



FINAL DEGREE WORK
COMPUTER SCIENCE DEGREE

SMS

A processes accelerator for educational centers.

Author

Juan Antonio Fernández Sánchez

Directores

Juan Julian Merelo Cuervos



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, Junio de 2017

SMS: A processes accelerator for educational centers

Juan Antonio Fernández Sánchez

Keywords: microservices, cloud, education,

Abstract

This work is about the building of a simple open source students management web-based app but with technologies and a architecture opposite to the common. This work is about the design a microservices based architecture that will be able to grow, easily scalable, technology agnostic and with the enough decoupled to be functionall separately. But above all, is an excuse to learn, research, about programming, workteam manage, project structuration and planification.

Me, **Juan Antonio Fernández Sánchez**, Computer Science Degree student at **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación**, ETSIIT, at **Granada University** with DNI 45601217Z, allow the emplacement of this a copy of this work in the library of the center to be able to consult by anyone that desire.

Fdo: Juan Antonio Fernández Sánchez

Granada, 30 June 2017

D. **Nombre Apellido1 Apellido2 (tutor1)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

D. **Nombre Apellido1 Apellido2 (tutor2)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Nombre Apellido1 Apellido2 (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Acknowledgment

All this would not have been possible without all the support, patient and encourage of my family, so thank for all.

Thanks to Juan Jose Ramos, mentor at Telefonica Talentum program, for all their tips and all their help with business and technical approach, and of course to all fellows of the program, really competent professional guys, now friends also, that I hope to match in future projects.

Thanks also of course to JJ Merelo, for their help, inspiration and overall, for the opportunity to meet cool people and overall their patient with a student like me.

And especially to Susana. Thank you for being there.

Contents

1	Introduction	17
1.1	What about it?	17
1.2	Why to build a new tool?	18
1.3	Domain of the problem	18
1.4	The cost of non-digitalization	19
1.4.1	Paper	19
1.4.2	Time and Money	21
1.4.3	Value of business	22
1.5	State of Art	23
2	Requirements	27
2.1	Minimal user requirements	27
2.1.1	General needs	27
2.1.2	Academic Management System	29
2.1.3	Class Controls	33
2.1.4	Marks	34
2.1.5	Disciplinary Notes	34
2.1.6	Simple and advance reports	35
2.1.7	Autonomic Official System Connection	36
2.2	System requirements	36
2.2.1	Modularity	36
2.2.2	Communications	37
3	Planification	39
3.1	Methodology and planification.	39
4	Tools, patterns and firsts decisions	41
4.1	Architecture	41
4.1.1	Kind of software	41
4.1.2	Why microservices?	41
4.1.3	Why polyglot database?	43
4.2	Code strategy	43
4.2.1	Version Control System	43

4.2.2	Hub	44
4.2.3	Develop workflow	44
4.3	Languages and frameworks	46
4.3.1	Communications	46
4.3.2	Testing	49
4.3.3	Documentation	49
4.4	Storage	49
4.4.1	Versions	50
4.4.2	Frameworks	50
4.5	Platforms	50
4.6	Documentation	52
4.7	License	52
5	Design	53
5.1	Responsabilites distribution	53
5.1.1	Be more realistic	58
5.2	API standar status codes	60
5.3	Pesistence Strategy	61
5.4	API Gateway microService	62
5.5	Teaching Data Base microService	63
5.5.1	Domain and design	63
5.5.2	Api definition	64
5.5.3	Database logical design	67
5.5.4	Way to access to raw data	70
5.6	Students Control microService	71
5.6.1	Domain and design	71
5.6.2	Api definition	71
5.7	User Interface microService	78
5.7.1	Domain and design	78
5.8	Analysis microService	81
5.8.1	Domain and design	81
5.8.2	API Definition	83
5.9	Auxiliar tools	84
5.9.1	Provisioner	84
5.9.2	Data base helpers and testing tools	85
6	Develop	87
6.1	Api Gateway microService	87
6.1.1	Cloud Endpoint spike	88
6.1.2	As a simple dispatcher	89
6.2	Teaching Data Base microService	90
6.2.1	Optional subjects, the “class” Table.	90
6.3	Students Control microService	92
6.4	Analysis Control microService	92

CONTENTS	15
----------	----

6.5	User Interface microService	92
6.5.1	Save processes flows	92
6.6	Analysis microService	97
7	Conclusions	99
7.0.1	Scheduling	99
7.0.2	Technologies and frameworks	100
7.0.3	Design	102
7.0.4	The develop process	103
7.0.5	Opportunities	103
7.0.6	Future of the project	104
7.0.7	Open Source	104
7.0.8	Closing	105
	Bibliografía	105

Chapter 1

Introduction

1.1 What about it?

The idea behind of the develop of this software isn't only to do a useful tool for educational centers is also an excuse to research and learn about a lot of technologies and patterns, but above all it's about how to do clean, simple and easily readable software.

This software has been thought to help a lot of teachers in their work, but which is exactly the problem? And how it will help?

Until a few years ago the digitalization of education was something impossible to think, the cost of equipment and the digital illiteracy did impossible to think in to do things of another way. The only computers that we could see were inside the "*Computer room*", where students learned to use a simple text processor, received simple notions of the internet of websites, mail system or storage devices are not bad to start, but We don't talk about this. And what about teachers? Maybe a few of computers in some rooms or office allow them to send emails, maybe build a simple website or offers some resource to the students (in the better case). As we can imagine, the paper was the main support of all about management of the center.

If we think about processes in a scholar center, we can't imagine the amount of paper that is necessary to registry all about students, teachers, subjects, relationships between them and the rest of work. And the worst of this is that every three months and every year the most of this documents, papers, records, etc. will need to redo, because the majority of this relations will change and will be required record all again.

Can you imagