc".

(Task 168) The main source code is open-source.

(Task 169) The user account information can only be accessed by the own user.

(Task 170) The subject's in the database can't be edited or deleted by users, but they can create new subjects.

(Task 171) The decimal separator is always displayed with a period. When the user types a number can use indistinctly a period or a comma.

(Task 172) The site uses Google Analytics.

(Task 173) The app's deployment is automated.

(Task 174) The app's testing is automated.

(Task 175) The app has icons for most of the platforms (favicon, android, iOS...).

(Task 176) The app has a privacy and policy page.

Chapter Four

SOFTWARE DESIGN

4.1 Software architecture

This section explains the application's internal structure.

4.1.1 Logical architecture

The application's architecture is multilayered, it uses the widespread three-tier architecture. This is a clear explanation of the three-tier architecture:

> Three-tier architecture is a client-server software architecture pattern in which the user interface (presentation), functional process logic ("business rules"), computer data storage, and data access are developed and maintained as independent modules, most often on separate platforms. [9]

In the case of this project, the presentation layer is the most important one with the majority of the code. The app focuses a lot on productivity, so most of the GradeCalc's features are on its interface. The domain layer has the app's logic, and the data layer is composed of the database structure and thanks to Firebase it doesn't have any code.

Design patterns

Design patterns are typical solutions to common problems in software design. Each pattern is like a blueprint that you can customize to solve a particular design problem in your code.[23]

In this section, I'm going to explain and justify the design patterns that the software of this project uses. Identifying the design patterns leads to better code because they are proved to work. Coding a custom solution certain cases may be forgotten thous introducing bugs is more frequent.

Singleton

Singleton is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance.[25]

This pattern is used in the database connection object as recommended by the Cloud Firestore documentation[17]. The code doesn't enforce the use of a singleton, but the constructor is only called once. This approach is considered a better practice because singletons inherently cause code to be tightly coupled. This makes faking them out under test rather difficult in many cases. But not enforcing the "singletoness" of the class mitigates its issues.

Proxy

Proxy is a structural design pattern that lets you provide a substitute or placeholder for another object. A proxy controls access to the original object, allowing you to perform something either before or after the request gets through to the original object.[24]

This pattern is used to interact with the database. There's a class that provides methods for uploading data. This is good for testing because the class can be mocked to not use the real database connections when testing certain code.

Null object

Null object is a behavioral design pattern that avoids null references by providing a default object.

This pattern is used in the login system, when the user is not logged in, the user information object contains a default name and picture. This name and picture are displayed in the user popup.

Servant

Servant or Utility class is a structural design pattern that defines a class with common functionality for a group of classes. The helper classes generally have no objects hence they have all static methods that act upon different kinds of class objects.

This pattern is used specially for mathematical functions like: `random(min, max)` or `round(number, decimalPlaces)`; and other kind of generic tasks like: `getRandomID()` or `isEmpty(object)`.

Adapter

Adapter is a structural design pattern that allows objects with incompatible interfaces to collaborate.[22]

This pattern is seamlessly used when displaying information from JSON format into HTML format because the browser can only display HTML. For example, it is used when the user searches for a subject the response obtained is a JSON object that is transformed into HTML to be displayed.

4.1.2 Physical architecture

The physical architecture is heavily influenced by the choice of using Firebase. Doug Stevenson explains how firebase shapes your app's architecture very well in his Medium article "What is Firebase? The complete story, abridged."[40], this is an extract of his article:

> Firebase is a toolset to "build, improve, and grow your app", and the tools it gives you cover a large portion of the services that developers would normally have to build themselves, but don't really want to build, because they'd rather be focusing on the app experience itself. This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud, and scale with little to no effort on the part of the developer.
>
> This is different than traditional app development, which typically involves writing both frontend and backend software. The frontend code just invokes API endpoints exposed by the backend, and the backend code actually does the work. However, with Firebase products, the traditional backend is bypassed, putting the work into the client.

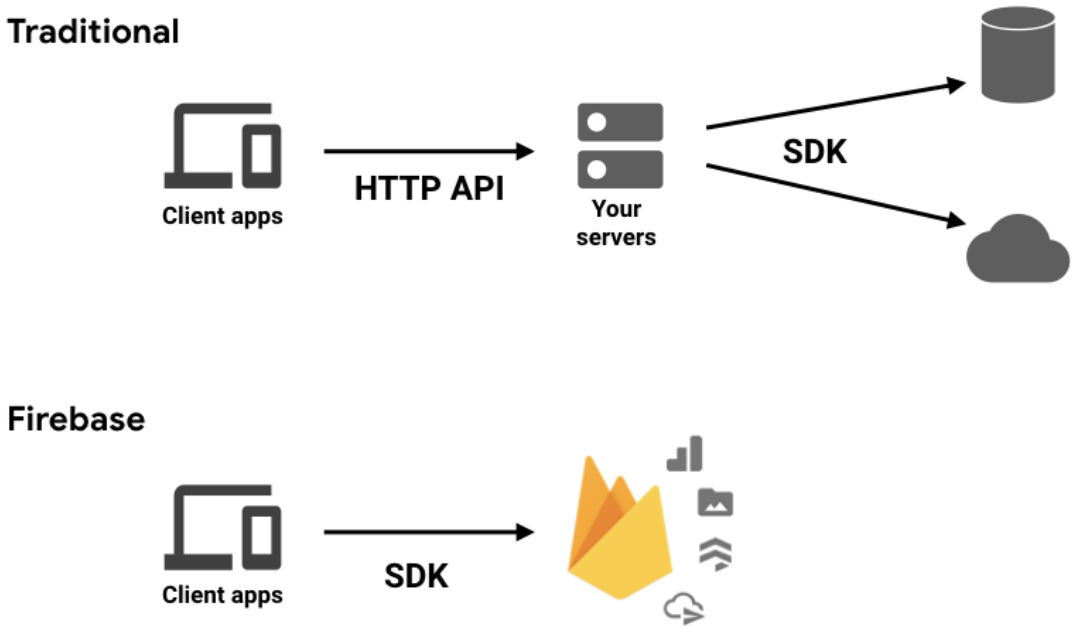


Figure 4.1 Traditional vs Firebase architecture. [40]

The Firebase suite offers many individual products (Fig. 4.2). The ones that this project uses are authentication and database.

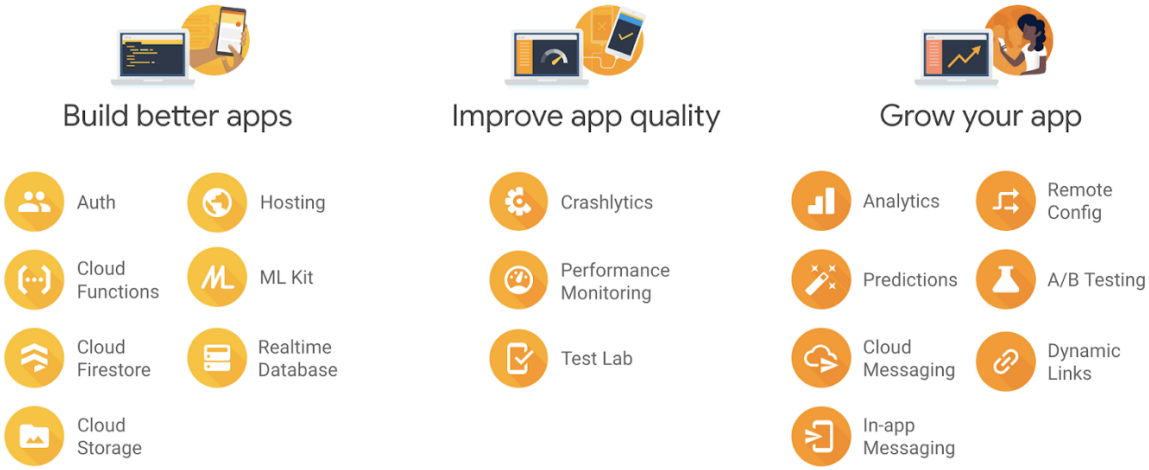


Figure 4.2 Firebase products. [40]

The 3 application layers are in the front-end. There we use Gulp to generate the needed and optimized static HTML, CSS, and JS files. These optimized files are hosted in Netlify.

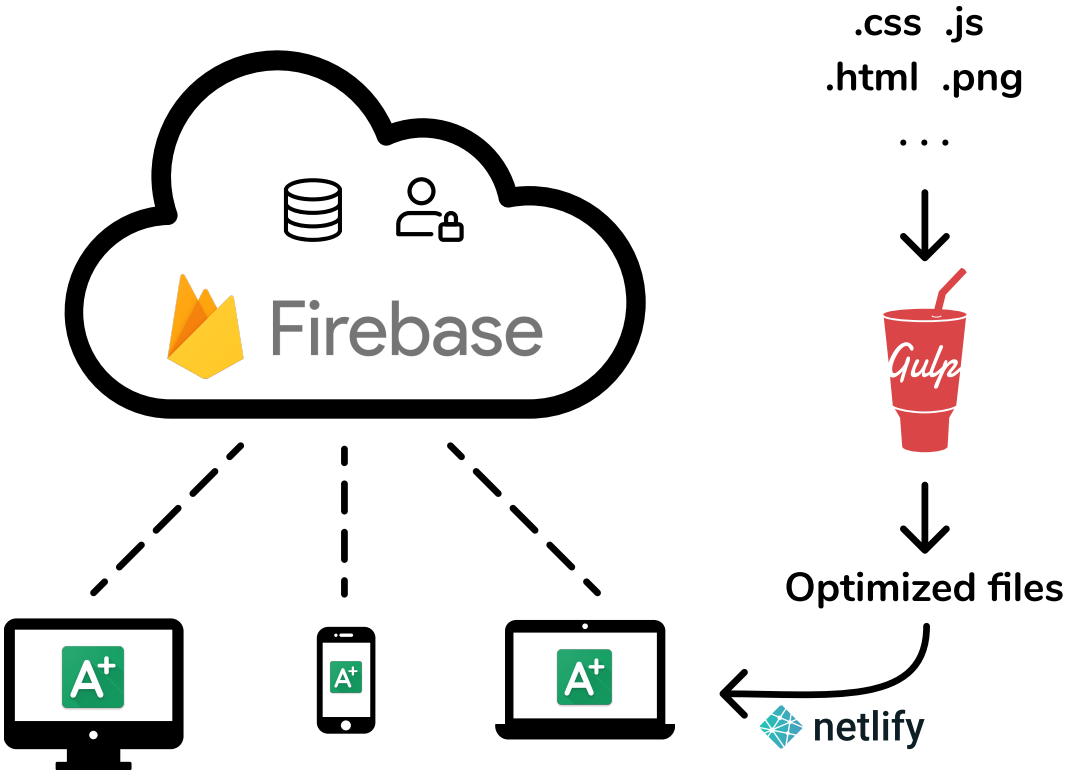


Figure 4.3 GradeCalc architecture

GradeCalc is a Progressive Web App (PWA) instead of a native app. This is how Mozilla Web Docs defines PWAs [8]:

PWAs are web apps developed using a number of specific technologies and standard patterns to allow them to take advantage of both web and native app features. There are some key principles a web app should try to observe to be identified as a PWA. It should be:

- **Discoverable**, so the contents can be found through search engines.
- **Installable**, so it can be available on the device's home screen or app launcher.
- **Linkable**, so you can share it by simply sending a URL.
- **Network independent**, so it works offline or with a poor network connection.
- **Progressive**, so it's still usable on a basic level on older browsers, but fully-functional on the latest ones.
- **Re-engageable**, so it's able to send notifications whenever there's new content available.
- **Responsive**, so it's usable on any device with a screen and a browser—mobile phones, tablets, laptops, TVs, refrigerators, etc.
- **Safe**, so the connections between the user, the app, and your server are secured against any third parties trying to get access to sensitive data.

Offering these features and making use of all the advantages offered by web applications can create a compelling, highly flexible offering for your users and customers.

GradeCalc is PWA mainly **because it saves up a lot of development time** at the expense of some, really specific, capabilities. A PWA can run in any of the most popular operating systems (Android, iOS, Windows, macOS, and Linux), contrary to a native app that only runs in its respective OS. So, the same code will work everywhere.

4.2 Presentation layer's design

This section explains the user interface and user experience design of the app.

4.2.1 User Interface

In this subsection, we'll go through the screens of the application showcasing what features do they implement and explaining how they work. All the screenshots are of the real application developed and working in production.

Dashboard and welcome

The dashboard is the main screen of the app and also the entry point. Every time a user opens the app it goes to this screen.

On the top of the screen there's the header bar with three elements: the account_circle button that goes to the login screen (Fig. 4.13), the app name in the middle, and the + button that goes to the search subject screen (Fig. 4.8).

The account_circle button turns into the user's profile picture if he's logged in.

On the main part of the screen, there are all the subjects that the student is coursing in the form of cards (Fig. 4.6). By default all the cards are collapsed, like the *XC* and *IDI* ones in the example (Fig. 4.6). The subject cards are explained in depth in section 4.2.1.

In general, this is how to interpret what's going on in Figure 4.4:

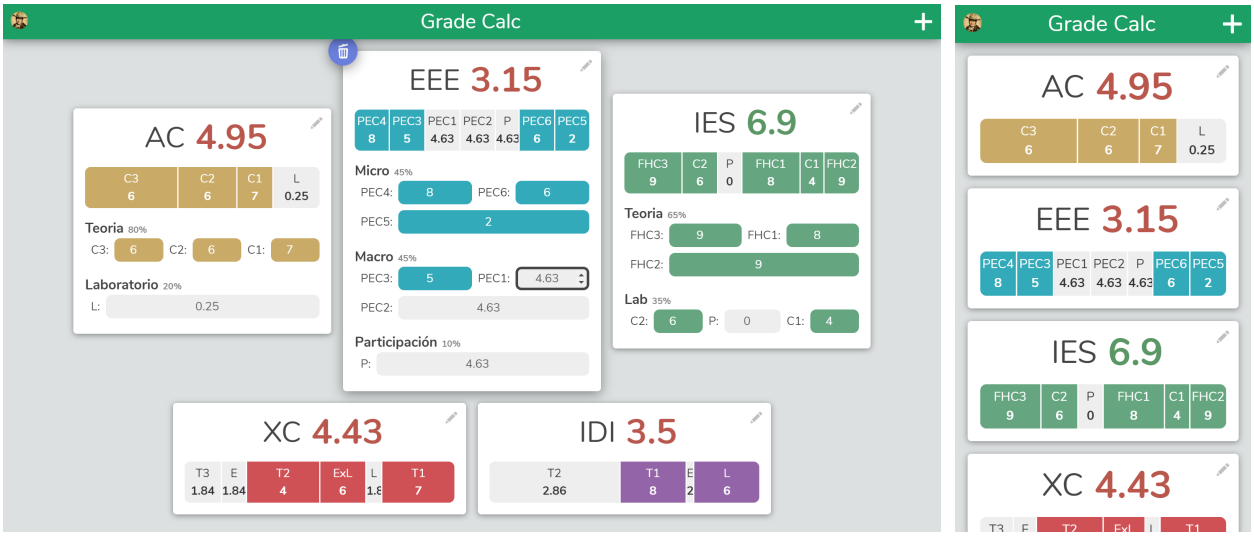
The student is in the 4th semester of Computer Engineering in FIB UPC and he has most of the grades.

In subject *AC* he has done all the exams but he doesn't have the *Laboratorio* grades yet, although he can be confident that he will pass because he needs just a 0.25 to pass.

The same cannot be said for *EEE*, he needs a 4.63 in the two last exams. He has the cursor on the *PEC1* exam and he's typing and erasing numbers to see how the necessary grade changes, he skipped a few classes and he wants to know what grade he would need in the *PEC2* exam if he gets a 3 or 4 in *PEC1*, to see if it's viable.

By looking at the next card, *IES*, we can see that he already passed, with at least a 6.9, regardless of the grade he gets in *P*.

Finally, in *IDI* he has good grades but the exam *T2* weights so much that he still hasn't passed. *T2* takes half of the space in the bar, meaning that his weight is also half of the total, that is to say, 50%.

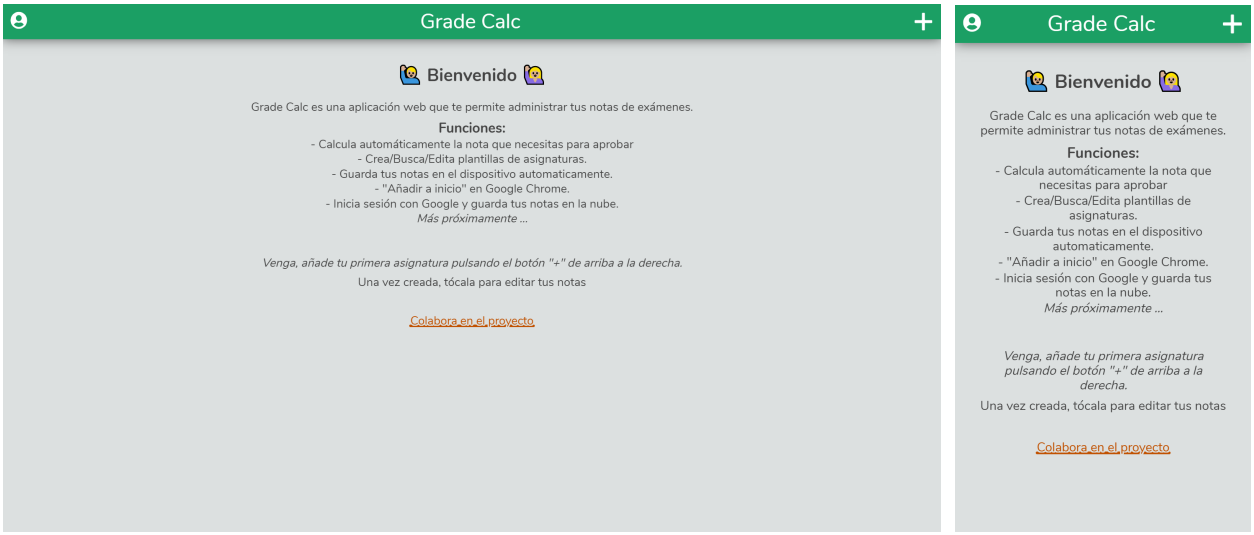


(a) Desktop version **(b)** Mobile version

Figure 4.4 Dashboard

When there are no subject cards in the dashboard, the empty space is used to display a text that explains what is the app and its main features. This text is going to be the first screen of the app that a new user will see, so it aims to help them start using the app. For this reason, the last phrase of the text encourages them to press the + button. To make it more obvious, the + button starts a looping animation (rubberBand from the Animate.css[11] library) that draws attention to it because of its movement.

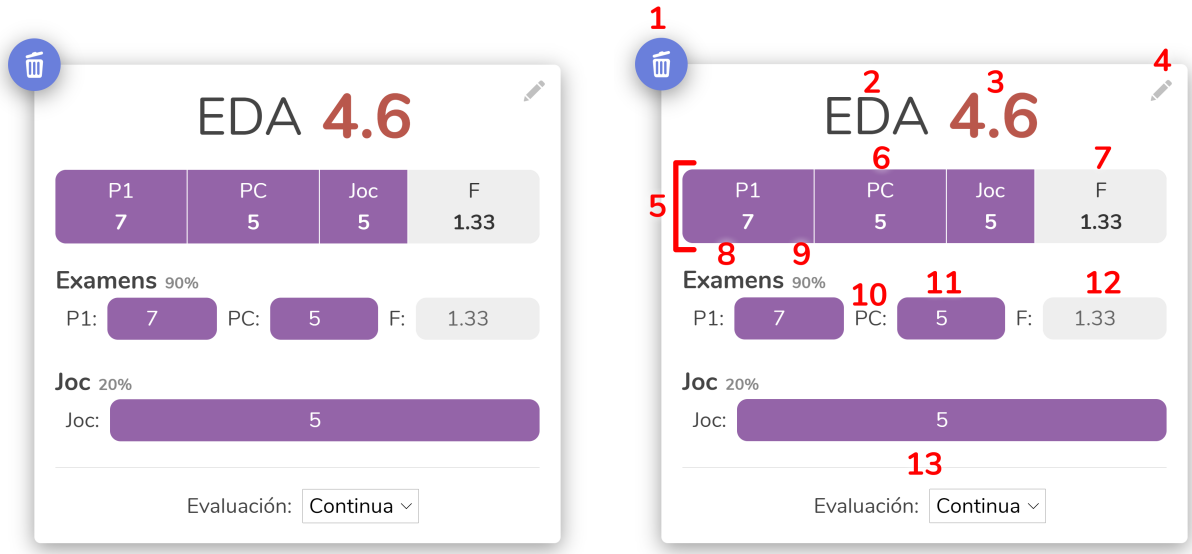
This screen serves another important purpose, it makes the app indexable by search engines like Google. In order to be searchable, the app must have some text to let Google understand what it is. So this text has to contain the keywords that we want to relate to the app.



(a) Desktop version **(b)** Mobile version

Figure 4.5 Welcome

Subject card



(a) Clean **(b)** Marked

Figure 4.6 Subject card

This is a subject card (Fig. 4.6). It's the component that users interact with most of their time in the app, so it is very optimized in terms of usability.

It is important to define what is *necessary grade*. The **necessary grade** is the necessary grade to receive in all the remaining exams to get a 5 in the final grade.

When a card is clicked, it's expanded or collapsed depending on its previous state.

The card's information is updated in real-time as the student changes the grades. This means that the final and necessary grades are calculated every time a value changes automatically. This feature enables users to quickly test several combinations of grades to find the one that will make them pass the subject.

1. **Delete button**: When hovered[1] the button shakes indicating (like it is afraid of being clicked) that indicates that the button is dangerous.

   When clicked the card disappears and another animation occurs, the cards move

[1]Hover means placing the mouse cursor on top of an element without clicking it

smoothly to their new positions. Without the animation, it's very disorienting for the user, it's unclear where each card went, especially when they change row. This is not a trivial animation and I had to code it[26].

2. **Short name**: The short name of the subject, usually it's initial letters.
3. **Current grade**: The current final grade calculated with the filled exam inputs and assuming a 0 in the remaining exams. It's rounded to two decimal places. If it's lower than 5, it's red and if its greater or equal than 5, green. This indicates whether the subject is passed or failed.

   When this value changes from red to green the confetti animation is triggered. Confetti is shooted on the card celebrating that the student passed the subject.
4. **Edit button**: When clicked, the edit screen (Fig. 4.11) of the subject is opened. There the exam weights can be modified, among other things.
5. **Evaluation representation**: It's a bar that contains all exams in the evaluation where the width of every exam is proportional to its weight. This is a simple and comprehensible graphical representation of all the exams' weighs.

   When a section is clicked the respective exam input is focused and the student can start typing the grade in the input field. And when it's hovered a hint below the cursor shows up, indicating the exact weight of the exam.
6. **Exam representation (done)**: The background color is the subject's color and the number is the received grade.
7. **Exam representation (undone)**: The background color is light gray and the number is the necessary grade.
8. **Exam type**: It's a section that contains the exams with that type of selected evaluation. In the example, there are 3 exams with the type *Examens*.
9. **Exam type weight**: This number represents the sum of weights of the exams inside

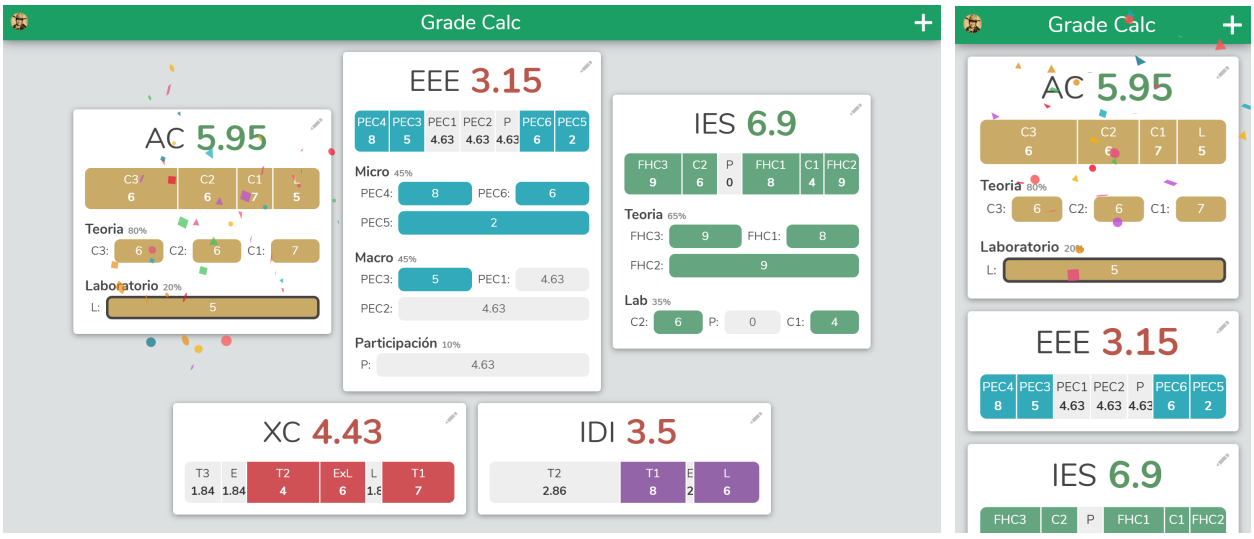
the section. In this case, 90% is the sum of the weights of the exams *P1*, *PC* and *F*. This is helpful when there are many exams inside one category that are weighted very little because the category takes more space it seems more weighted but it is not always the case. In Figure 4.4 you can see its use in the purple subject *F*.

10. **Exam name**: This is the label of the input to the right, it contains the name of the exam. When clicked the input is focused.

11. **Grade input (done)**: he background color is the subject's color and the value is the received grade.

12. **Grade input (undone)**: The background color is light gray and the value is a placeholder with is the necessary grade. When the user types something, the number is overwritten without the need of deleting it.

13. **Selected evaluation**: This is a dropdown with all possible evaluations. If there's only one evaluation it's hidden.



(a) Desktop version **(b)** Mobile version

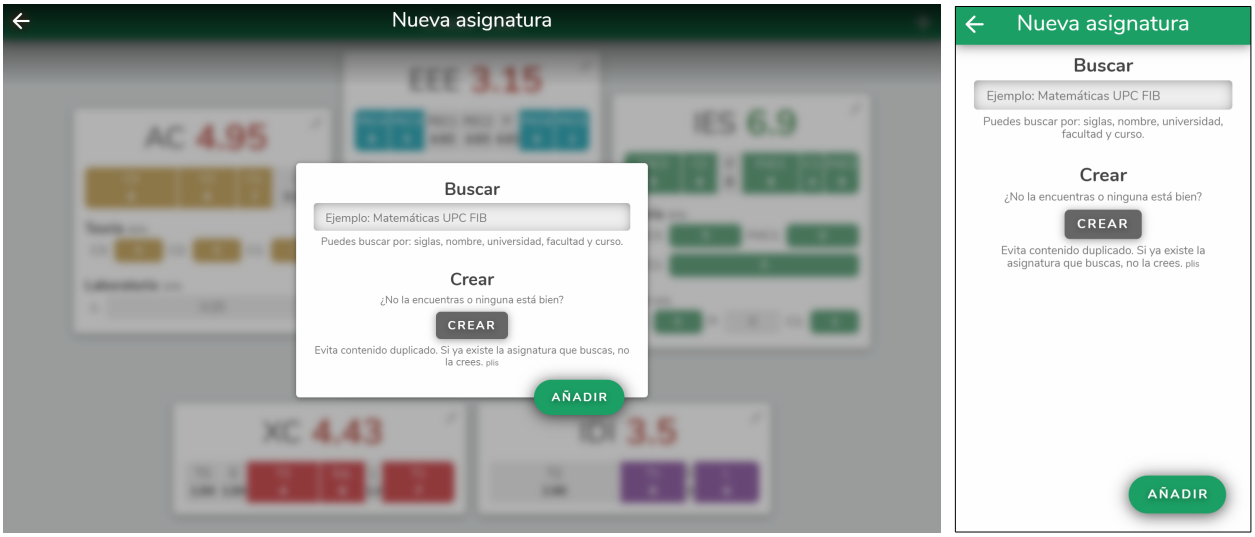
Figure 4.7 Confetti

Search subjects

When the user clicks the + button from the dashboard (Fig. 4.4) this screen appears. In the desktop, it's a popup and in the mobile, it uses the entire screen. Here the user can:

1. Go back by clicking the ← button, clicking the popup background, or clicking the back button from the browser or mobile.
2. Open the create screen (Fig. 4.12) by clicking the CREAR button.
3. Show search results by typing in the search input.
4. Add the selected subjects to the dashboard and go to the dashboard by clicking the AÑADIR button. If there isn't any subject selected it just goes to the dashboard

The CREAR button is in gray to not draw much attention. The app doesn't have a system to manage duplicate information, so we try to guide users into reusing subjects instead of creating new ones. The search field is automatically focused, and in mobile, the keyboard pops up automatically, this makes user start typing instead of clicking the CREAR button.



(a) Desktop version **(b)** Mobile version

Figure 4.8 Search subjects empty input

Notice that when the search input is empty (Fig. 4.8) a there is a placeholder displaying an example, this is a way to teach users what can they search without explaining it.

The instant search engine performs a search each time a key is pressed, showing the 20 most relevant results. The fields relevance depends on what attribute the match occurs, being in this order *shortName* > *longName* > *course* > *faculty* > *university*. The search engine is typo-tolerant in *English*, *Spanish* and *Catalan*, notice the typo in the (Fig. 4.9). The results have the matching parts highlighted to help the user understand why they are there. It also keeps a record of the queries, so I can search what is the subject that more people are looking for and also the queries that don't provide any result.

Having instant search makes the app feel really fast and responsive. And with its configurations, it ensures that the users find what they are looking for on the first query.

Once the search results are on the screen the student can click the empty checkboxes ○ and check them ✓ to add those subjects to the dashboard when the AÑADIR is pressed. If the checkbox is disabled ✓ it means that the subject is already added. When the ✎ button is clicked the edit subject screen is opened (Fig. 4.11) this way the user can check that the evaluation is correct.

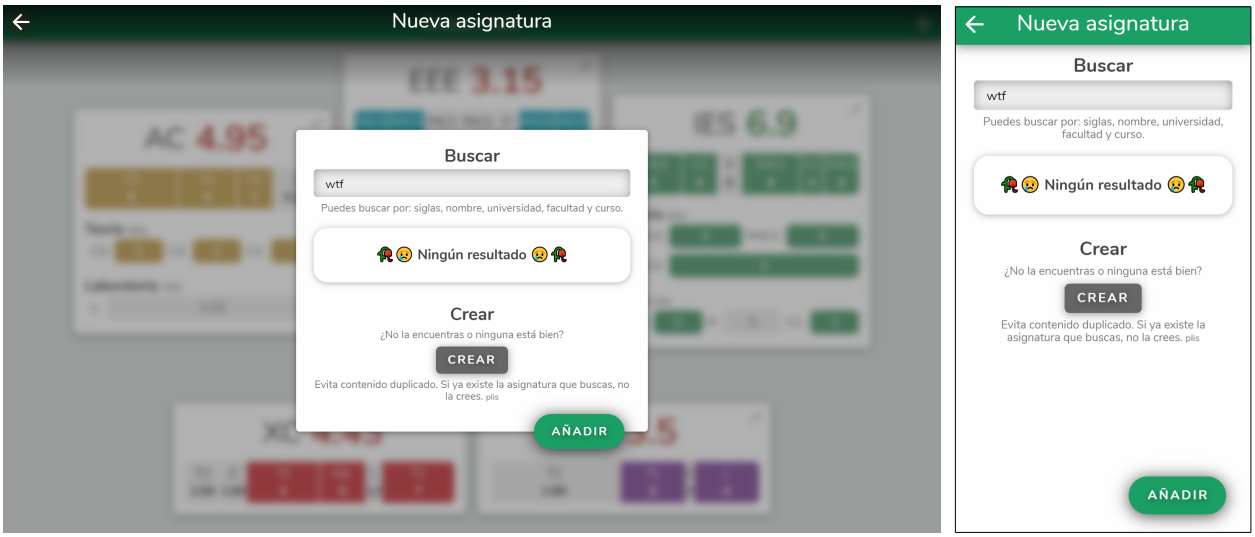
(a) Desktop version **(b)** Mobile version

Figure 4.9 Search subjects input filled

If no results are matching the query a flashy test appears. If nothing is displayed the users think that it's loading or it doesn't work.

The CREAR button hides if there are search results and shows back when there are not. The reasoning behind it is that if the student can't find a subject he'll have to inevitably create it, making it easier for him to find the button.

Algolia's community plan[1], the free one, requires to place their logo close to the search results, so this is why the logo of the search engine is there.



(a) Desktop version **(b)** Mobile version

Figure 4.10 No search results found

Edit and create subject

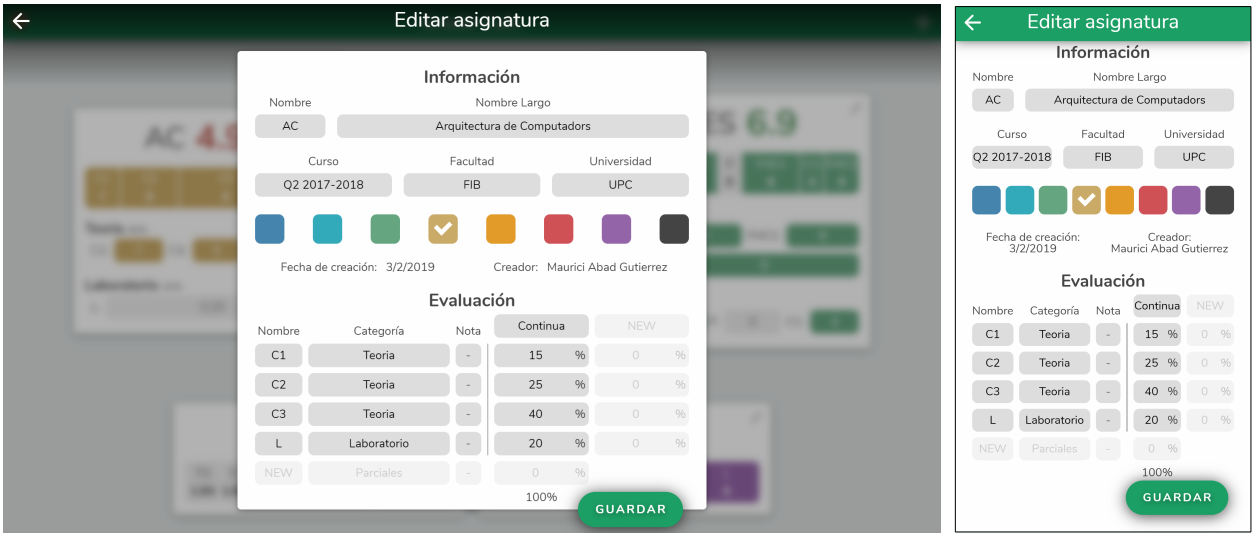
When the ✎ button is clicked, ether from the subject card (Fig. 4.6) or the search result (Fig. 4.6) this screen is opened. The evaluation can be edited using the grid on the bottom.

In the grid, each row is a different exam. And this is what every column represents:

- The first column holds the name of the exam.
- The second column holds the name of the type of the exam, this is only used to better organize exams inside the subject card (Fig. 4.6).
- The third column holds the initial value of the exam. The usual is to leave it empty.
- Every additional column represents a different evaluation. The first cell is the name of the evaluation and the rest cells are the weight of the exam over 100.

Conveniently at the bottom of each evaluation column, there's the sum of all values of the column, to make sure they add up to 100% exactly and avoid mistakes.

To create new rows and columns just type in the faded cells, and to delete a row or column, just empty all of its values and it will disappear.



(a) Desktop version **(b)** Mobile version

Figure 4.11 Edit subject

When creating a new subject by clicking the CREAR button this screen is also open, but its content is empty. As shown in Figure 4.12, there's a validation on the inputs, none of them can be empty, that's why they have a red border. If the user tries to create or edit the subject by pressing the CREAR or GUARDAR button a notification will pop up describing the error (Fig. 4.19). Another validation made is that there must be at least one evaluation and one exam.

Once the user clicks the CREAR button, a blocking spinner (Fig. 4.17) will show because creating the record takes approximately between 0.5s to 2s. Otherwise, users would click the button several times, generating many duplicates. It also lets them know that the app is loading.

(a) Desktop version **(b)** Mobile version

Figure 4.12 Create subject

Log-in and log-out

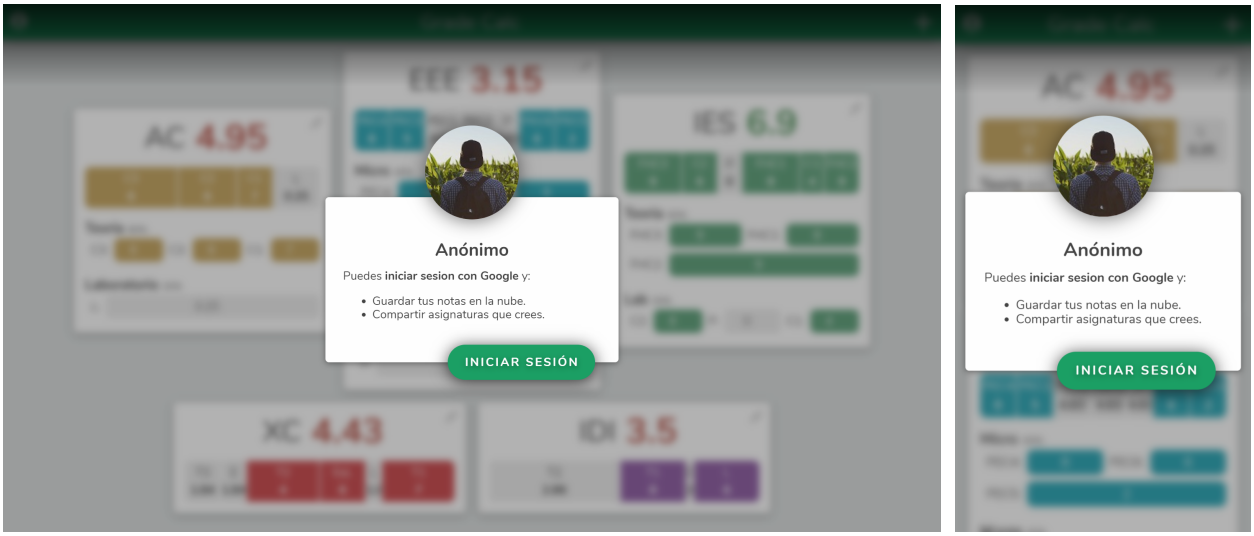
The login screen is the simplest one, it's just a popup in the desktop and mobile, it has a profile picture, name, description of the benefits of logging in and the log-in INICIAR SESIÓN button or log-out CERRAR SESIÓN button. It's opened when the button is clicked.

Then the user is not logged, the profile picture is a placeholder one, and the name is *Anonymous*.

Clicking the INICIAR SESIÓN button goes to Google's log-in page (Fig. 4.14) where the student picks his account and then he's redirected to GradeCalc again.

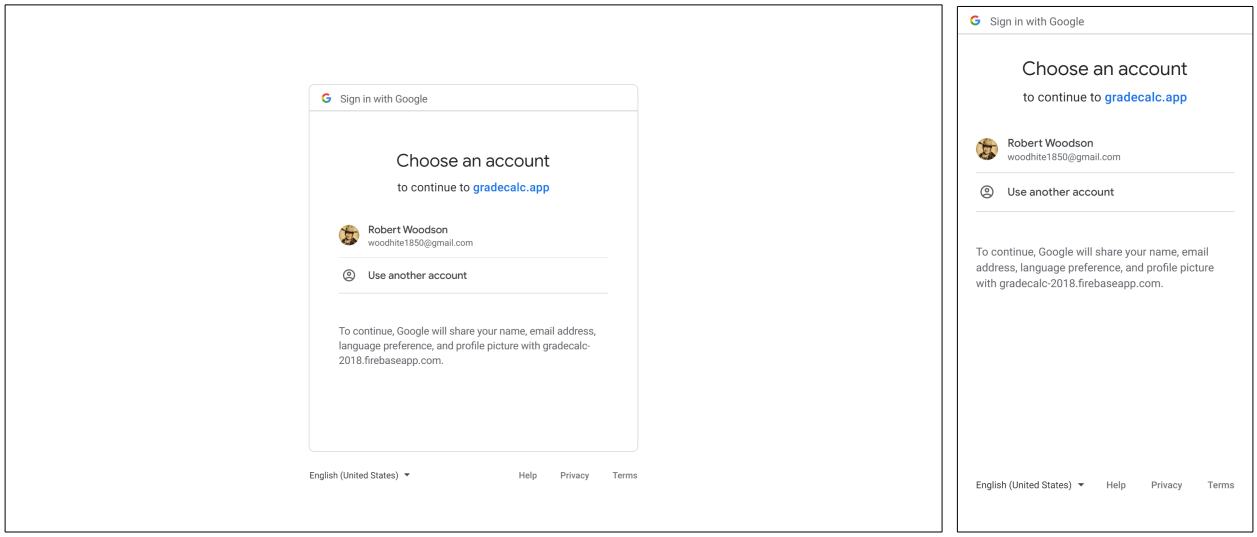
To log-out, the user just has to open the popup and click the CERRAR SESIÓN button.

When the user logins the subjects that he had on the dashboard are synced to the ones in his account. For example, if he added a grade from the mobile and opens the app on the laptop the changes will be downloaded.



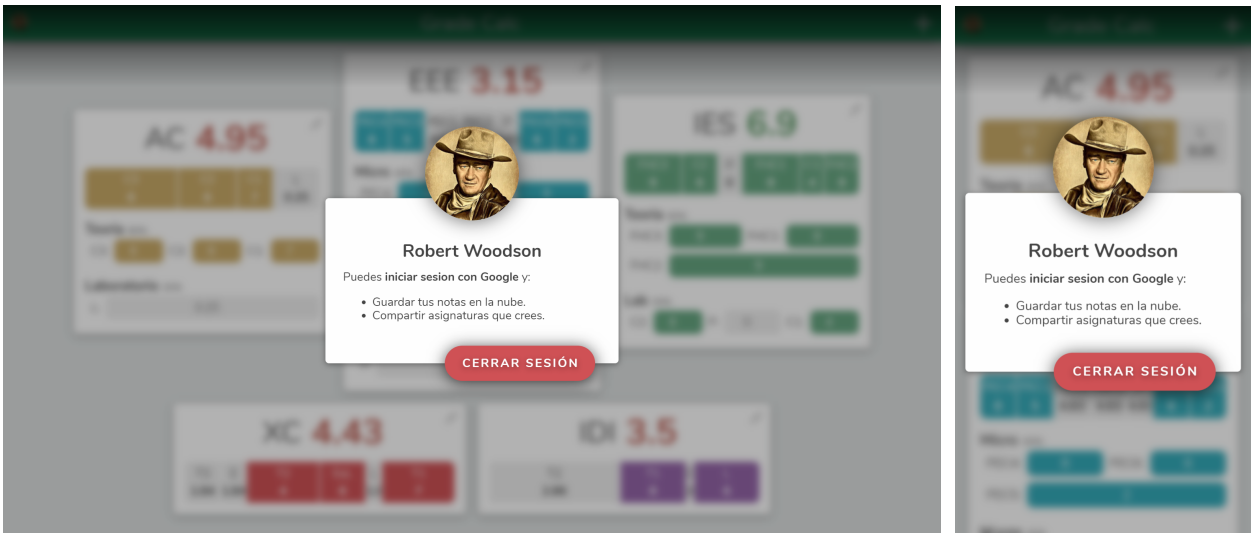
(a) Desktop version **(b)** Mobile version

Figure 4.13 Log-in user is not logged in



(a) Desktop version **(b)** Mobile version

Figure 4.14 Log-in with google



(a) Desktop version **(b)** Mobile version

Figure 4.15 Log-in user is logged in

Background tasks spinner

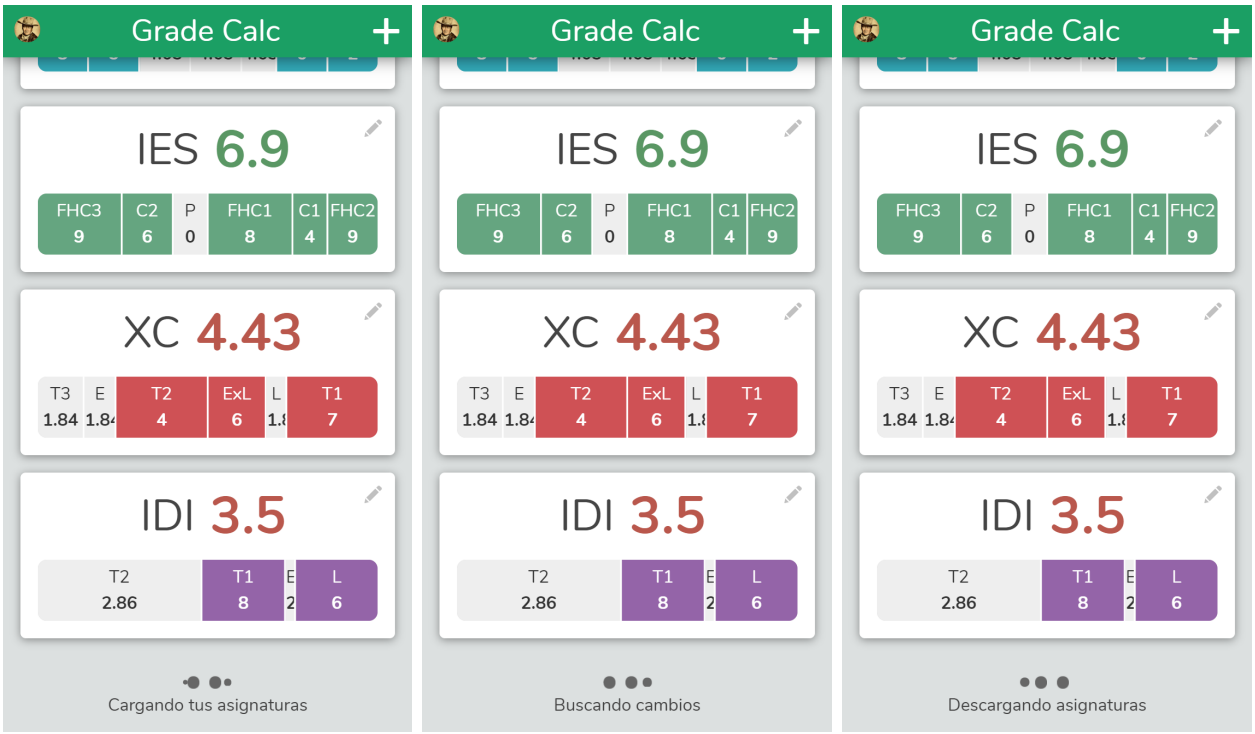
This component is a spinner for background tasks, that informs the user that a background process is happening, but lets him continue using the app. It appears at the bottom of the dashboard and its a bit faded to not be distracting.

In the examples bellow (Fig. 4.16) we can see the three most common messages:

(a) Loading: Getting subjects from the device's local storage.

(b) Searching: Logging-in and connecting to the database.

(c) Downloading: Getting subjects from user's account in the database.

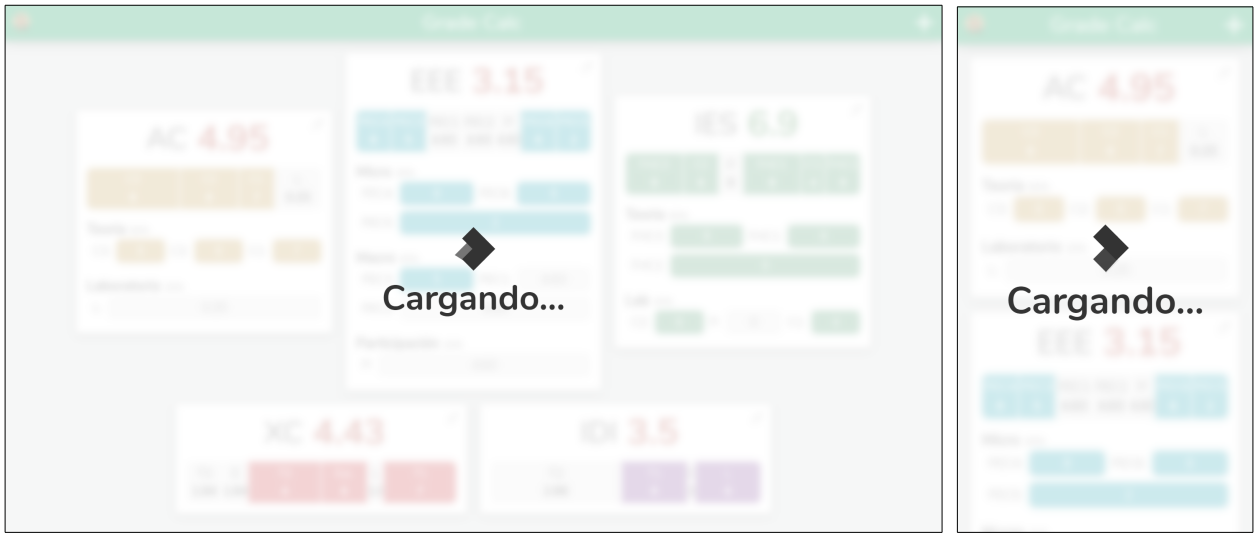


(a) Loading **(b)** Searching **(c)** Downloading

Figure 4.16 Background tasks spinner

Blocking spinner

It's a loading screen that doesn't allow any interaction with the app until it finishes loading. It's annoying, so it's only used when it's unavoidable. If it exceeds the maximum expected loading time, a message appears to let the user know that something went wrong.



(a) Desktop version **(b)** Mobile version

Figure 4.17 Blocking spinner

(a) Desktop version **(b)** Mobile version

Figure 4.18 Blocking spinner taking too much time

Notifications

These are notifications in a snackbar[29] style that appears at the bottom of the screen for a short time, the default is 8 seconds. They consist of a text and an optional action.

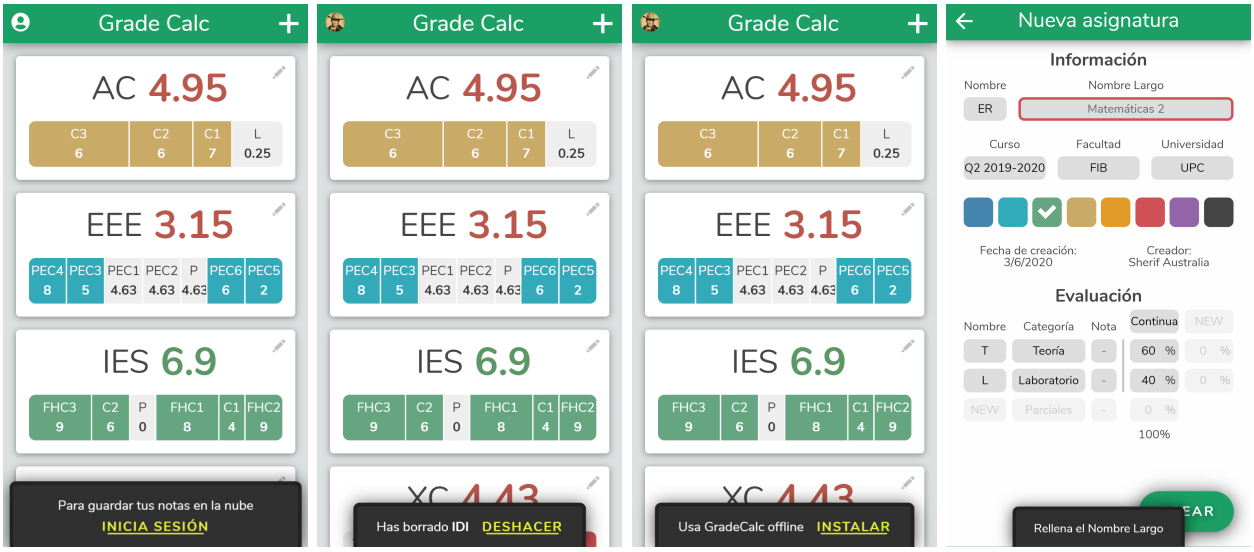
In the examples below (Fig. 4.19) we can see the four most common notifications:

(a) Log-in: It appears when the app is opened and the user is not logged-in. Its action is the same as the INICIAR SESIÓN button.

(b) Undo: It appears when a subject card is deleted. Its action restores the subject. It lets undo a destructive action to prevent unwanted losses of data.

(c) Install: It appears when the app is opened, after the log-in notification, if the app meets the PWA installation criteria[30]. Its action shows the install prompt, more details in section 4.2.1.

(d) Error: It appears when there's an error, for example, an invalid value in the edit subject screen (Fig. 4.11).



(a) Log-in **(b)** Undo **(c)** Install **(d)** Error

Figure 4.19 Notifications

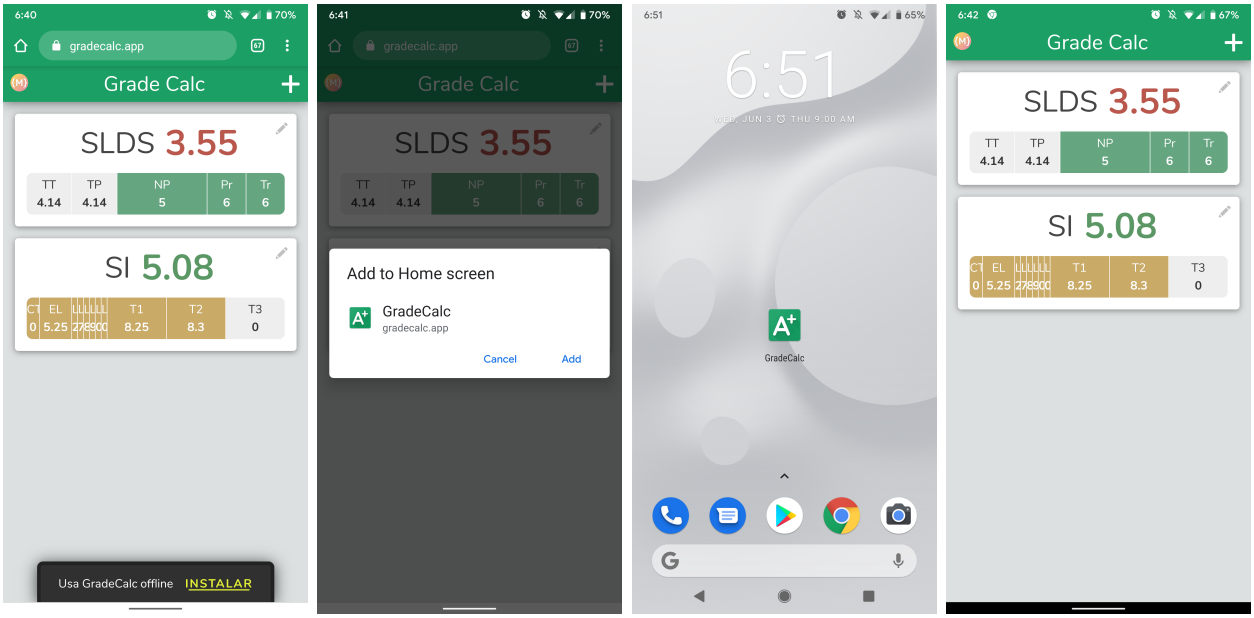
Install prompt

This is the prompt that the operating system shows to confirm the installation of the app (Fig. 4.20b). It shows up in these two scenarios:

- When the user clicks the action of the install notification (Fig. 4.20a).
- When the user clicks the *Install* button from the browser's UI, its location and label may be different across browsers.

Once the app is installed it's icon appears in the device's home screen and the app will open in standalone mode[2]. The icon has 3 versions (Fig. 5.5) to comply to each platform design guidelines.

The app can be installed in: *Android*, *iOS*, *Windows*, *macOS* and *Linux*. The example below (Fig. 4.20) is how to installation process looks in Android 10.



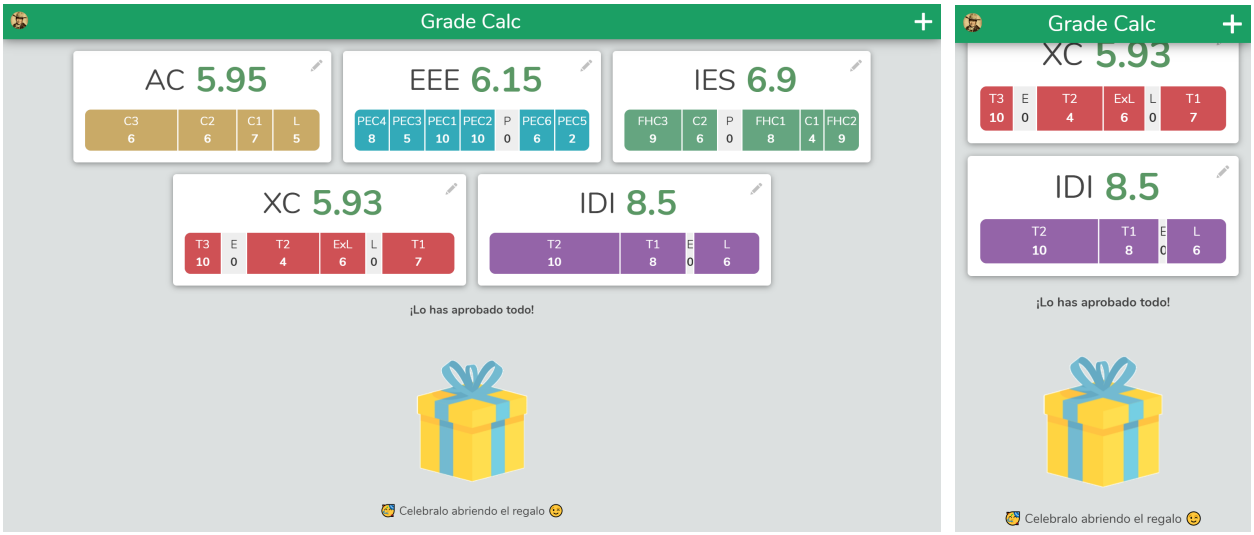
(a) Banner **(b)** Confirmation **(c)** Icon **(d)** Installed app

Figure 4.20 Installation process

[2]The browser UI elements like the address bar and navigation are hidden.

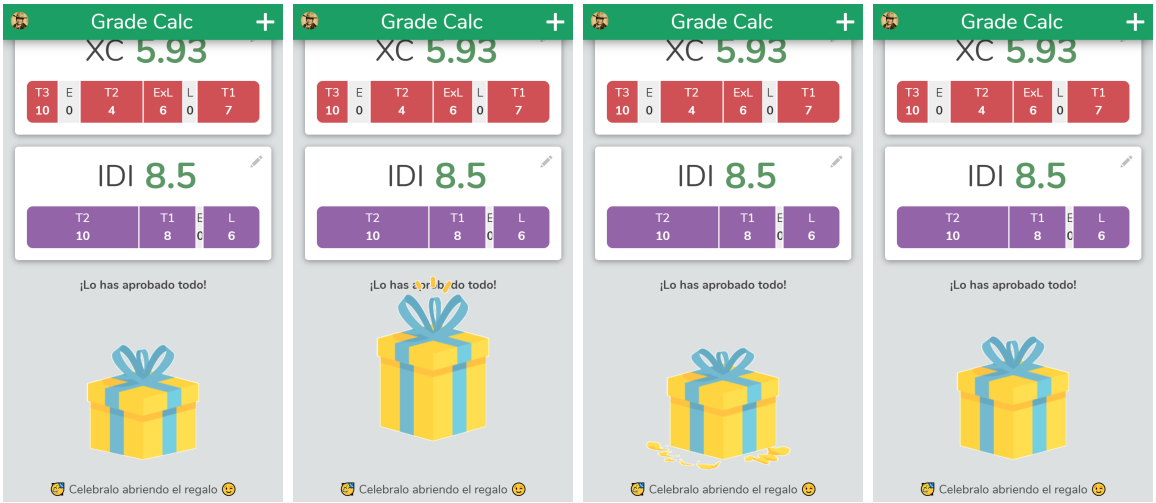
Congratulations

Finally, there's an **easter egg**[3] in the app. To unlock it the student must pass all the subjects in the dashboard. When that happens, a gift appears (Fig. 4.21), with a fancy animation (Fig. 4.22), at the bottom of the dashboard along with a congratulating text.



(a) Desktop version **(b)** Mobile version

Figure 4.21 Surprise gift



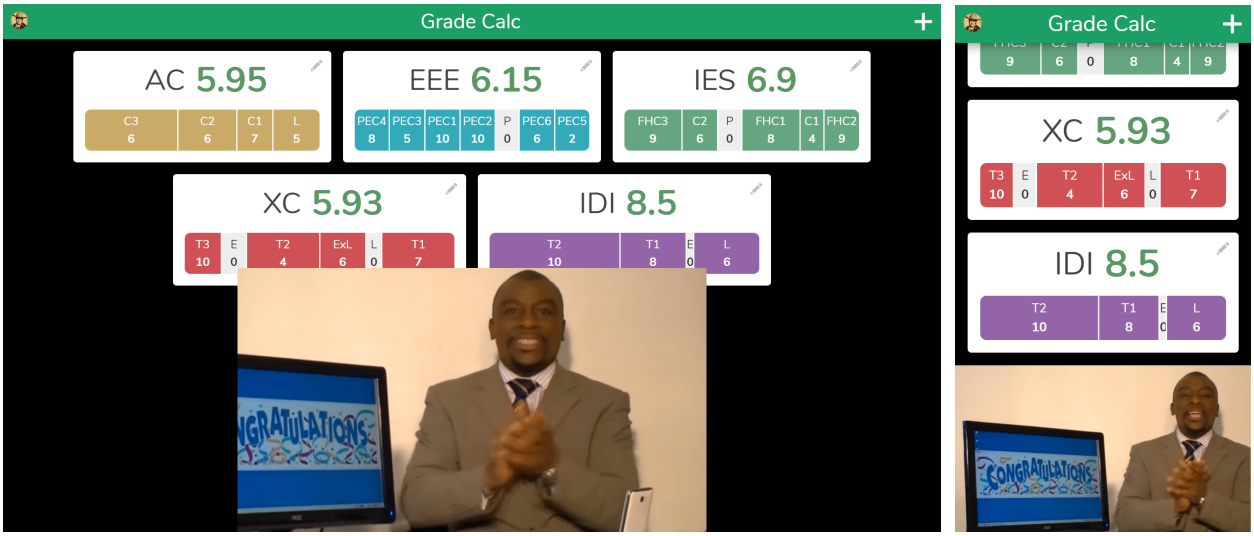
(a) Animation 1 **(b)** Animation 2 **(c)** Animation 3 **(d)** Animation 4

Figure 4.22 Surprise gift animation

[3] An easter egg is a hidden message, or feature in a video game, film, or other, usually electronic, medium.

When the gift is clicked, the background turns black, recalling a cinema switching off the lights, and a video appears at the bottom of the screen (Fig. 4.23). The video plays automatically in a loop and it's changed over time, to keep being engaging. The one right now the featured video is the famous *Congratulations!!!!* meme[27], it's a parody of the EVA[4] ending[2].

Having this feature motivates students to show the app to their friends because they find it funny and interesting. It also engages users to fill their grades to unlock the gift. This is a gamification technique.



(a) Desktop version **(b)** Mobile version

Figure 4.23 Congratulations

[4]Neon Genesis Evangelion, an anime and manga series

4.2.2 Navigational map

This is the sitemap of the app, it shows how it can be navigated. It consists of boxes representing the screens and arrows with icons as labels that represent the button that navigates to the corresponding page. The empty arrows mean that the user can go back, either by pressing the back button of the browser, or the back button of the app.

This map aims to be as simple as possible, to be easier to navigate by the users. To achieve that, the *Dashboard/Welcome* and *Login/Logout* screens are merged into one. Moreover, the *Edit subject*, *Edit & add subject* and *Create subject* screens look practically the same, so when the user learns how to use one, the gained knowledge can be applied to the other screens.

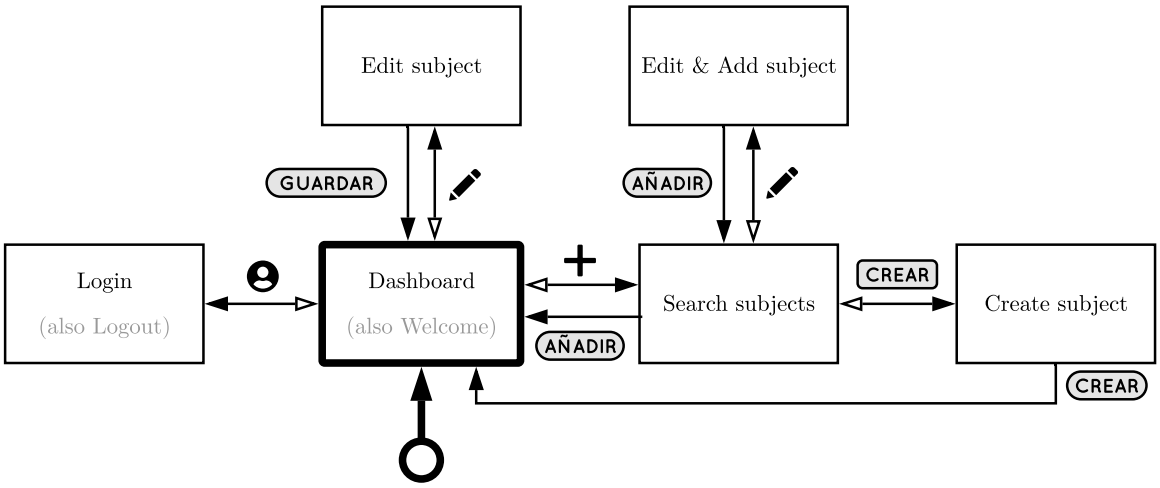


Figure 4.24 Navigation diagram

4.2.3 Flowcharts

The processes of the app are optimized to be notably simple because it has to be easy-to-use. Most of the common actions are 1 to 3 clicks away from the dashboard.

Add subject

This process consists of adding subjects to the dashboard. It is precisely represented in the **Figure 4.25**. Check it out carefully to understand all the paths.

Although it may look complex, usually, it's stunningly simple. With 4 interactions a subject can be added:

1. Click the add button in the dashboard. To open the search screen.
2. Type the name of the subject. The search field is focused automatically, so the user doesn't have to click it.
3. Click the desired search result's checkbox.
4. Click the add button in the search screen.

The process becomes difficult when the subject is not already in the database. When that happens the user has to create the subject. There's also the possibility of editing a subject before adding it, but it's totally optional.

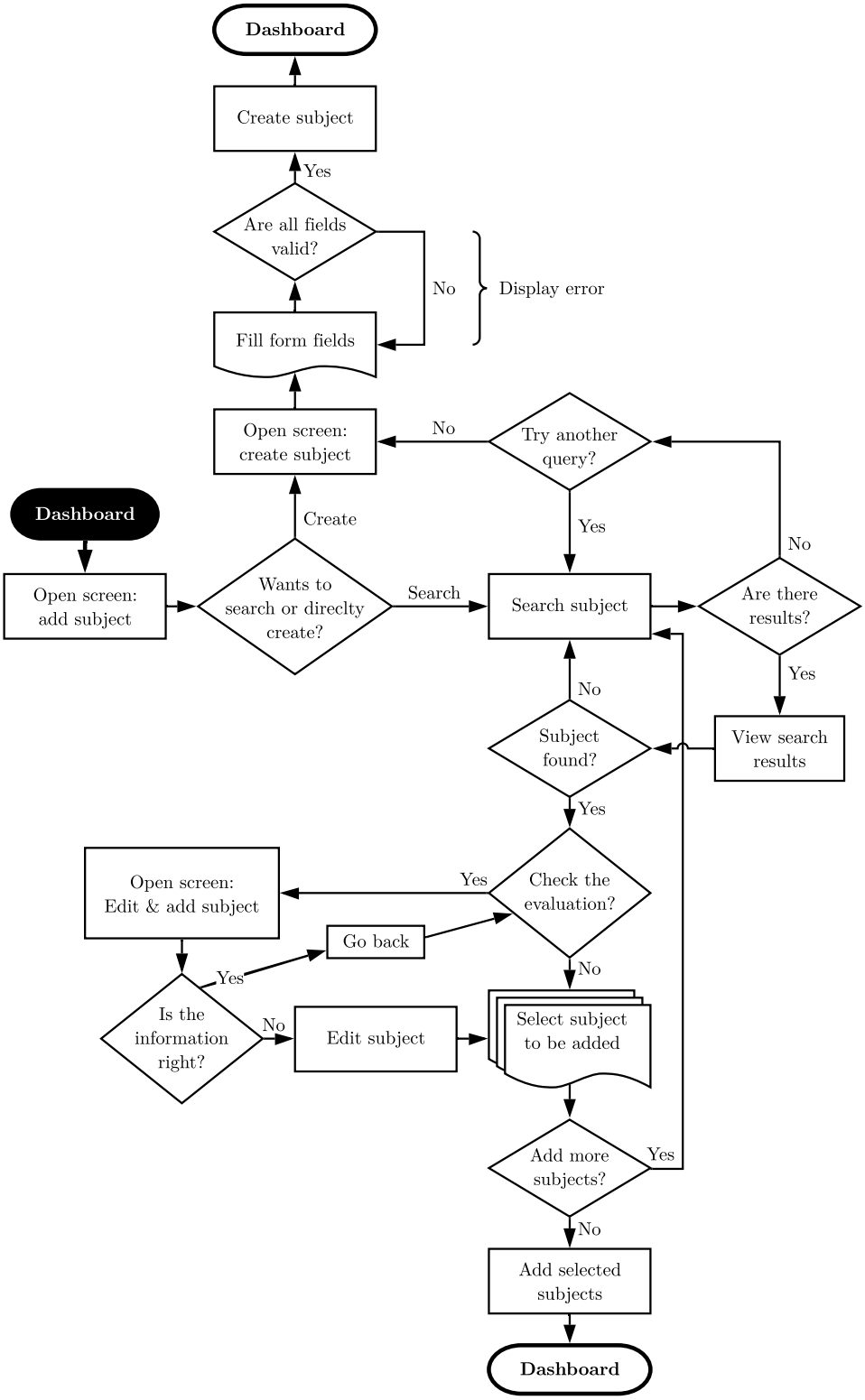


Figure 4.25 Flowchart of adding a subject

More relevant processes

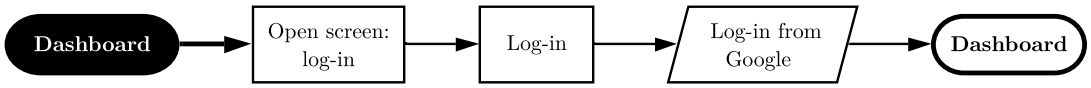


Figure 4.26 Flowchart of logging-in

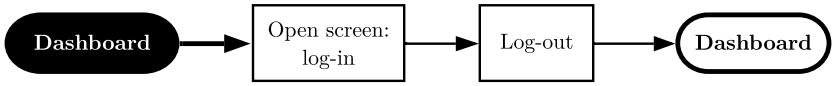


Figure 4.27 Flowchart of logging-out

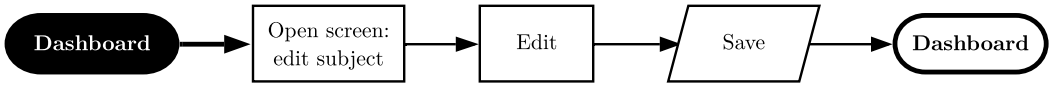


Figure 4.28 Flowchart of editing a subject card

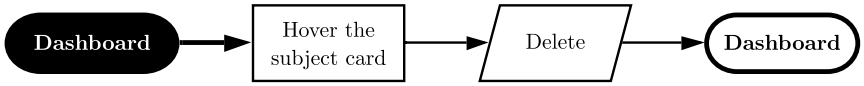


Figure 4.29 Flowchart of deleting a subject card

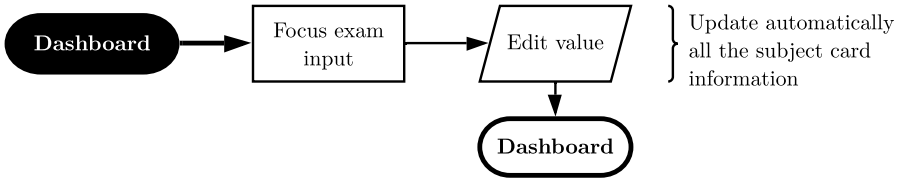


Figure 4.30 Flowchart of editing a grade

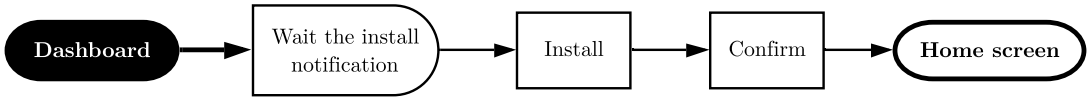


Figure 4.31 Flowchart of the installation

4.3 Domain layer's design

In this project, the domain layer is very thin because the app doesn't have many entities.

The main entity is the subject class. It contains the information about the subject, the user's grades, and functions to make calculations with the grades, like calculating the final grade.

The diagram in figure 4.32 is a summarized version of the domain layer, it doesn't contain all the actual methods.

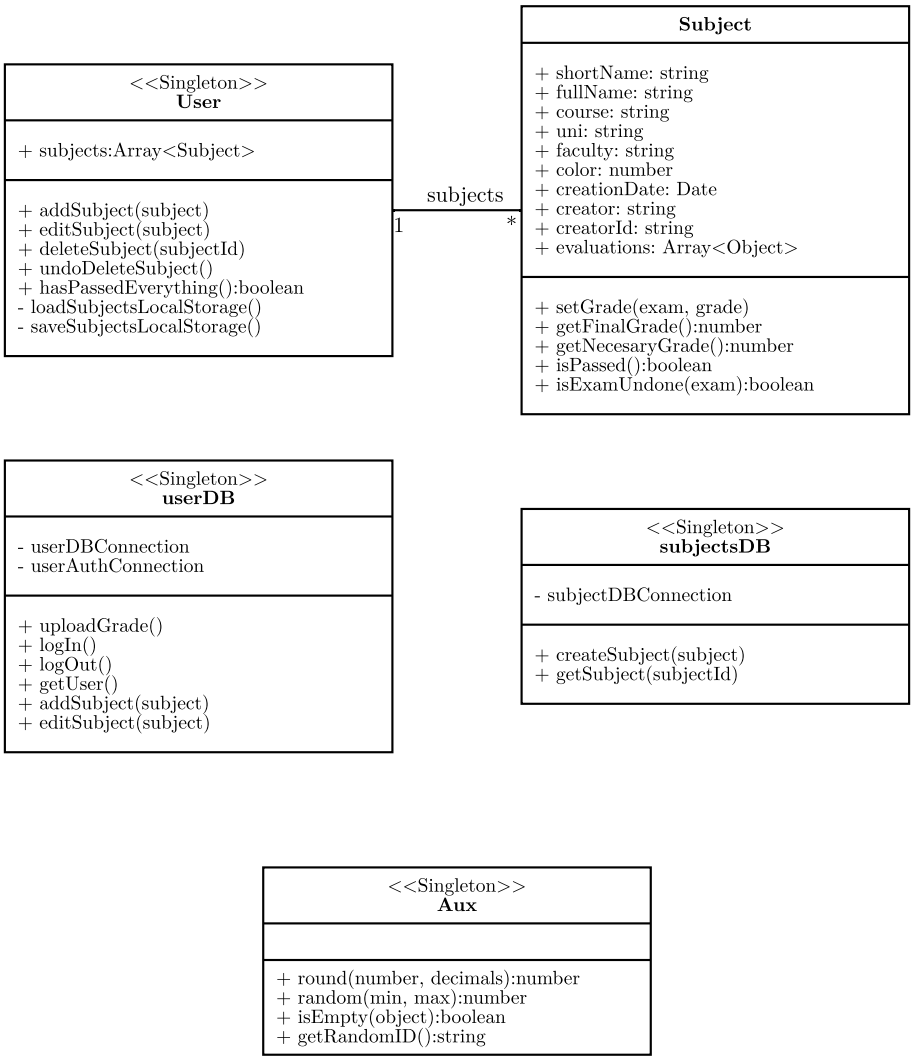


Figure 4.32 Classes diagram

4.4 Data layer's design

This project uses Firebase Cloud Firestore as a database. It is a NoSQL, document-oriented database. Unlike a SQL database, there are no tables or rows, instead, the data is stored in documents, which are organized into collections.

Each document contains a set of key-value pairs. Cloud Firestore is optimized for storing large collections of small documents. All documents must be stored in collections. Documents can contain subcollections and nested objects, both of which can include primitive fields like strings or complex objects like lists. [15]

The point of using Firebase is that it makes reading and writing very easy directly from the client app and you don't have to deal with any server configuration. The standard for a project like this one is to have a server with a SQL database installed, code a middle-ware that exposes the data through a REST API, and access the data from the client. But all these steps require a lot of time. And with Firebase you just access the data from the client. Firebase makes it exaggeratedly fast setup, simple to scale, and easy to maintain.

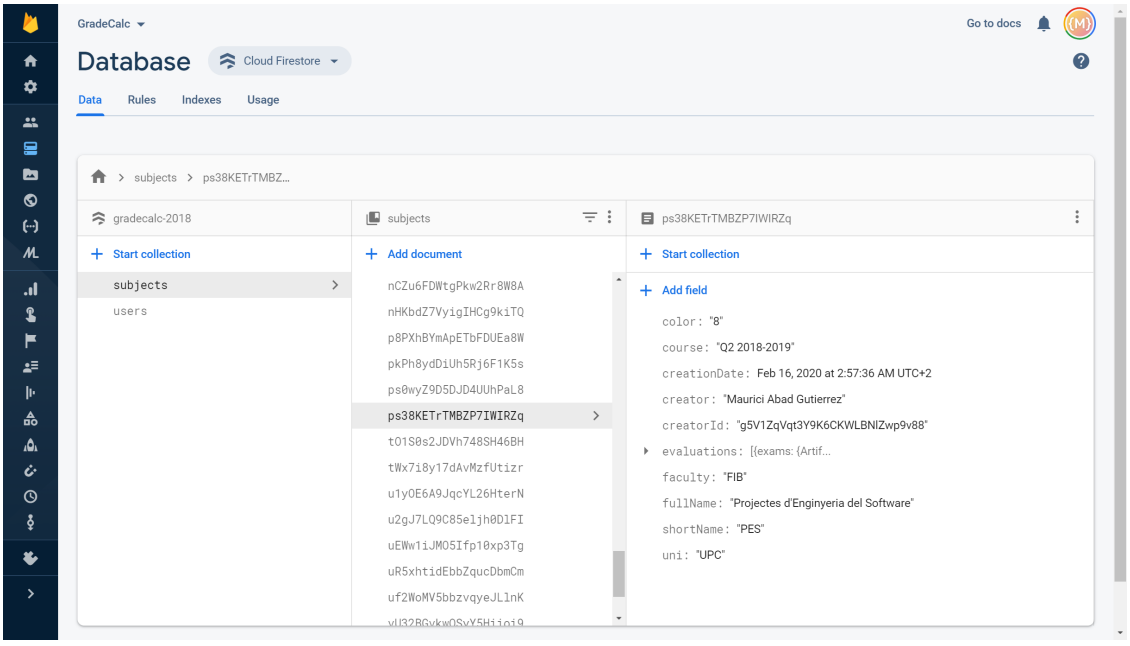


Figure 4.33 Screenshot of Firebase Console

4.4.1 Data model

The firebase database has two collections, users and subjects, that contain `User` and `Subject` objects respectively. The database is defined in TypeScript, so this is its definition in TypeScript.

The attributes `Evaluation.name` and `Exam.name` must be unique in the array. The attribute `Exam.grade` can be undefined when the user hasn't done the exam. And, the attribute `Evaluation.selected` can be undefined when the subject is not assigned to a user.

```
interface User {
  subjects: Array<Subject>
}


interface Subject {
  color: number,
  course: string,
  creationDate: Date,
  creator: string,
  creatorId: string,
  evaluations: Array<Evaluation>,
  faculty: string,
  fullName: string,
  shortName: string,
  uni: string,
}

interface Evaluation {
  name: string,
  selected?: boolean,
  exams: Array<Exam>
}

interface Exam {
  name: string,
  type: string,
  weight: number,
  grade?: number
}
```

For a clearer explanation, here is a UML diagram for the database. Although TypeScript represents it better.

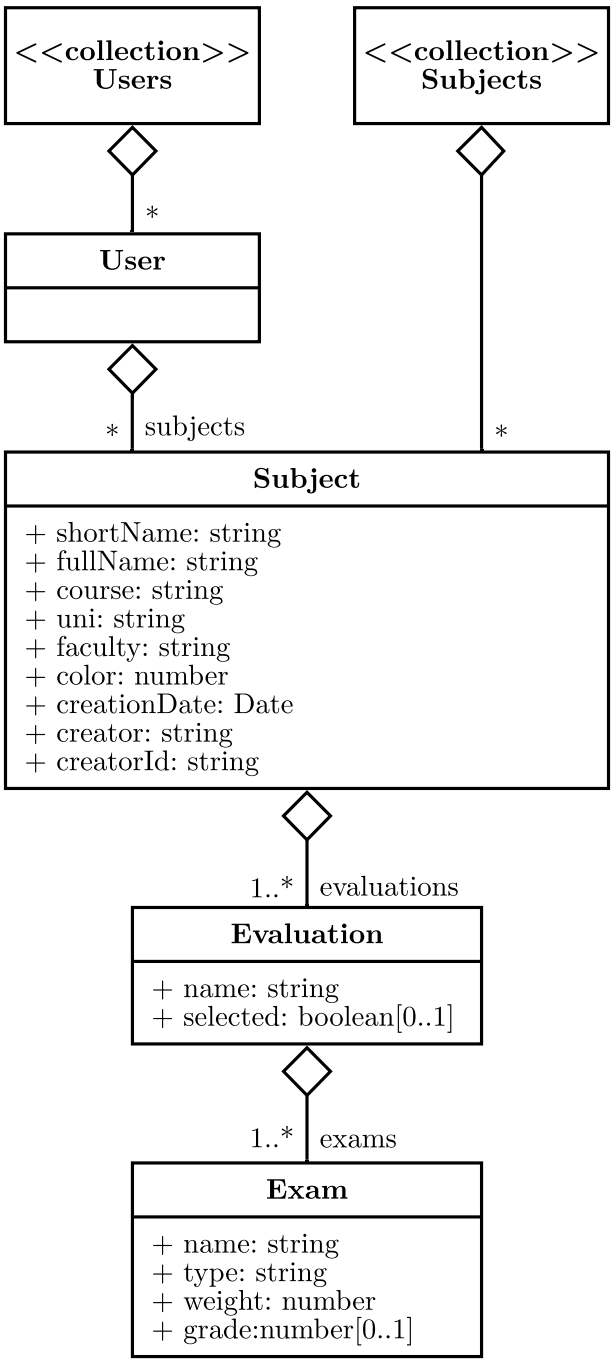


Figure 4.34 Data model's UML Diagram

4.4.2 Example objects

These examples will help in understanding the object's structure.

Example of a subject

This is an example of a subject, in the subjects collection. It has basic information like `shortName`, `fullName` or `course`, and also two evaluations. Each evaluation has exams that have a `name`, `weight` and `type`. Notice that some exams are in both evaluations, this means that they represent the same available item but can be weighted different in each evaluation.

```
{
  shortName: "EDA",
  fullName: "Estructures de Dades i Algorismes",
  course: "Q2 2019-2020",
  uni: "UPC",
  faculty: "FIB",
  color: 2,
  evaluations: [
    {
      name: "Continua",
      exams: [
        { name: "P1",  weight: 0.3, type: "Exàmens" },
        { name: "PC",  weight: 0.3, type: "Exàmens" }
        { name: "F",   weight: 0.3, type: "Exàmens" },
        { name: "Joc", weight: 0.2, type: "Joc" }
      ]
    }, {
      name: "Final",
      exams: [
        { name: "PC",  weight: 0.3, type: "Exàmens" },
        { name: "F",   weight: 0.6, type: "Exàmens" },
        { name: "Joc", weight: 0.2, type: "Joc" }
      ]
    }
  ],
  creationDate: "February 28, 2020 at 9:39:50 PM UTC+1",
  creator: "Maurici Abad Gutierrez",
  creatorId: "4wUPZqVqt1Y9K6CAWLBNlZwe3b12"
}
```

Example of a user

This user has only one subject saved. He changed its color (from color 2 to color 7) and saved some grades (8.33 in P1 and 5.5 in PC). Notice that because he hasn't done the exam "Joc", it's not stored. Also, he has the *Continua* evaluation selected and changed the `color`.

The subject's entire information is duplicated because if the original subject is edited, the data inside the user doesn't change, to prevent unexpected changes.

This object structure is optimized for NoSQL databases because it contains all the information needed to load the screen.

```
{
  subjects: [
    {
      shortName: "EDA",
      fullName: "Estructures de Dades i Algorismes",
      course: "Q2 2019-2020",
      uni: "UPC",
      faculty: "FIB",
      color: 7,
      evaluations: [
        {
          name: "Continua",
          selected: true,
          exams: [
            { name: "P1",  weight: 0.3, type: "Exàmens", grade: 8.33 },
            { name: "PC",  weight: 0.3, type: "Exàmens", grade: 5.5 }
            { name: "F",   weight: 0.3, type: "Exàmens" },
            { name: "Joc", weight: 0.2, type: "Joc" }
          ]
        }, {
          name: "Final",
          selected: false,
          exams: [
            { name: "PC",  weight: 0.3, type: "Exàmens", grade: 5.5 },
            { name: "F",   weight: 0.6, type: "Exàmens" },
            { name: "Joc", weight: 0.2, type: "Joc" }
          ]
        }
      ],
      creationDate: "February 28, 2020 at 9:39:50 PM UTC+1",
      creator: "Maurici Abad Gutierrez",
      creatorId: "4wUPZqVqt1Y9K6CAWLBNlZwe3b12"
    }
  ]
}
```

Chapter Five

IMPLEMENTATION

This chapter explains the technologies and brand chosen.

5.1 Tech stack

A tech stack is the set of technologies an organization uses to build a web or mobile application. It is a combination of programming languages, frameworks, libraries, patterns, servers, UI/UX solutions, software, and tools used by its developers.[39]

This the GradeCalc's Tech stack and what is each technology used for:

Application & Data

- **Firebase**: Backend and non-relational database.
- **Netlify**: Web hosting and Continuous deployment.
- **Algolia**: Full-text search service for Firebase.
- **Heroku**: Run a cron job to send data from Firebase to Algolia.
- **Google fonts**: Serves fonts. Nunito [19] is the one used.

Languages

- **JavaScript**
- **CSS**
- **HTML**
- **JSON**

Business Tools

- **Visual Studio Code**: Code editor with debugging, intelligent code completion, embedded Git and more.
- **Figma**: Modern interface design application.
- **Jira**: Issue tracking applicaion that allows agile project management and more.
- **Google analytics**: Web analytics service that tracks and reports website traffic.
- **Google search console**: SEO application to check indexing status and optimize visibility of their websites
- **Namecheap**: Domain name registrar.
- **Linux**: Operating system to develop.

DevOps

- **GitHub**: Git version management system, store the code and allow many automatons.
- **GitHub bots**: Do some tasks automatically, like update depencencies and compress images.
- **Lighthouse**: Audits the PWA for performance, accessibility, progressive web apps, SEO and more.
- **Code Climate**: Static analysis of the code quality, to avoid code repetition, unused code and more.
- **Travis CI**: Run CI.
- **ESLint**: Static analysis of JavaScript to prevent run-time errors and enforce a standard style and practices.
- **Babel**: Compile modern JavaScript into old JavaScript, this improves browser compatibility.
- **Autoprefixer**: Adds vendor prefixes to unsupported CSS properties.
- **Gulp**: Optimize the files. It compresses images, minifies code, runs babel and autoprefixer and more.

5.1.1 Analysis of alternatives

Most technologies have alternatives so I'll explain why I chose each one.

Application & Data

- **Firebase**: Very generous free plan and offers a great developer experience. It has many tools that simplify a lot of tasks, like login, database, permissions, cloud functions, hosting... I also had prior experience with it, so I didn't have to learn it from scratch.

- **Netlify**: Very generous free plan and offers a great developer experience. Alternatives considered: Firebase, Zeit, and GitHub Pages. GitHub Pages can't be used because the files need to be built and it didn't allow that when the project started. Firebase is not used to avoid relying on it too much and it offers way less storage and bandwidth than Netlify. Zeit is a great alternative they are equivalent, I just personally like more Netlify.

  Another important point for Netlify is that they are really committed to the open-source community, and they offer their Pro plan completely for free to Open Source organizations [31]. By using their service and giving them visibility, more companies will follow their strategies, helping a lot the open-source community.

- **Algolia**: The solution recommended by Firebase itself. They mention that you can also use ElasticSearch but its really expensive and not as easy to setup. [16]

- **Heroku**: The easiest workaround I found to automatically update Algolia.

- **Google fonts**: The most used font hosting by difference, it's the default go-to.

Business Tools

- **Visual Studio Code**: It's becoming the standard for web development, it' brings a really good developer experience.
- **Figma**: It's very similar to its alternatives: Adobe XD and Sketch. Sketch is discarded because it can only be installed in macOS. Adobe XD is equivalent in terms of features, but I like more Figma as a company than Adobe, because they deliver meaningful updates, innovate and don't overprice their products.
- **Jira**: It's an excellent app used by a lot of companies, so learning it will benefit me in future jobs. I also considered some alternatives like Trello, but it's really limited and its UX is poor. The same happens for Taiga although it has more features than Trello.
- **Google search console**: The only available to manage SEO in Google. It has many useful features, like seeing a report by the crawler and getting notifications when the crawler detects errors.
- **Namecheap**: There are lots of cheap domain registrars out there. I chose this one because their support is effective and I like its simple UI. But it's a subjective decision, in this case almost any service works.
- **Linux**: In terms of usability and compatibility it's not as good as macOS or even Windows. But for development all the necessary apps are compatible and the terminal is really useful. It's open-source so promoting it leads to a positive impact on society.

DevOps

- **GitHub**: It's the best platform for hosting Git repositories, it has a great community a tone of integrations with other services. GitLab is also great but it's intended to be a self-hosted solution for git repositories. Then there's bitbucket, but its main target

is companies with private and proprietary code. Most OpenSource projects are on GitHub, so this one is no exception.

- **GitHub bots**: All of them are the only of it's kind. I use: Dependabot to update dependencies. Imgbot to compress images. Stale to close inactive issues.
- **Lighthouse**: It's the best free auditing tool for websites as of 2020. Some other audits are still relevant because they are specialized in certain aspects, like Google PageSpeed Insights that uses Real-World Field Data. But for this project Lighthouse is more than enough.
- **Code Climate**: It provides really good advice and it's free for open-source projects. I didn't choose it for anything in particular, besides that, I already knew it. It's more than enough for this project, so spending time looking for a better option wouldn't affect that much and be a waste of time.
- **Travis CI**: Very generous free plan, offers a great developer experience, and it's the most popular solution. There's also Cirlce CI, which is almost the same, but less popular. I rather TravisCI because it has more community and I can find more code snippets for Travis than any other CI tool.
- **ESLint**: The best JavaScript lantern by far.
- **Babel**: The only and best one.
- **Autoprefixer**: The only and best one.
- **Gulp**: It's super simple and fast to setup. I chose it over Webpack because of its simplicity, although if the project grows more I'll have to migrate to Webpack. I chose Gulp over Grunt because Grunt can only run one task at a time, while Gulp can run multiple ones in parallel.

5.2 Brand identity

Brand identity is the visible elements of a brand, such as color, design, and logo, that identify and distinguish the brand in consumers' minds. Brand identity is distinct from brand image.[28]

In this section, I'm going to explain the process and decisions taken in order to design a solid brand identity.

5.2.1 Color Palette

When choosing a brand's color it's useful to define the concepts we want to relate to the brand. In this case, GradeCalc is related to: *grades*, *exam*, *school*, *math*, *study*... And we want to communicate: *success*, *hope*, *intelligence* and sometimes *failure*.

The most representative colors of grades are green and red because they represent the correctness of an answer. Where green means *right* and red means *wrong*. So, these two colors must be indubitably present in the app.

Figure 5.1 GradeCalc's colors in the color wheel

Because we want to communicate success over failure, green is GradeCalc's primary color. And red is going to be used to convey warning and failure, as usual in interfaces, but due to the nature of this app it's going to be more present than usual, and this is why it's the app's tertiary color.

Red and green are complementary[1] colors, so if we want to add another one to the palette using color theory[5], the only option is to use a tetradic color combination, ether rectangular[2] or square[3]. Using a square tetradic scheme gives us blue or orange (Fig. 5.1), I chose blue because it's less vibrant and can be used easier as a compliment.

Finally, the background is white, and the text dark gray. I chose this combination because it recalls a paper written with a pencil. This is the final color palette (Fig. 5.2):

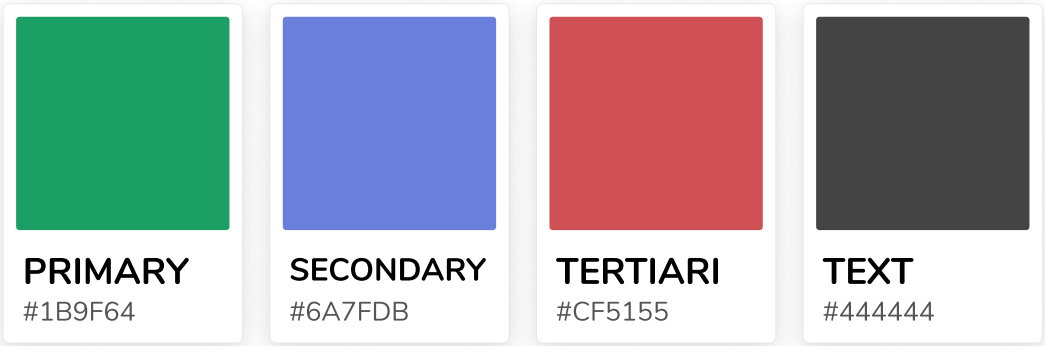


Figure 5.2 GradeCalc's color palette

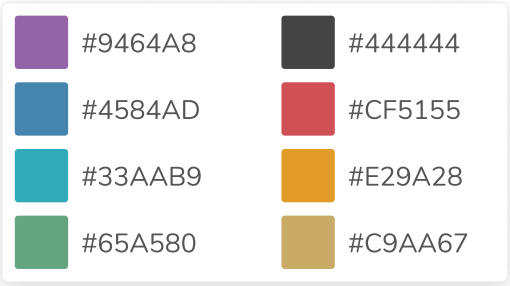


Figure 5.3 GradeCalc's subject's colors

[1]Two colors are **complementary** when they are opposite each other on the color wheel.
[2]A **rectangular tetradic** color scheme has four colors arranged into two complementary pairs.
[3]In a **square tetradic** color scheme all four colors spaced evenly around the color circle.

5.2.2 Typography

Nunito is an open-source typeface created by Vernon Adams. Its a rounded terminal sans-serif font for display but there's also a non-rounded terminal version.

Nunito is a well balanced, highly-readable sans-serif typeface. The characters have thin, uniform stroke widths that work well for both body and display copy. The project began as a rounded terminal sans-serif for display typography, before being extended to a terminal version.[14]

I chose Nunito because it's roundness makes it resembles slightly a handwritten typeface. In exams, the grades are always handwritten, and using this typography makes the grades in the app look like the ones in paper, but with highly-readability.

Another reason for choosing Nunito is that it pairs well with itself, allowing it to be the only font in the app.

Figure 5.4 Nunito font character map[20]

5.2.3 Logo and icons

To design the logo I followed the 5 rules that Saul Edmonds explains in his article *The 5 Rules of Successful Logo Design*[12].

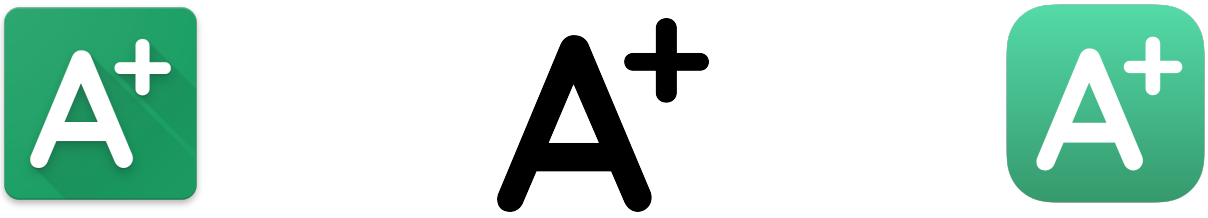
1. **Logo Design Must Reflect Your Business**: The logo is an A+ which is the maximum grade in a letter grading system. Although the target students of the app don't use this system it's so iconic that they relate it more to grades an isolated "A+" than an isolated "10".

2. **Keep It Simple**: Instead of adding decorations I simply used the Nunito font and primary color. This makes the logo less iconic and unique but it makes it more understandable.

3. **Make A Statement In Black & White And Colour**: It works with any color as long as it has enough contrast with the background.

4. **A Scalable Solution**: The logo is a simple character that can be scaled down a lot and still be recognizable.

5. **Keep It Balanced**: To keep it balanced the plus sign is in the top right corner and smaller, otherwise it looked unbalanced.



(a) Android **(b)** Logo **(c)** iOS

Figure 5.5 GradeCalc app icon

5.2.4 Voice

Brand voice refers to the personality and emotion infused into a company's communications.[10]. The first and most important thing to do when defining a brand voice is to understand the clients.

In GradeCalc they are **students, mostly boys from 18 to 25 years old, interested in technology and probably introverts**. This profile is based on all the people I've met in FIB UPC during my bachelor's.

If we consider also other universities in Barcelona, the profile would be: students, from 18 to 25 years old, interested in social media trends and living cool experiences. And this one is based on the people I've met from other faculties.

Notice that this sample is not accurate nor representative, but it's enough to know how they want to be treated. Also, I'm part of the target, so it's very easy for me to understand GradeCalc's users.

So, GradeCalc voice aims to be **friendly**, **practical** and **fresh**. This is how we do it:

- **Friendly**: In the texts, like the welcome screen (Fig. 4.5), we use emojis, colloquialisms, and funny expressions. And the app has nice details that are not annoying and make them smile, like shooting confetti (Fig. 4.7) when a subject is passed and receiving a congratulations gift (Fig. 4.21) when all subjects are passed that plays a funny video.

- **Practical**: Although we are informal we want to go to the point and be an easy-to-use the app. So those funny expressions are always at the end of the texts.

- **Fresh**: We want to provide a sensation of change to better, that GradeCalc is the new way of managing their grades.

Chapter Six

TESTING

This chapter explains the tests done to ensure that the app works. The quality of the app is also measured.

6.1 Manual tests

These tests are end to end tests that test the application's workflow from beginning to end. They aim to replicate real user scenarios so that the system can be validated for integration and data integrity. They have to be followed sequentially.

E2E 1 - Sign-in

Prerequisit: A google account that hasn't been registered yet in GradeCalc.

	Action	**Result**
1.	Click the user icon.	Opens the log-in/log-out screen.
2.	Click the log-in button.	Redirects to Google's log-in.
3.	Log-in with an unregistered account.	Redirects to GradeCalc's dashboard screen. The user is now registered. The user is logged-in into the app.

E2E 2 - Log-in

Prerequisit: A google account that is registered in GradeCalc.

	Action	Result
1.	Click the user icon.	Opens the log-in/log-out screen.
2.	Click the log-in button.	Redirects to Google's log-in.
3.	Log-in with an unregistered account.	Redirects to GradeCalc's dashboard screen. The user is logged-in into the app.
4.	Close and open the app again.	The user is still logged-in into the app.

E2E 3 - Log-out

Prerequisit: The user is logged-in.

	Action	Result
1.	Click the user icon.	Opens the log-in/log-out screen.
2.	Click the log-out button.	The app navigates to the dashboard screen. The user is no longer logged-in into the app. The subjects in the dashboard are the same.

E2E 4 - Subject card

Prerequisit: There is a subject card in the dashboard. This subject has to have 2 evaluations: "Partials": L1(25%) L2(25%) P1(25%) P2(25%), and "Final": L1(25%) L2(25%) F(50%).

	Action	**Result**
1.	Click the collapsed card.	The card expands.
2.	Empty all input fields and select the "Partial" evaluation.	The final grade is 0. The necessary grade is 5.
3.	Input a 10 in L1.	The final grade changes to 2.5. The necessary grade changes to 3.33.
4.	Input a 0 in P1.	The final grade stays 2.5. The necessary grade changes to 5.
5.	Select Final evaluation.	The card now hides P1 and P1. The L1's grade stays to 10. The final grade stays at 2.5. The necessary grade changes to 3.33.
6.	Input an 8 in F.	The final grade changes to 6.5. The necessary grade changes to 0. The subject shoots confetti.
7.	Input a "a" character in L1.	It can't be done, the keypress is ignored.
8.	Input an 11 in L2.	The final grade changes to 9.25. The necessary stays 0. The input is marked with a red border.
9.	Close and open the app.	The information is still the same.

E2E 5 - Search and add subject

Prerequisit: *None.*

	Action	Result
1.	Click the plus icon.	Opens the search screen.
2.	Type "A"	It shows 20 results maximum. The "a"s in the results are emphasized. The create button disappears.
3.	Erase everything	The screen looks like before typing the "a".
4.	Type "AC"	It shows, at least, AC and AC2.
5.	Click the checkbox of AC	The checkbox looks checked.
6.	Click the add button	The app navigates to the dashboard. The AC subject is added to the dashboard.
7.	Close and open the app.	The subject is still in the dashboard.
8.	Click the plus icon.	Opens the search screen.
9.	Type a long string of random letters, something like "asgfchwkaoqxjls"	There is no results and a message shows up along with the create button.
10.	Type "Arquitectura"	It shows, at least, AC and AC2. The AC subject's checkbox is disabled because the subject is already added.
11.	Type "Arquitecture". (it has a typo)	It shows, at least, AC and AC2.
12.	Click the checkbox of AC2.	The checkbox looks checked.
13.	Click the back button.	The app navigates to the dashboard. The AC2 subject is not added to the dashboard.

E2E 6 - Create subject

Prerequisit: *None.*

	Action	Result
1.	Click the plus icon.	Opens the search screen.
2.	Click the create button.	Opens the create screen.
3.	Click the create button	It shows an error and doesn't proceed with the creation. It asks you to fill the information fields.
4.	Fill the information fields	Nothing.
5.	Click the create button	It shows an error and doesn't proceed with the creation. It asks you to fill the evaluation grid.
6.	Type something in any cell of the first row	The row becomes part of the grid and an empty row is added at the bottom.
7.	Type something in any cell of the first column	The column becomes part of the grid and an empty column is added at the right.
8.	Fill more rows and columns.	Nothing.
9.	Empty all fields of a row	The row is deleted.
10.	Empty all fields of a column	The column is deleted.
11.	Input one weight of 101, -1 and 0.00001	The input appears with a red border, but lets you use the values.
12.	Input weights that add to 100%	The percentage sun at the bottom of the column adds to also 100%.
13.	Click the create button	The app navigates to the dashboard. The created subject is added to the dashboard.

E2E 7 - Edit subject

Prerequisit: There is a subject card in the dashboard.

	Action	**Result**
1.	Click the pencil icon in the subject card.	Opens the edit screen.
2.	Append an ”a” to the end of all information fields and change the color.	Nothing.
3.	Click the back button and open the edit screen again.	The subject is not edited.
4.	Append an ”a” to the end of all information fields and change the color.	Nothing.
5.	Add a new exam in the evaluation.	Nothing.
6.	Click the edit button.	The app navigates to the dashboard. The subject is edited.
7.	Close and open the app.	The information is still the same.

E2E 8 - Delete subject

Prerequisit: Have a subject in the dashboard.

	Action	Result
1.	Hover the subject card and click the delete button	The subject card is removed. A notification banner shows, letting the user undo the action.
2.	Click undo in the notification.	The subject is re-added with the same grades and editions it had.
3.	Close and open the app.	The subject is still in the dashboard.
4.	Hover the subject card and click the delete button	The subject card is removed. The notification banner shows again.
5.	Close and open the app.	The subject is not in the dashboard.

E2E 9 - Offline

Prerequisit: There is a subject card in the dashboard.

	Action	Result
1.	Enable airplane mode.	Nothing visually happens.
2.	Modify one grade.	Works as usual.
3.	Close and open the app.	The information is still the same.
4.	Disable airplane mode.	Nothing visually happens.
5.	Close and open the app.	The information is still the same.

6.2 Automated tests

The project runs automatic unitary tests at every commit. The tests run in the front-end's domain layer with the Jest library.

I created this test subject that has several evaluations, each one with particularities, to run tests on them using different combinations of grades.

```
{
  shortName: "TEST",
  fullName: "Testing subject",
  course: "Q2 2019-2020",
  uni: "UPC",
  faculty: "FIB",
  color: 1,
  evaluations: [
    {
      name: "E1 All same weights",
      exams: [
        { name: "A1", weight: 0.25, type: "A" },
        { name: "A2", weight: 0.25, type: "A" },
        { name: "A3", weight: 0.25, type: "A" },
        { name: "A4", weight: 0.25, type: "A" }
      ]
    }, {
      name: "E2 All different weights",
      exams: [
        { name: "B1", weight: 0.2, type: "B" },
        { name: "B2", weight: 0.3, type: "B" },
        { name: "B3", weight: 0.5, type: "B" },
      ]
    }, {
      name: "E3 Some same and different weights",
      exams: [
        { name: "C1", weight: 0.1, type: "C" },
        { name: "C2", weight: 0.1, type: "C" },
        { name: "C3", weight: 0.8, type: "C" }
      ]
    }, {
      name: "E4 Repeated exam names",
      exams: [
        { name: "C1", weight: 0.5, type: "C" },
        { name: "C2", weight: 0.5, type: "C" }
      ]
    }, {
      name: "E5 Weight's sum greater than 1",
      exams: [
        { name: "D1", weight: 1, type: "D" },
        { name: "D2", weight: 1, type: "D" }
      ]
    }
  ],
  creationDate: "April 1, 2020 at 7:12:44 PM UTC+1",
  creator: "Maurici Abad Gutierrez",
  creatorId: "4wUPZqVqt1Y9K6CAWLBN1Zwe3b12"
}
```

Below are the unitary tests (UT) grouped by what they test.

UT 1 - The order doesn't matter

- **Description**: Sets all the grades to 10 in all possible orders.
- **Result**: `necessary`=0 and `final`=$(10 * \text{sum of weights})$
- **Variations**: All evaluations × All exam orders.

UT 2 - Empty grades

- **Description**: Leave all exams undone.
- **Result**: `necessary`=$\frac{5}{\text{sum of weights}}$ and `final`=0
- **Variations**: All evaluations

UT 3 - Normal grades, not all filled

- **Description**: Fill the evaluations with some grades from 0 to 10 with decimals, but leaving at least one empty exam.
- **Result**: `necessary` and `final` depend on the combination.
- **Variations**: All evaluations × 5 grades combinations

UT 4 - Normal grades, all filled

- **Description**: Fill the evaluations with some grades from 0 to 10 with decimals, but filling all the exams.
- **Result**: `necessary` and `final` depend on the combination.
- **Variations**: All evaluations × 5 grades combinations

UT 5 - Necessary is never negative

- **Description**: Set some grades that make `final` ≥ 5

  In E4, set C1 to 10

  In E5, set D1 to 10

  In E2, set B2 and B3 to 10

- **Result**:

  `necessary`=0 and `final`=5

  `necessary`=0 and `final`=10

  `necessary`=0 and `final`=7

- **Variations**: 3 cases.

UT 6 - Necessary can be greater than 10

- **Description**: Set some grades that make `necessary` ≥ 10

  In E1, set A1, A2 and A3 to 0

  In E2, set B2 and B3 to 0

  In E3, set C2 and C3 to 0

  In E3, set C2 and C3 to 4.33

- **Result**:

  `necessary`=20 and `final`=0

  `necessary`=25 and `final`=0

  `necessary`=100 and `final`=0

  `necessary`=11 and `final`=0

- **Variations**: 4 cases.

UT 7 - Negative grades

- **Description**: Set some exams to -10.

  In E1, set A1 to -10

  In E1, set A1 and A2 to -10

  In E2, set B2 and B3 to -10

  In E5, set D1 to -10

  In E2, set B1 to -10

- **Result**:

  `necessary`=10 and `final`=-2.5

  `necessary`=20 and `final`=-5

  `necessary`=65 and `final`=-8

  `necessary`=15 and `final`=-10

  `necessary`=8.75 and `final`=-10

- **Variations**: 5 cases.

UT 8 - Grades greater than 10

- **Description**: Set some exams to -10.

  In E2, set B1 to 11

  In E4, set C1 to 11

  In E4, set C1 to 100

- **Result**:

  `necessary`=3.5 and `final`=2.2

  `necessary`=0 and `final`=5.5

  `necessary`=0 and `final`=50

- **Variations**: 3 cases.

UT 9 - Grades are shared

- **Description**: In E3 set C1 and C2 to 10. And then change the evaluation to E4.
- **Result**:

  In E3: `necessary`=3.75 and `final`=2

  In E4: `necessary`=0 and `final`=10
- **Variations**: Fill E3 and then E4 + Fill E4 and then E3.

6.3 Satisfaction survey

This section showcases and analyses the survey made to validate the non-functional requirements.

6.3.1 Acquisition

To get users to answer the survey I've made a popup that shows when the user opens the app and there is at least one subject card in the dashboard. This way we don't ask new users, that wouldn't know what to answer.

In the popup there's a very short text explaining what is the survey for and the estimated time, to motivate users to take it. Below the test there's the answer button and the URL, so they can share it if they want. When the button or the URL are clicked, the *Don't ask again* checkbox is checked.

If the user clicks the *Later* button, the popup closes and it doesn't open again until the user opens the app and one hour has passed. But if he checks the *Don't ask again* checkbox, the button's label changes to *Done*, and the popup will never show again in that device.

(a) Desktop version **(b)** Mobile version

Figure 6.1 Satisfaction survey popup

The survey is done through Google Forms, and I themed it to match with GradeCalc's aesthetics. It's split up in 4 sections: Introduction, Reliable, Easy to use, and Engaging.

The survey was available for 4 days and 69 users answered it. During this period of time, 321 users opened the app, and 60 of those where new users, so these 60 weren't prompted to take the survey. So 69/261 or 26.4%, of the prompted users answered the survey.

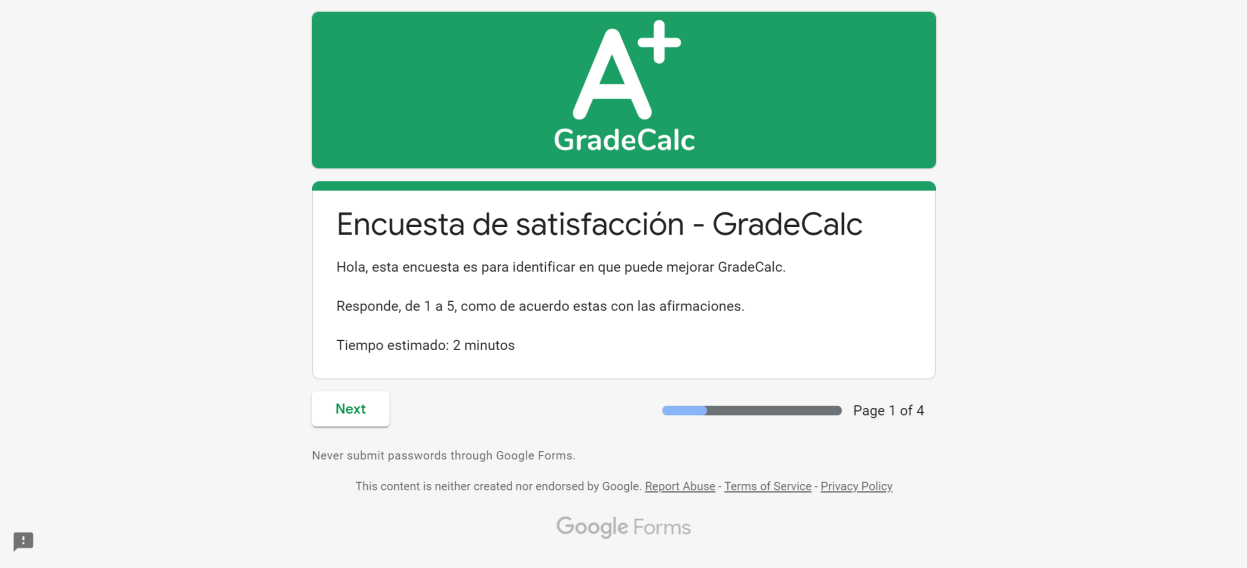


Figure 6.2 Satisfaction survey - Introduction

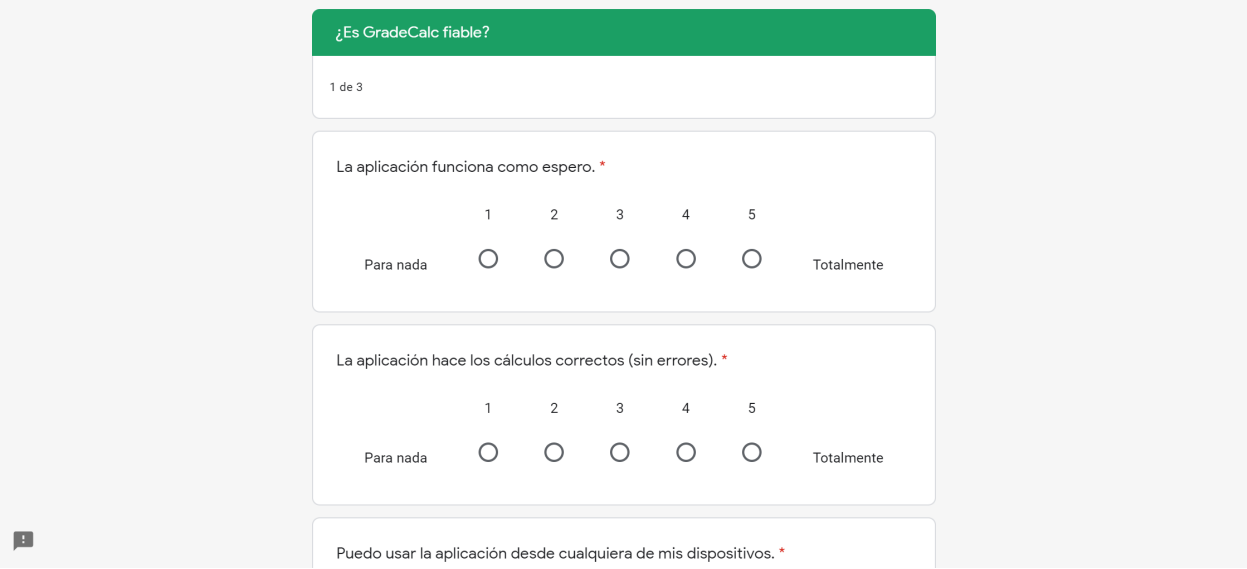


Figure 6.3 Satisfaction survey - Section 1

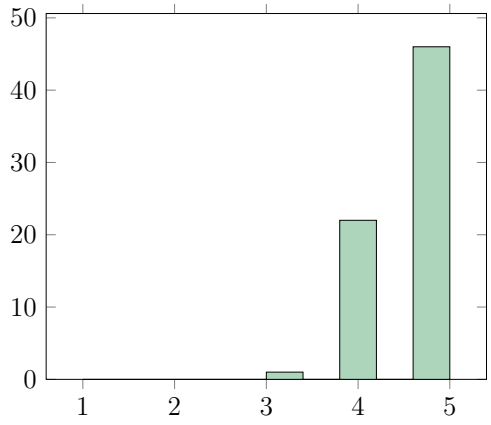
6.3.2 Questions

The questions made in the survey are the ones formulated to validate the non-functional requirements in section 3.2. Users are asked to answer from 1 to 5 how much they agree to the statements. Additionally, there's an extra open question to let users give any feedback they want. These are the statements in Spanish:

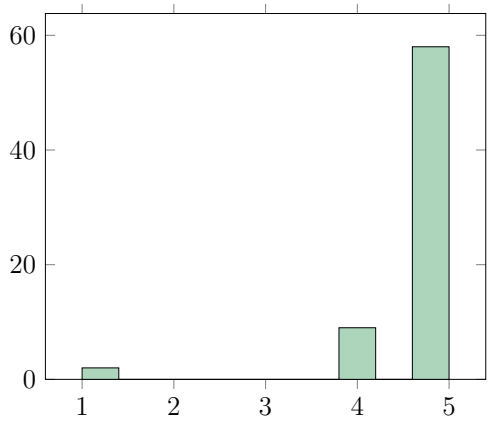
1. La aplicación funciona como espero.
2. La aplicación hace los cálculos correctos (sin errores).
3. Puedo usar la aplicación desde cualquiera de mis dispositivos.
4. La aplicación guarda mis datos. (Nunca ha perdido datos)
5. Puedo usar la aplicación para las asignaturas que curso.
6. La aplicación es fácil de aprender.
7. La aplicación es fácil de usar.
8. La aplicación responde rápidamente a mis acciones.
9. Sé cómo usar la aplicación en general.
10. Sé cómo crear una asignatura.
11. Entiendo toda la información de las tarjetas de las asignaturas.
12. Disfruto usando la aplicación.
13. Recomendaría la aplicación.
14. Hay personas que usan la aplicación gracias a mí.
15. La aplicación me ayuda a lograr mis objetivos. (En relación con la gestión de tus notas)
16. La aplicación me ayuda a aprobar las asignaturas.
17. Abro la aplicación con frecuencia.
18. Abro la aplicación cuando recibo una nueva nota.
19. El logo, nombre y estilo de GradeCalc son fáciles de recordar.
20. Me gustan el logo, nombre y estilo de GradeCalc.
21. Si quieres, puedes dejar un comentario sobre cualquier otro aspecto de la app.

6.3.3 Answers

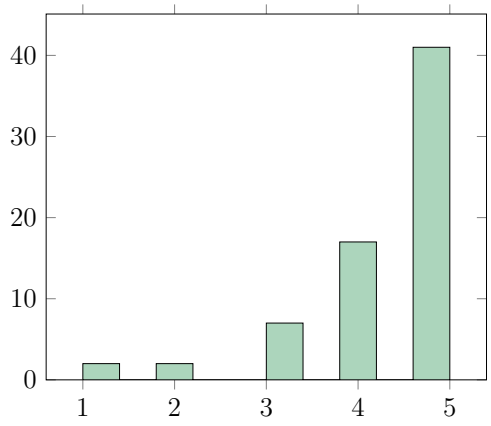
69 users took the survey, these are their answers in the form of plots. All the answers are in the appendix A in the form of a table.



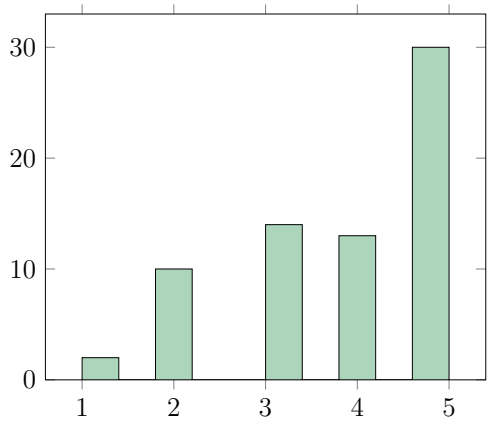
1. La aplicación funciona como espero.



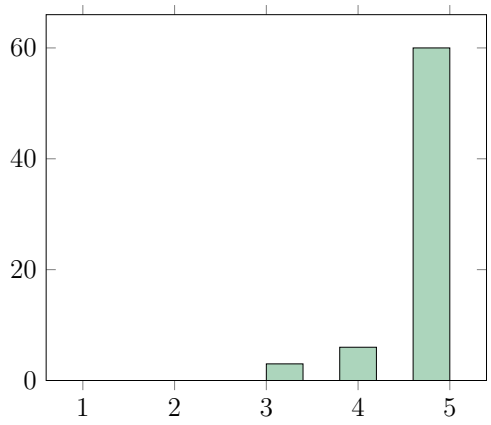
2. La aplicación hace los cálculos correctos (sin errores).



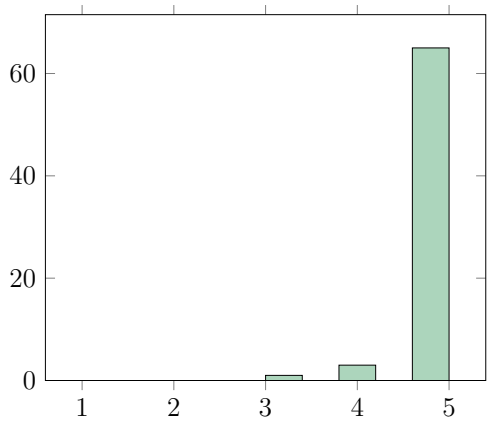
3. Puedo usar la aplicación desde cualquiera de mis dispositivos.



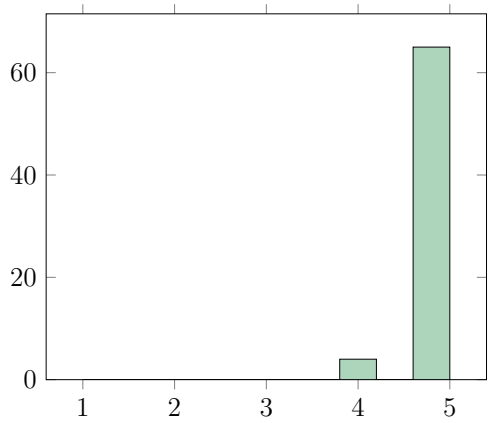
4. La aplicación guarda mis datos. (Nunca ha perdido datos)



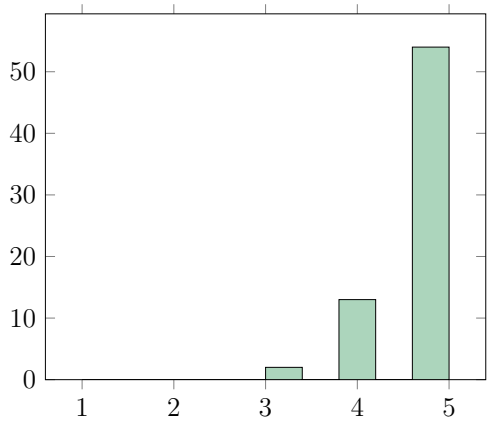
5. Puedo usar la aplicación para las asignaturas que curso.



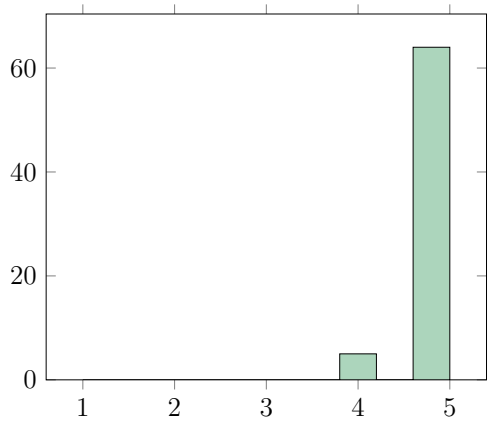
6. La aplicación es fácil de aprender.



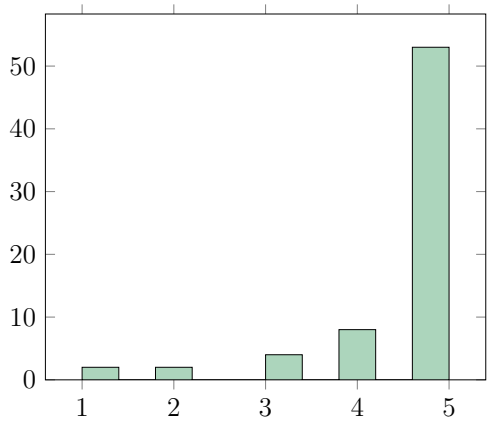
7. La aplicación es fácil de usar.



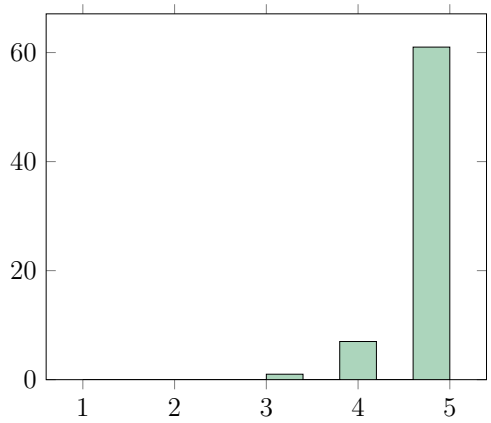
8. La aplicación responde rápidamente a mis acciones.



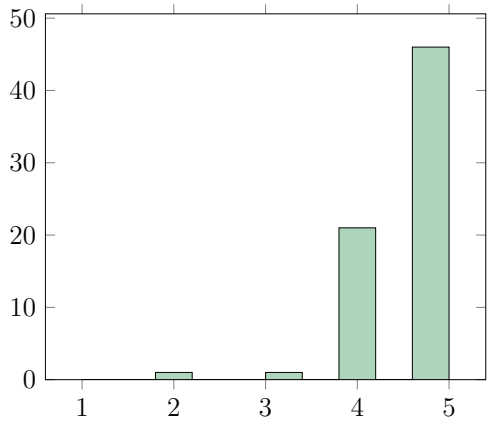
9. Sé cómo usar la aplicación en general.



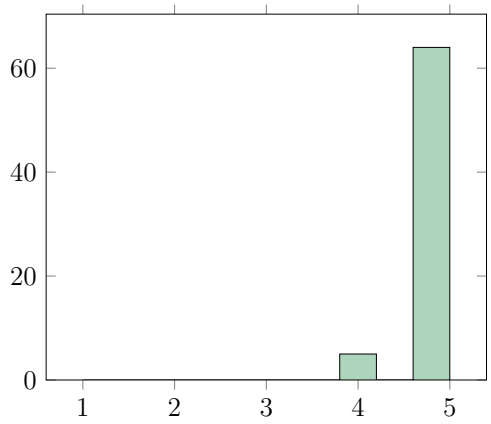
10. Sé cómo crear una asignatura.



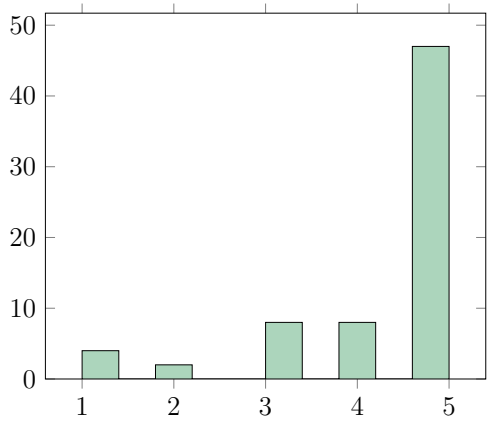
11. Entiendo toda la información de las tarjetas de las asignaturas.



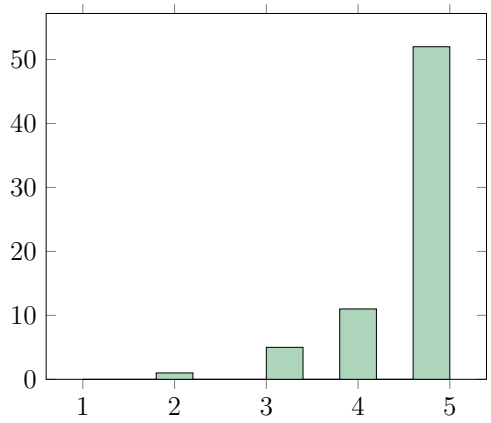
12. Disfruto usando la aplicación.



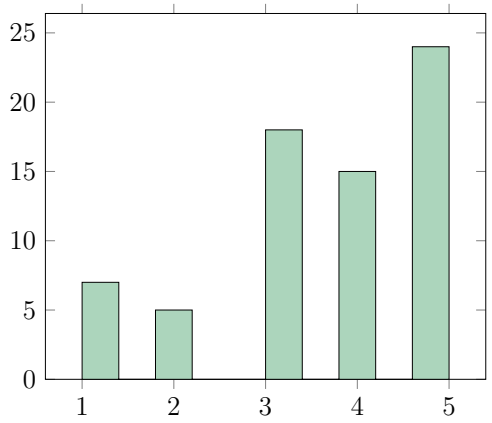
13. Recomendaría la aplicación.



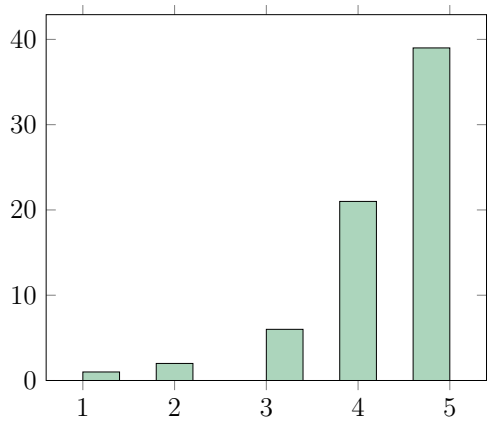
14. Hay personas que usan la aplicación gracias a mí.



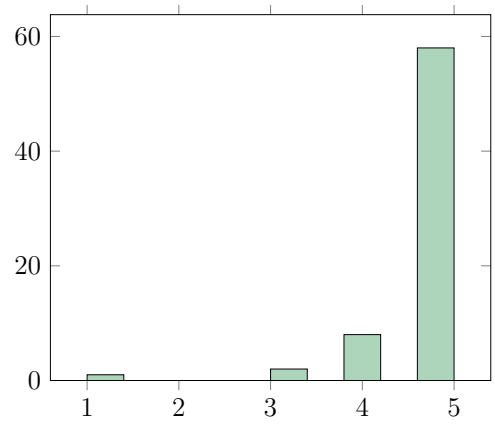
15. La aplicación me ayuda a lograr mis objetivos. (En relación con la gestión de tus notas)



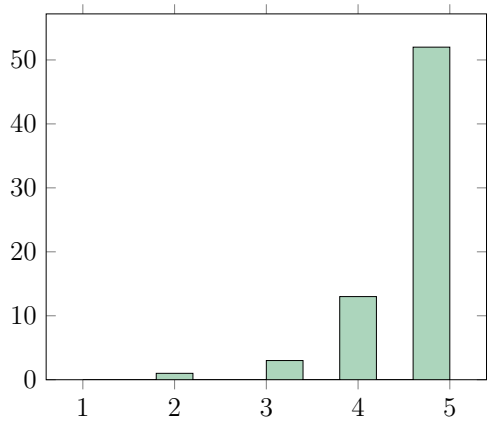
16. La aplicación me ayuda a aprobar las asignaturas.



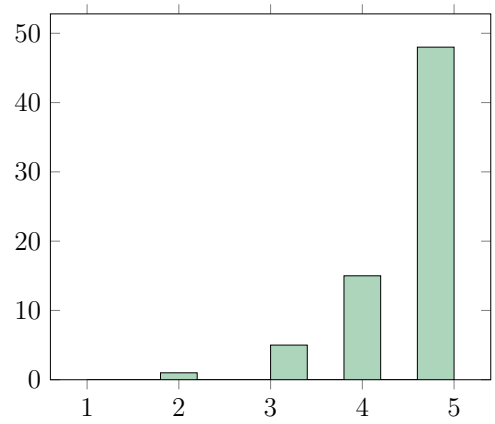
17. Abro la aplicación con frecuencia.



18. Abro la aplicación cuando recibo una nueva nota.



19. El logo, nombre y estilo de GradeCalc son fáciles de recordar.



20. Me gustan el logo, nombre y estilo de GradeCalc.

21. Si quieres, puedes dejar un comentario sobre cualquier otro aspecto de la app. (Errores, sugerencias...)

- Sinceramente, es genial
- A veces me resulta confuso el logo porque se parece al de spreadsheets de google. La aplicación es genial y funciona muy bien. Ojalá poder hacer proyectos así algún día.
- Podrias hacer una app de android/ios
- Sois muy grandes, aplicación súper útil!!!!

- Me gustarian mas colores para las diferentes asignaturas, para que no se repitan
- Crear asignaturas sin cuenta google
- Me gustaría poder reordenar las asignaturas (por ejemplo, arrastrando las tarjetas) y ajustar el tamaño de la cuadrícula en ordenador (poder tener 4 asignaturas en una fila, en una cuadrícula 2x2, etc. Actualmente te obliga a tener filas de 3 asignaturas). También he tenido problemas de pérdida de datos cuando uso la cuenta de google para sincronizar dispositivos. Cuando estén solucionados volveré a usar la app también en el móvil. A parte de esos puntos, es muy buena app y muy útil. No he encontrado ninguna otra que funcione así de bien.
- Hay veces que sin querer pongo una nota que no es razonable, como un 70 o algo así, y me sale como asignatura aprobada hasta que borro la nota. Estaría bien que avisara de que la nota no es válida
- el único error me aparece al usar la misma cuenta en distintos dispositivos. en ese caso, me aparecen asignaturas duplicadas con distintas notas
- Errores al conectarlo con el móvil. Respecto al resto, muy útil.
- En algunos dispositivos se queda atascado al crear una asignatura, y no se puede añadir.
- No soy ningún experto, y la app es la ostia, pero creo que hay lugar para pequeñas mejoras en UX(/UI).
- Hace poco me fallo con el calculo de las assigs pk cambié el % de los exámenes y proyectos y en próximas veces no lo tenía guardado con el valor que puse yo, sino con el original.
- A veces el metodo de avaluacion ha cambiado en la asignatura pero no se ha actualizado en grade calc. Aun así es más que comprensible ya que hay muchas asignaturas y posiblemente muy poca gente gestionando la aplicacion

- la parte en la que la gente no suele estar conforme es porque a veces no está actualizado respecto a los cambios en como se evalua la nota de una asignatura. pero todo es fijarse. Es normal que no se pueda estar en todos lados. Por eso siempre hay que ver como son los porcentajes

- Plantillas de asignaturas ""oficiales"", a veces para la misma asignatura hay 4 opciones con valores distintos

- La aplicación en general está genial y la uso prácticamente a diario o para actualizar cuando hay notas nuevas pero probé a iniciar sesión para que se guardasen mis notas y actualmente la uso sin iniciar sesión porque cuando lo hacía las asignaturas me desaparecían, algunas notas se modificaban o borraban y los campos se ponían aleatoriamente desordenados cada vez que entraba o refrescaba página. Excepto por esos errores la página sin iniciar sesión me ha funcionado siempre perfectamente.

- No he encontrado la app en play store y no puedo tener mis notas en el mobil. Cuando abro la web desde el chrome de mi móbil, no se me sincronizan las notas de mi cuenta. Por lo demás es una app maravillosa!

6.3.4 Conclusions

This is a classification of the questions according to their results.

- **Excellent**: 2, 5, 6, 7, 9, 10, 11, 13, 15, and 18.
- **Good**: 1, 8, 12, 14, 17, 19, and 20.
- **Regular**: 3, and 4.
- **Bad**: 16.

Before proceeding, I have to point out some aspects that may make this survey biased, but are consciously accepted because they still provide relevant feedback.

- The survey was available for 4 days, so the users that answered the survey are likely to be frequent users, thous more satisfied. The results may be more positive because many non-frequent users didn't participate in it.
- The question 4, has the word *never*, and this may make the users take the question more strictly, unlike the other ones.
- Most questions ask for the user's subjective perception that may not match the reality, For example, in question 11, if a user thinks that knows everything shown on the subject card, it doesn't really mean that he knows everything.

Excellent

These are the question where almost all answers are 5 out of 5. From the results, we can conclude that:

- 2. The app doesn't miscalculate.
- 5. The app is suitable for all the user's subjects.
- 6. The app is easy to learn.
- 7. The app is easy to use.

- 9. Users know how to use the app in general.
- 10. Users know how to create subjects. This surprised me, I thought that it would score worse.
- 11. Users understand everything in the dashboard.
- 13. Users would recommend the app.
- 15. The app gives users what they want.
- 18. Users open the app to input their grades when they got new ones, almost always. This means that they have a consistent reason to come back to the app.

These results are very positive. Most of the questions are excellent, meaning that the users are very satisfied with the app.

Good

These are the question where most of the answers are 5 out of 5, but not all. From the results, we can conclude that:

- 1. The app mainly works as expected, but 1 third of the users probably are missing a few functionalities. But none thinks that the app is unpredictable.
- 8. The app is fast, but some users feel that sometimes it could be faster. Everyone agrees the app is not slow.
- 12. Most of the users enjoy using the app, and I guess that the rest like the app, but isn't very into it. This indicates that GradeCalc adds value to users.
- 14. This result is amazing. It means that a large part of GradeCalc's users is promoting the app selflessly. This is the best marketing that the app can have, and also denotes that the users are very satisfied with the app.
- 17. Most of the users open the app frequently, in their opinion. So, they consider that they don't have the app forgotten.
- 19. They consider that the brand is easy to remember, but not everyone agrees.

- 20. Most of the user's like the branding, many don't care, and none dislikes it.

These results are also very positive. We see that many aspects work successfully but there's room for improvement.

Regular

These are the question that had more bad feedback in relative terms, but still, most of the answers were 5 out of 5. From the results, we can conclude that:

- 3. Most of the users can use the app in all their devices, but there's a group that can't. I should investigate what divide is not compatible. But I'd bet that those ones are iPhones and iPads because iOS safari is not optimized for web apps. More research and work needs to be done to improve this score.
- 4. Most of the users never had any data loss, but there's a group that did. Regarding the ones that lost data, I guess from their answers, that what they lost wasn't a big deal, so this issue doesn't look critical right now. Reading the comments from question 21, it appears to be a bug in the synchronization of the grades with the account. More research needs to be done to identify when and what data is lost.

Bad

- 16. This question got a bad scoring, but it's not negative. This question was to know whether users consider that the app helps them to pass subjects or not. Apparently, many people consider that it does, which is great. And many consider that the app has nothing to do with actually passing or not, which also makes sense. I, personally, was expecting that it wouldn't affect.

We can conclude that there isn't any main issue. At least not present in the survey.

The open-ended question

In the open-ended question 21, we can see many positive messages, which is very rewarding as the creator of the app.

Users mainly suggest quality of life features, little changes instead of big changes to the app. I'll note down those suggestions and implement them at some point in the future.

Chapter Seven

DEVOPS

This chapter displays the project's Continuous Integration and Continuous Deployment set up that allows automated builds, tests, releases, and more. Having a correct setup is key to enhance productivity and let developers focus on coding instead of repetitive tasks like deploying.

7.1 Continuous integration

Continuous integration (CI) is the practice of routinely integrating code changes into the main branch of a repository, and testing the changes, as early and often as possible. Ideally, developers will integrate their code daily, if not multiple times a day [3].

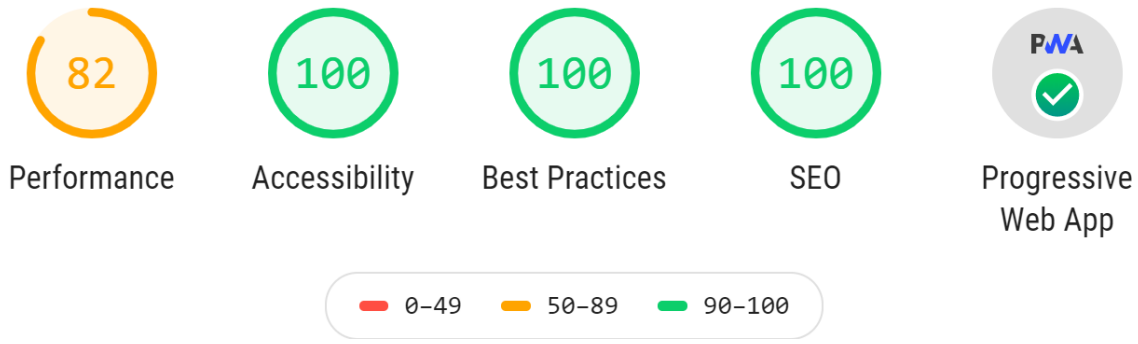


Figure 7.1 Lighthouse score of GradeCalc

Every time a commit happens, the ci pipeline starts. In this project, three processes start in parallel:

1. **Code Climate** runs a static code analysis and generates a score and a report with suggestions.
2. **Netlify** builds the page and generates a deploy preview, more details in section 7.2.
3. **Travis CI** builds the page and runs several processes:
   (a) **Automated test suite**: runs unitary tests.
   (b) **Linterns**: checks that the code style follows a standard and is consistent. The lintern used is ESLint.
   (c) **Lighthouse**: audits the page for performance, accessibility, progressive web apps, and SEO, and generates a report with detailed information about the issues and how to fix them.

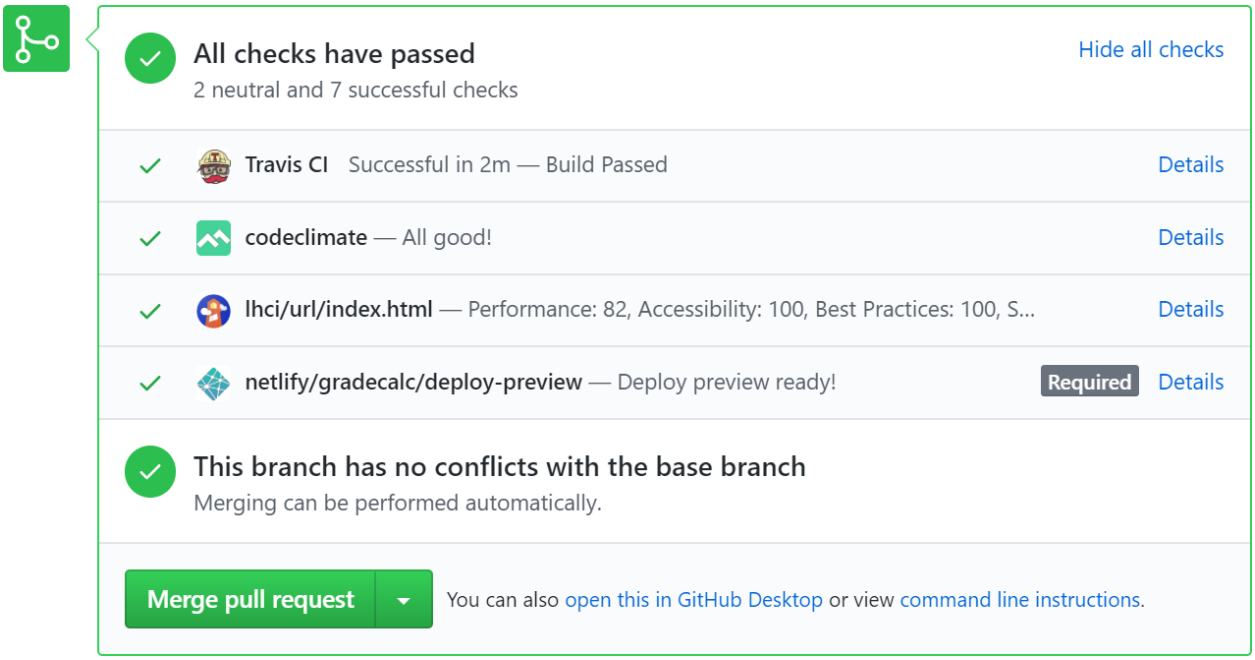


Figure 7.2 Pull request checks of GradeCalc

Other processes are not triggered by commits, but for other events:

1. **Dependabot** retrieves the dependency files, looks for any outdated or insecure dependency, and opens pull requests to update each requirement individually.
2. **Stale** automatically close stale Issues and Pull Requests that tend to accumulate during the project.
3. **Imgbot** watches for new images in the repository and opens pull requests optimizing those.

7.2 Continuous deployment

Continuous deployment (CD) is an approach in which software functionalities are delivered frequently through automated deployments. In contrast to occasionally releasing many changes at once.

The benefits that CD provides are many, the ones that favor this project are:

- Developers don't have to worry about deploys and they can focus on code. Deploys happen seamlessly without problems.
- Removes any human errors during the deployment process.

This project's CD setup is entirely handled by Netlify, they describe themselves as follows:

> Netlify is an all-in-one platform for automating modern web projects. Replace your hosting infrastructure, continuous integration, and deployment pipeline with a single workflow. Integrate dynamic functionality like serverless functions, user authentication, and form handling as your projects grow. [32]

This is how Netlify is used in this project:

In every commit, Netlify builds the page. In each build, a deploy preview is generated. This preview used for testing in a pre-production environment, and once everything works fine the changes can be merged into the master branch.

Once a commit happens in master, either from a merge or a direct commit, Netlify builds the page, and if it builds successfully it is deployed into Netlify's powerful hosting infrastructure, in our case this is production.

To build the page Netlify uses the npm command `npm run dist`. That command creates a `dist/` folder that contains all the files that need to be hosted. These files are optimized versions of the source code.

Chapter Eight

POST-DEVELOPMENT

Once the application is deployed, certain tasks have to be done periodically to keep the application running successfully and to analyze its performance. In this chapter, I'm going to explain those tasks.

8.1 Maintenance tasks

8.1.1 Fix critical bugs

Sometimes, often due to insufficient testing, critical bugs appear and they make a part of the app unusable. These bugs have to be solved as fast as possible.

One example of a critical bug that happened is that the domain expired, without me noticing, and the app could only be used by users that already opened it. It worked for them because the app can work offline. GradeCalc downloads all the necessary files in the first load. So I didn't notice the error until a real user reported it.

8.1.2 Git repository administration

The git repository needs to be administered in order to open/comment/close issues and pull requests. Because I use Git Flow (Chap. 2.2) and Continuous Integration (Sec. 7.1), I have to constantly check the status of pull requests.

8.1.3 Tidy up subjects in the database

Every time a logged-in user creates a subject it's uploaded to the database. But GradeCalc doesn't have an automatic system to remove duplicate subjects because it's very complex to know what is the right and newest information, so deleting duplicates has to be done manually.

Also, the format of certain information can't be checked automatically. For example, the *course* attribute, usually looks like "Q2 2019-2020" but in other universities they may have a different notation, so the users can't be forced to use this format.

To perform these tasks I access the Firebase console (Fig. 4.33) do 2 checks:

1. Sort the subjects by `creationDate`, then I open each one and check that the information has the right format and that the author didn't create the subject several times.

2. Sort the subjects by `shortName`, then I go through them watching that there are no duplicates. If I find any, I usually remove the oldest one.

8.1.4 Promote the app

Whenever I have the opportunity I mention that this app exists for people that would benefit from it. Usually, new people I meet that are studying, especially from the FIB faculty. Also during exam periods, I send it through student's WhatsApp groups and my personal Instagram stories.

8.1.5 Note down sporadic feedback

Related to promoting the app, sometimes when I share the app, the people already knew and use it. So I take the opportunity to ask for feedback, what bugs have they found and what would they change. Then I note down this feedback an when I get home add them to the to-do tasks.

8.2 Web analytics

Web analytics is the measurement, collection, analysis, and reporting of web data for purposes of understanding and optimizing web usage. It's also used as a tool for business and market research, and to assess and improve the effectiveness of a website.

In this chapter, I'm going to explain the analytics tools GradeCalc uses, and how each one is used to improve the application.

8.2.1 Algolia's search reports

The instant search engine used to search for subjects (Fig. 4.9), Algolia, sends a report through mail. Algolia offers analytics tools, but only for paid plans, the email report is the only information that this project can access. Receiving an email weekly is a bit inconvenient because it clutters the inbox. So, I configured a Gmail filter that applies the label "Algolia" and skips the Inbox tray for the reports. This way I can check them whenever I want.
The report, displayed in figure 8.1, contains very relevant information:

- **Popular searches**: These are the most searched queries. This information is used to check that those subjects information is okay, to ensure that all of the users using it will have a good experience.
- **Searches with no results**: These are the most recurrent queries without results. This information is used to create the missing subjects. If I find a subject from the FIB faculty I add it with the information in the official syllabus. Other times the searches are for other university's subjects, but I can't create those.
- **Usage**: This is the current use of the free plan. This information is used to check that the plan doesn't need to be updated.

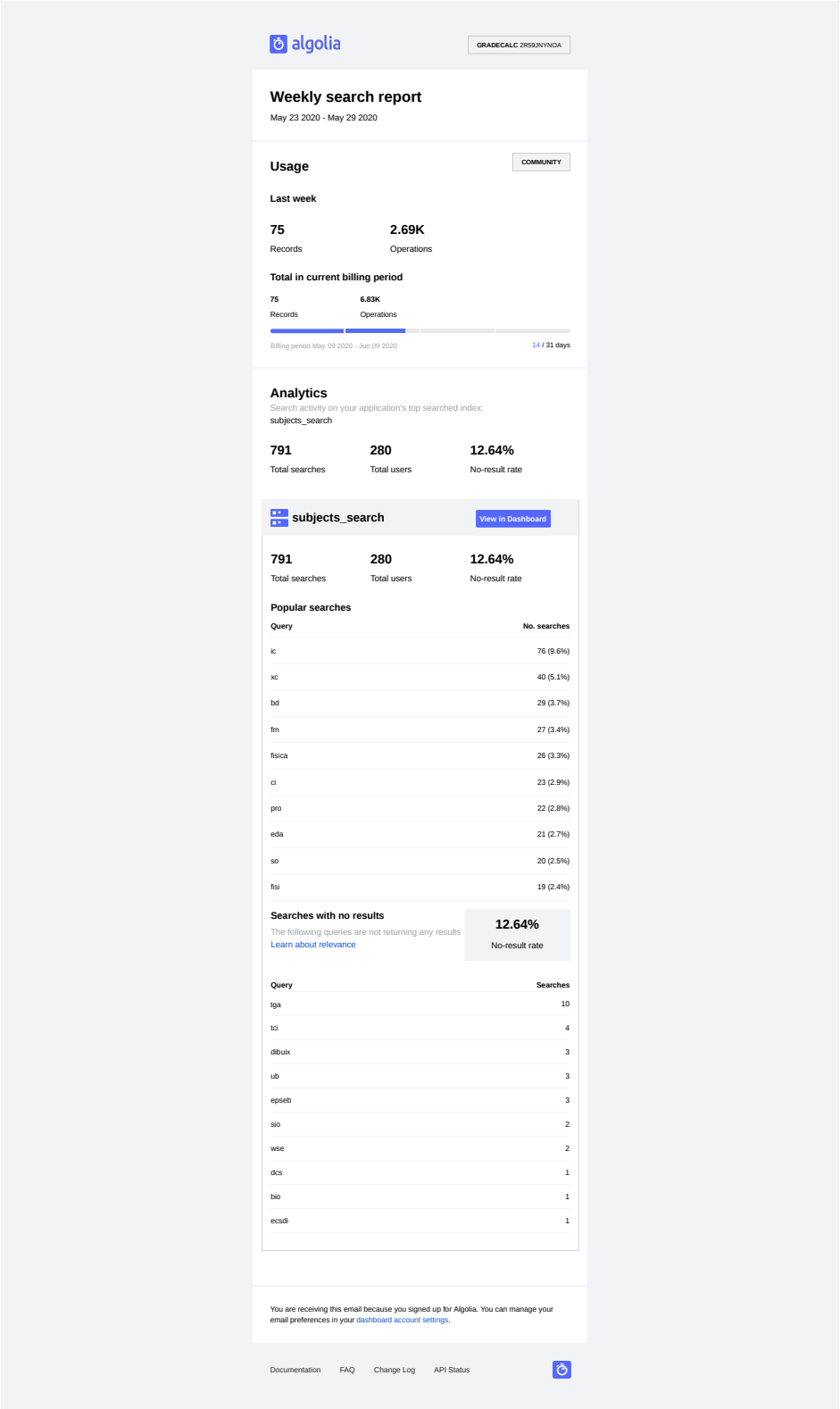


Figure 8.1 Algolia's weekly search report email

8.2.2 Google Analytics

To analyze GradeCalc's web usage, Google Analytics is used. From time to time I check it to understand the patterns that the visits follow and other information.

If we observe the visits in figure 8.3, we can see that there are peaks of activity during the final exam periods. Each time there are more visits. This last semester, there have been fewer visits during the exams period, but users have become more active during non-exam periods.

An *Avg. Session Duration* of 1 minute and 44 seconds is a good number because it means that users engage with the app.

The *bounce rate* is very high, *79.02%*, but it makes sense and it's not a bad thing in our case. Because the app usual usage happens only in the dashboard, it's normal that users just open one page and leave.

We can see that most of the users, *68.8%*, are *new visitors*. But on the other hand, approximately one-third of users return.

Finally, half of the visits come from typing the URL in the browser directly. This is a very high value, but it's executable because the app can be installed, meaning that each time a user opens it counts as direct.

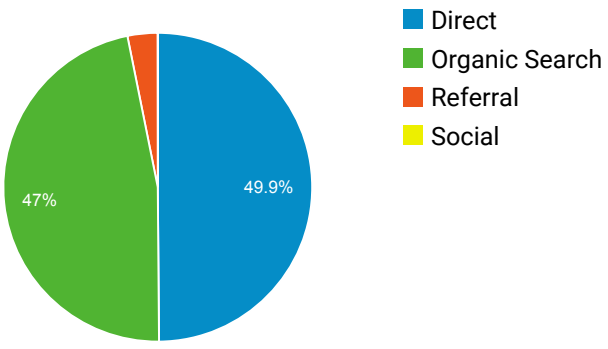


Figure 8.2 Google analytics acquisition sources

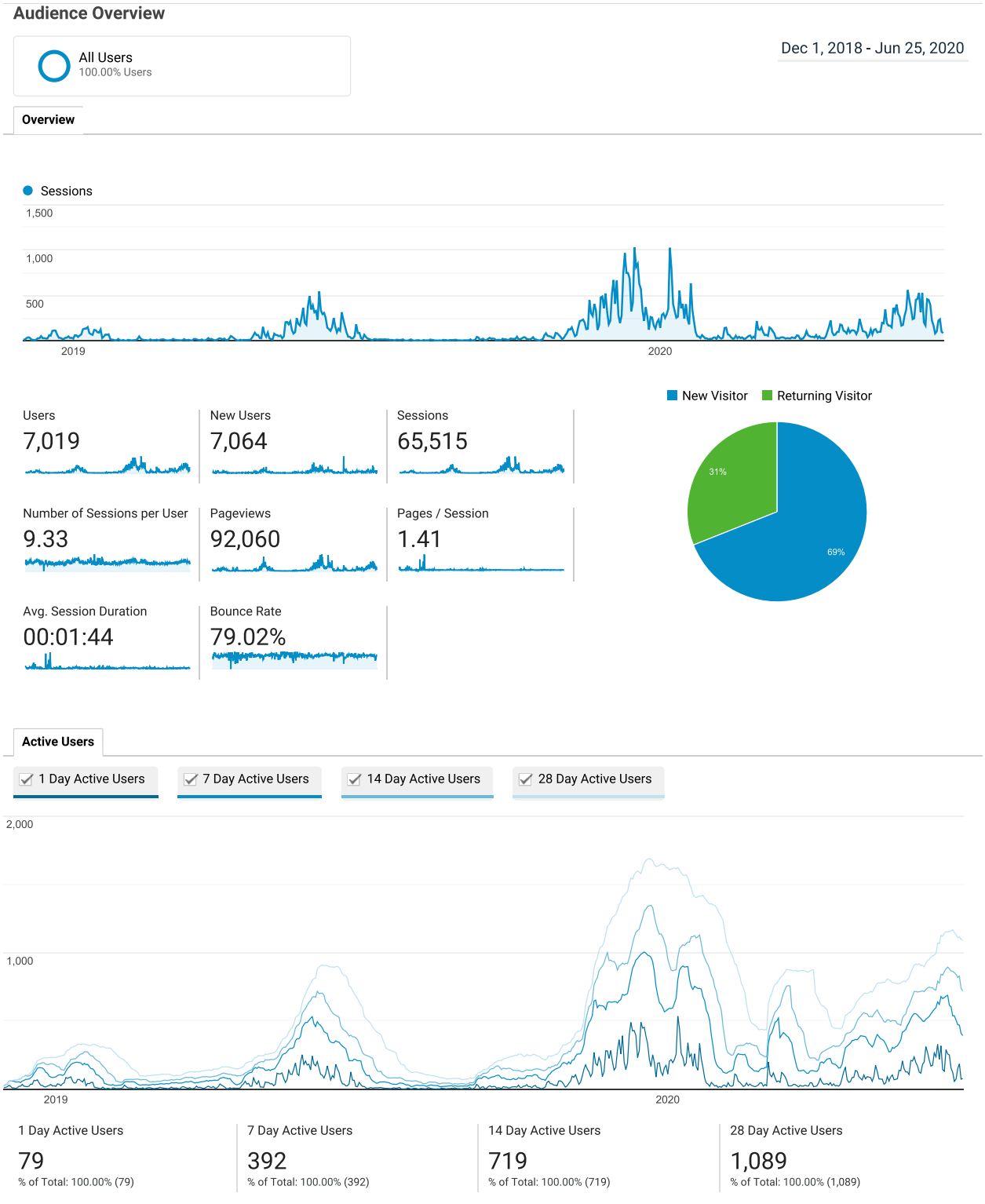


Figure 8.3 Sessions report and Active users report - Google analytics

8.2.3 Google Search Console

Finally, to ensure that the app is searchable in Google. The domain is registered in Google Search Console. If there are any crawling errors I'll receive an email. But the typical use of this app is to understand how users find the page.

In the figure 8.4 we can see that all the queries are combinations of the words *Grade* and *Calc*, in some cases with typos. And there are no queries like *Calculadora notas*, they are highly competitive queries that are not worth fighting for the first place. But if they search *Calculadora notas FIB* or *Calcular notas FIB* the app is in the first result. This second query is more likely to happen, although, as we see, none is searching for it.

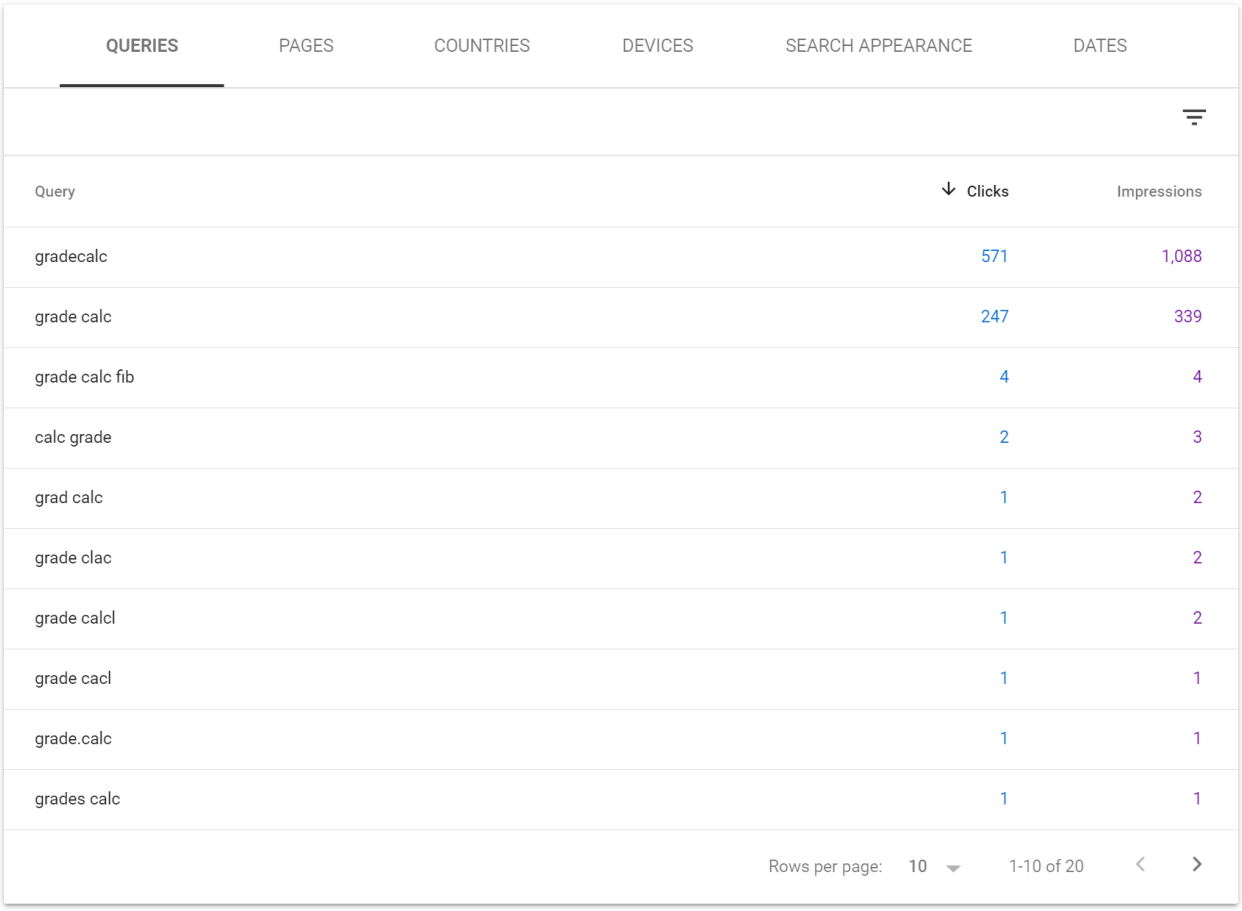


Figure 8.4 Google search console queries

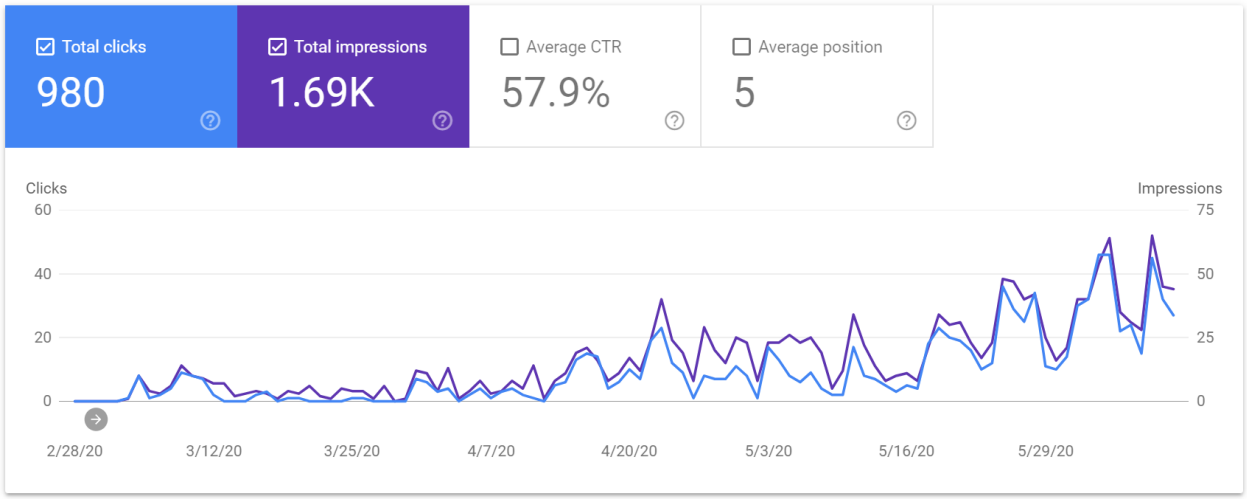


Figure 8.5 Google search console performance

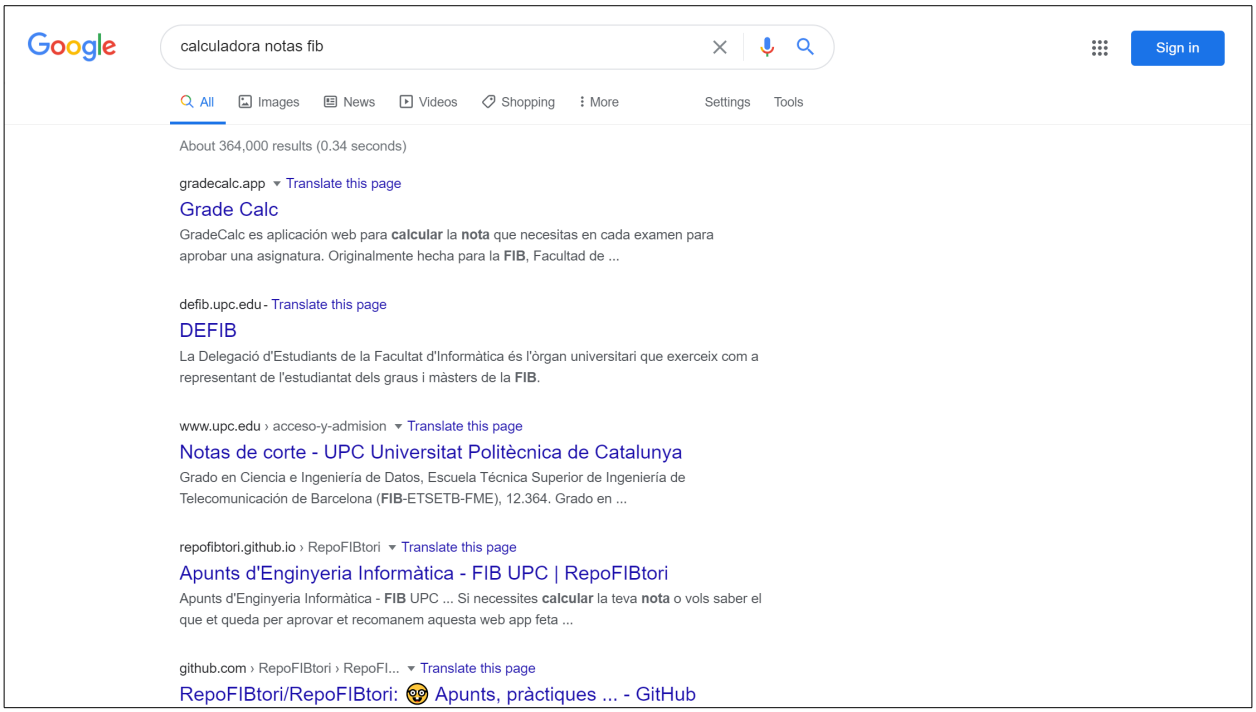


Figure 8.6 Google result for *Calculadora notas FIB*

Chapter Nine

CONCLUSIONS

9.1 Encountered issues

Although the app development has been smooth, there have been some difficulties. These are the ones that stand out the most:

- **Duplicate subjects**: When I implemented the subject's creation, some subjects were added 2, 3, or even 4 times to the database with the same information. It turned out to happen when the user clicked the create button, the app sent the subject to the database and once it got an affirmative reply it closed the create screen. But if the connection was slow this could take some seconds, and impatient users clicked the create button multiple times, leading to copies in the database. I fixed it by freezing the screen until the response was received (Fig. 4.17). The cause of this problem was very difficult to find because I could not recreate it, and I spend extra time on it.

- **Auto-update subject's structure**: The subject's object structure changed over the course of the project, adding, deleting, renaming, and moving its fields to accommodate new functionalities. Each time I had to change the specification I had to parse the old object structure into the new one of all the subject's in the database and the localStorage of the user's device. This made me spend an unplanned time on it.

- **Edit subject grid**: The evaluation grid inside the edit subject screen (Fig. 4.11) was quite complex to implement because the rows had to be added and deleted and at the end all the information parsed from the inputs into a JSON object. I underestimated this task, but in the end, I could do it.

- **Instant search**: Firebase doesn't provide full-text search [16], so I had to use Algolia. The problem was that Algolia requires a copy of the entire database, the usual approach is to make the copy at the beginning, and for future entries add them in both databases (Firebase's and Algolia's). But Firebase's free plan doesn't allow to run a function that sends data to an external source and I needed a workaround. My workaround was to set up a cron job in Heroku that runs every day and sends the new records to Algolia. This problem was a headache for some time because I wasn't coming up with any solution.

In the end, all these problems were successfully solved.

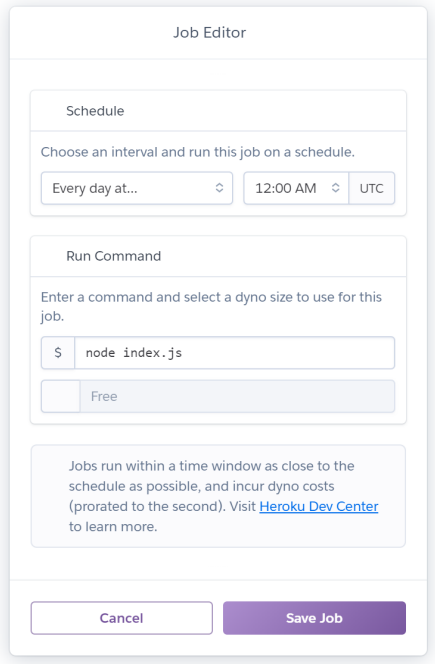


Figure 9.1 Heroku cron job configuration

9.2 Accomplished objectives

All of the objectives of the project defined in section 2.1.1, have been correctly achieved. These are justifications:

- **Add functionalities**: The app now has, shared subjects database, many small details, create and edit subjects, it's installable...
- **Migrate code to a JavaScript task runner or bundler.**: The app now uses Gulp.
- **Improve UI design**: Now the subjects are displayed in beautiful cards in the dashboard and all other components follow the same style. All the user interface is explained in section 4.2.1.
- **Brand design**: GradeCalc now has a defined brand and its style is used consistently. The details are explained in section 5.2.
- **Bug fixing**: The app now doesn't have any critical bugs and can be used without problems.
- **Add tests to the code**: Tests have been defined and added to the code. They are explained in chapter 6.
- **Implement Continuous Integration and Deployment**: The app has both Continuous integration and deployment, and it takes advantage of its benefits. The setup is explained in chapter 7.
- **Implement users accounts**: The app now uses Firebase and users can sign-up with Google to synchronize their grades, among other things.

9.3 Technical competences' accomplishment

The technical competencies defined for this project are the following.

CES1.4: To develop, maintain, and evaluate distributed services and applications with network support. [In depth]

The application is based in web technologies (section 5.1), so it has intrinsic network support.

- **Develop**: In chapter 4, all the application's software is explained and this is how it has been developed.
- **Maintain**: In chapter 8, the maintenance of the app is explained.
- **Evaluate**: In chapter 6, it is explained how the app is tested and evaluated.

This competence has been accomplished successfully.

CES1.5: To specify, design, implement and evaluate databases. [Enough]

- **Specify**: In section 4.4, the database is specified in both UML and TypeScript to provide a clearer understanding.
- **Design**: Also section 4.4, explains that the database is designed specifically for the app to use the advantages of noSQL.
- **Implement**: The final app uses the defined database.

This competence has been accomplished successfully.

CES2.1: To define and manage the requirements of a software system. [Enough]

In chapter 3, the requirements of the software system have been defined and managed.

This competence has been accomplished successfully.

CES1.2: To solve integration problems in function of the strategies, standards and available technologies. [A little bit]

The integration problems that the application faced where:

- Connecting and using Firebase's database.
- Using Google's authentication service.
- Bundling the web app into an android app with android studio.
- Setting up continuous integration and continuous deployment.
- Setting up Gulp as a bundler and using libraries.

These problems were solved in accordance with the:

- **Strategies**: In section4.1.1, the used design patterns are showcased.
- **Standards**: The standards are taken for granted, the app uses popular and loved by the community technologies that use well-established standards. Like using JSON in the request's body, or using standard UI components like popups, notification banners, cards...
- **Available technologies**: In section5.1, the technologies are chosen, taking into consideration the requirements of section 3.

This competence has been accomplished successfully.

CES1.7: To control the quality and design tests in the software production. [A little bit]

- **Control the quality**: In section 6.3, the satisfaction survey checks the quality.
- **Design tests**: In chapter 6, the tests are designed.

This competence has been accomplished successfully.

CES2.2: To design adequate solutions in one or more application domains, using software engineering methods which integrate ethical, social, legal and economical aspects. [A little bit]

GradeCalc has been designed to provide value to the students from the beginning (Chapter 1). This is how it integrates other domain aspects:

- **Social**: It modifies the way students study for exams, letting them optimize more their time and invest the saved time into other activities.
- **Ethical**: The app itself is completely transparent. It can be argued that the app makes students lazier by aiming for lower grades, but the reality is that there are many legit use cases of the app. For example, if a subject doesn't add value to a student, he may spend the minimum time on it and use the saved time in other, more valuable, activities. Or another example, a student that failed an exam at the beginning of the course will study harder to pass, making him learn more.
- **Legal**: It complies to the applicable legislation, it's explained in section 2.6.
- **Economical**: The project's costs are kept to the minimum, so this is an aspect that has been present when choosing the technologies (section 5.1) and also taken into account in the budget (section 2.4).

This competence has been accomplished successfully.

9.4 Integration of knowledge

Most of the technical knowledge used in this project has been acquired by self-learning, especially watching a lot of YouTube videos, reading articles, doing prior projects, and searching problems in stack-overflow between many more things. Nonetheless, I learned a solid base from the bachelor course in FIB UPC. These are the subjects that taught me relevant knowledge for this project.

- **ASW**: Understanding the usual architecture for web applications is a must to build one. It may have been the most relevant subject for this project, although it didn't go deep into the topic as I would have liked.
- **IDI**: The design principals they taught were really useful when designing an accessible and clean UI.
- **AS**: I learned how to evaluate code quality and many design patterns. This improved the quality of the code and facilitated pointing its flaws.
- **ER**: It was helpful when it came to identifying stakeholders, requisites, and its satisfaction criteria. They taught us how well how to those tasks.
- **SLDS**: I learned how open source software licenses work, how they can be sustainable, and how the workflow is.
- **MI**: I learned how to copywrite[1] the texts for the app, how to use analytics tools, and how to make the app accessible and google friendly.
- **Erasmus**: I improved my English language skills, which helped me write this memory.

[1]Copywriting is the act or occupation of writing text for advertising or other forms of marketing.

9.5 Future improvements

GradeCalc has already many functionalities, but as time goes on I get more ideas and suggestions. These are the improvements I'll work on in the future:

- Try to monetize the app and make it economically sustainable. I tried to add ads, but the advertising program (like Google AdSense) reject the app because it doesn't qualify for it. So, I need to find a realistic alternative to ads.
- Allow subjects to have more complex evaluation formulas. Especially I'd like to add these cases:
  - Add multipliers, sometimes thatchers may multiply your grade by 1.1 if you do a certain task.
  - Optionally, add a minimum grade in an exam to make an evaluation valid, sometimes students need to get a 4 or more in the final exam to make it valid.
  - Optionally, change the grade to pass, sometimes students need a 5.5 instead of a 5 to pass.
- Add more usage indicators (Google Analytics custom events) to understand more the user's behavior and improve the app based on that.
- Improve the dashboard. Add things like rearranging subjects, rearranging exams inside subject cards, achieving subjects without deleting them.
- Create a guided tour for new users, to help them learn to use the app.
- Fix all the known bugs.

9.6 Personal takeaways

This project has been a very gratifying, particularly, when I did the satisfaction survey (section 6.3), reading the open-ended question answers made me really happy, because the app is helping real people and I improved their lives by a little bit. This feeling of having a real impact on people is truly gratifying.

The project also taught me a ton of concepts and technologies. I learned many things, from using Firebase to setting up automatic deploys. I especially learned plenty of UI/UX tips to make the best experience possible. I also improved a lot my JS and CSS skills.

Another important lesson I learned is that it's very important to have a stable version always in production and the pressure to fix bugs as fast as possible.

Overall, doing this project allowed me to learn many little, and not so little, things. So, I can firmly state that this project made me a better software engineer.

REFERENCES

[1] Algolia. Algolia for Open Source. URL: https://www.algolia.com/for-open-source (visited on 03/04/2020).

[2] Hideaki Anno. Congratulations! - EVA ending. URL: https://www.youtube.com/watch?v=oyFQVZ2h0V8 (visited on 06/06/2020).

[3] Atlassian. Continuous integration. URL: https://www.atlassian.com/agile/software-development/continuous-integration (visited on 06/08/2020).

[4] browserl.ist. Browsers with more than >2% usage. URL: https://browserl.ist/?q=%3E2%5C%25 (visited on 03/09/2020).

[5] Tiger Color. Color Harmonies. URL: https://www.tigercolor.com/color-lab/color-theory/color-harmonies.htm (visited on 06/04/2020).

[6] intersoft consulting. art. 4 GDPR - Definitions. URL: https://gdpr-info.eu/art-4-gdpr/ (visited on 03/26/2020).

[7] intersoft consulting. General Data Protection Regulation - GDPR. URL: https://gdpr-info.eu/ (visited on 03/26/2020).

[8] MDN contributors. Introduction to progressive web apps. URL: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction (visited on 06/20/2020).

[9] Wikipedia contributors. Multitier architecture. URL: https://en.wikipedia.org/wiki/Multitier_architecture (visited on 06/20/2020).

[10] CoSchedule. How to Define Your Unique Brand Voice. URL: https://coschedule.com/marketing-strategy/brand-voice/ (visited on 04/29/2020).

[11] Daniel Eden. Animate.css. URL: https://animate.style/ (visited on 05/28/2020).

[12] Saul Edmonds. The 5 Rules of Successful Logo Design. URL: https://roundhouse.cc/us/5-rules-of-successful-logo-design (visited on 04/29/2020).

[13] Ben Eggleston. Grade Calculator - Ben Eggleston. URL: http://www.benegg.net/grade_calculator.html (visited on 03/16/2020).

[14] Figma. Nunito Google Fonts Pairing. URL: https://www.figma.com/google-fonts/nunito-font-pairings (visited on 05/29/2020).

[15] Firebase. Cloud Firestore Data model. URL: https://firebase.google.com/docs/firestore/data-model (visited on 05/22/2020).

[16] Firebase. Full-text search. URL: https://firebase.google.com/docs/firestore/solutions/search.

[17] Firebase. Get started with Cloud Firestore - Initialize Cloud Firestore. URL: https://firebase.google.com/docs/firestore/quickstart#initialize (visited on 05/30/2020).

[18] Firebase. Privacy and Security in Firebase. URL: https://firebase.google.com/support/privacy (visited on 05/12/2020).

[19] Google fonts. Nunito font. URL: https://fonts.google.com/specimen/Nunito (visited on 03/20/2020).

[20] Google fonts. Nunito font - Character Map. URL: https://www.1001freefonts.com/nunito.font (visited on 03/20/2020).

[21] GitHub. Understanding the GitHub flow. Nov. 30, 2017. URL: https://guides.github.com/introduction/flow/ (visited on 03/16/2020).

[22] Refactoring Guru. Adapter. URL: https://refactoring.guru/design-patterns/adapter (visited on 05/30/2020).

[23] Refactoring Guru. Design Patterns. URL: https://refactoring.guru/design-patterns (visited on 05/30/2020).

[24] Refactoring Guru. Proxy. URL: https://refactoring.guru/design-patterns/proxy (visited on 05/30/2020).

[25] Refactoring Guru. Singleton. URL: https://refactoring.guru/design-patterns/singleton (visited on 05/30/2020).

[26] Maurici Abad Gutierrez. FlexBox Animation move elements when one is removed. URL: https://codepen.io/MauriciAbad/pen/eQoQbK (visited on 03/01/2020).

[27] idempotency. Congratulations! URL: https://www.youtube.com/watch?v=1Bix44C1EzY (visited on 06/06/2020).

[28] Investopedia. Brand Identity. URL: https://www.investopedia.com/terms/b/brand-identity.asp (visited on 05/29/2020).

[29] Pete LePage. Snackbars. URL: https://material.io/components/snackbars (visited on 03/15/2020).

[30] Pete LePage. What does it take to be installable? URL: https://web.dev/install-criteria/ (visited on 02/14/2020).

[31] Netlify. Open Source Plan Policy. URL: https://www.netlify.com/legal/open-source-policy.

[32] Netlify. Welcome to Netlify | Neflify Docs. URL: https://docs.netlify.com/ (visited on 06/08/2020).

[33] ProductPlan. Stakeholder Analysis. URL: https://www.productplan.com/glossary/stakeholder-analysis/ (visited on 03/01/2020).

[34] RapidTables. Grade Calculator - RapidTables. URL: https://www.rapidtables.com/calc/grade/grade-calculator.html (visited on 03/16/2020).

[35] LinkedIn Salary. Frontend Developer Salaries in Barcelona. URL: https://www.linkedin.com/salary/explorer?countryCode=es&geoId=107025191&titleId=3172 (visited on 03/15/2020).

[36] LinkedIn Salary. Full Stack Developer Salaries in Barcelona. URL: https://www.linkedin.com/salary/explorer?countryCode=es&geoId=107025191&titleId=25201 (visited on 03/15/2020).

[37] LinkedIn Salary. Project Manager Salaries in Barcelona. URL: https://www.linkedin.com/salary/explorer?countryCode=es&geoId=107025191&titleId=4 (visited on 03/15/2020).

[38] SOWO. Coworking rates. URL: https://www.sowo.es/en/tarifas/ (visited on 03/10/2020).

[39] stackshare. Popular Tech Stacks. URL: https://stackshare.io/stacks (visited on 05/24/2020).

[40] Doug Stevenson. What is Firebase? The complete story, abridged. URL: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 (visited on 06/20/2020).

[41] UPC. Atenea. URL: https://atenea.upc.edu (visited on 03/16/2020).

List of Tables

2.1 Description of sprint tasks and estimations 17

2.2 Description of tasks and estimations 18

2.3 Risks and mitigations 20

2.4 Ideal team 23

2.5 Ideal hardware 25

2.6 Browsers with more than >2% usage [4] 26

2.7 Software expenses 27

2.8 Indirect expenses 28

2.9 Contingencies expenses 29

2.10 Unforeseen expenses 29

2.11 Total expenses 30

List of Figures

1.1 GradeCalc analytics - Sessions from Dec 2018 to June 2020 4

1.2 Generic grade calculators 5

1.3 Atenea UPC 6

1.4 Power-Interest grid 7

2.1 Kanban board in Jira 12

2.2 GitHub Flow 13

2.3 GitHub CI 14

2.4 Sprint Gantt diagram 17

2.5 Gantt diagram 19

2.6 Updated Gantt diagram 22

3.1 Conceptual model's UML Diagram 46

4.1 Traditional vs Firebase architecture. [40] 64

4.2 Firebase products. [40] 65

4.3 GradeCalc architecture 65

4.4 Dashboard 68

4.5 Welcome 69

4.6 Subject card 70
4.7 Confetti 72
4.8 Search subjects empty input 73
4.9 Search subjects input filled 74
4.10 No search results found 75
4.11 Edit subject 76
4.12 Create subject 77
4.13 Log-in user is not logged in 78
4.14 Log-in with google 79
4.15 Log-in user is logged in 79
4.16 Background tasks spinner 80
4.17 Blocking spinner 81
4.18 Blocking spinner taking too much time 81
4.19 Notifications 82
4.20 Installation process 83
4.21 Surprise gift 84
4.22 Surprise gift animation 84
4.23 Congratulations 85
4.24 Navigation diagram 86
4.25 Flowchart of adding a subject 88
4.26 Flowchart of logging-in 89
4.27 Flowchart of logging-out 89

4.28 Flowchart of editing a subject card 89
4.29 Flowchart of deleting a subject card 89
4.30 Flowchart of editing a grade 89
4.31 Flowchart of the installation 89
4.32 Classes diagram 90
4.33 Screenshot of Firebase Console 91
4.34 Data model's UML Diagram 93
5.1 GradeCalc's colors in the color wheel 101
5.2 GradeCalc's color palette 102
5.3 GradeCalc's subject's colors 102
5.4 Nunito font character map[20] 103
5.5 GradeCalc app icon 104
6.1 Satisfaction survey popup 118
6.2 Satisfaction survey - Introduction 119
6.3 Satisfaction survey - Section 1 119
7.1 Lighthouse score of GradeCalc 131
7.2 Pull request checks of GradeCalc 132
8.1 Algolia's weekly search report email 138
8.2 Google analytics acquisition sources 139
8.3 Sessions report and Active users report - Google analytics 140
8.4 Google search console queries 141
8.5 Google search console performance 142

8.6 Google result for *Calculadora notas FIB* 142

9.1 Heroku cron job configuration 144

APPENDIX

Appendix A

Satisfaction survey answers

Id	Question																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	5	5	4	3	5	5	5	5	5	5	5	4	5	4	5	4	5	5	5	5
2	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	1	5	5	5	5
3	5	5	5	4	5	5	5	4	5	5	5	5	5	1	3	1	4	5	5	3
4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	4	2	5	5	5	4	5	5	5	5	5	5	5	1	4	5	4	5
6	5	5	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5
7	4	5	5	4	5	5	5	5	5	5	5	4	5	5	3	3	5	4	2	3
8	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
9	4	4	1	3	4	4	4	4	4	1	4	4	4	4	4	4	4	4	4	4
10	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
11	5	5	4	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
12	4	5	3	2	5	4	5	3	4	5	5	4	5	5	4	4	5	5	5	5
13	4	5	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4
14	4	5	4	3	4	5	5	5	5	4	4	4	5	5	5	3	5	5	4	4
15	5	5	5	5	4	5	5	4	5	5	5	5	5	5	5	4	4	5	4	4
16	5	5	5	5	4	5	5	4	5	5	5	5	5	5	5	4	4	5	4	4
17	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	3	5	5	5	5
18	5	1	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
19	4	5	5	3	5	5	5	5	5	2	5	5	5	5	5	5	5	5	5	5
20	5	5	1	1	5	5	5	5	5	5	5	4	5	1	5	1	3	5	5	5
21	4	5	5	5	5	5	5	5	5	5	5	5	5	3	5	1	5	5	5	3
22	5	5	5	4	5	5	5	4	5	5	5	2	5	3	5	4	3	4	4	3
23	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	3	4	5	4	4
24	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
25	4	5	2	1	5	5	5	5	5	5	5	4	5	2	3	2	3	5	4	4
26	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
27	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	3	5	5	5	5
28	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
29	5	5	4	5	5	5	5	3	5	5	5	5	5	5	5	5	5	5	5	5
30	5	5	5	5	5	5	5	4	5	5	5	5	5	3	5	5	4	5	5	5

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
31	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4
32	5	5	5	5	5	5	4	5	4	5	5	4	5	4	4	4	4	5	5	5
33	4	5	4	3	5	5	5	5	5	5	5	4	4	3	4	2	4	5	4	5
34	4	5	3	2	5	5	5	5	5	5	5	5	5	5	5	4	5	5	3	4
35	5	1	3	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
36	5	5	5	3	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5
37	4	4	5	3	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5
38	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	4	4	5	5	5
39	5	4	5	4	5	5	5	4	5	5	5	5	5	1	5	2	4	5	5	5
40	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5
41	4	5	4	2	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5
42	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
43	5	5	5	5	5	5	5	5	5	4	4	5	5	5	5	4	5	5	5	5
44	5	5	5	4	5	5	5	5	5	3	5	5	5	5	5	3	5	5	5	5
45	5	5	5	3	5	4	5	5	5	1	5	5	5	1	5	1	5	5	4	5
46	5	5	5	3	3	5	5	5	5	5	5	4	5	5	5	3	5	5	5	5
47	4	5	4	3	3	5	5	4	5	5	4	3	5	3	4	4	4	5	5	5
48	4	5	3	4	3	5	5	5	5	5	5	4	4	4	5	2	3	3	5	5
49	5	4	4	2	5	5	5	4	5	5	5	5	5	5	5	3	2	5	3	2
50	5	5	5	5	5	5	5	5	5	2	5	5	5	5	5	5	5	5	5	5
51	5	4	4	4	5	5	5	5	5	4	5	4	5	4	5	3	4	5	5	5
52	4	5	3	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
53	4	5	4	5	5	5	5	5	5	3	3	4	5	2	5	5	5	4	4	4
54	4	5	5	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
55	5	5	5	5	5	5	4	4	4	4	5	5	5	5	5	3	4	5	5	5
56	5	5	5	4	5	5	5	5	5	4	4	5	5	5	5	4	5	5	5	5
57	5	4	5	5	5	5	5	5	5	5	5	4	5	5	5	1	2	4	5	5
58	5	5	5	3	5	5	5	5	5	3	5	4	5	3	5	3	3	5	5	5
59	4	5	4	2	5	5	5	4	5	5	5	4	4	3	4	3	3	5	5	4
60	5	5	5	5	5	3	4	5	5	3	5	4	5	5	5	3	4	3	5	5
61	4	4	4	4	4	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4
62	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	3	3	4	5	5
63	4	4	4	2	4	5	5	5	4	4	4	4	5	5	5	5	4	5	3	3
64	5	5	5	2	5	5	5	5	5	4	5	5	5	5	3	5	1	1	5	5
65	3	5	4	2	5	5	5	4	5	5	5	5	5	5	4	3	5	5	5	5
66	5	5	4	4	5	5	5	5	5	5	5	4	5	3	4	3	4	5	4	4
67	5	5	2	3	5	5	5	5	5	5	5	5	5	5	4	2	5	5	5	4
68	5	5	5	5	5	5	5	5	5	5	5	5	5	4	2	3	4	4	5	5
69	5	4	5	4	5	5	5	5	5	5	5	4	5	5	5	3	4	5	5	4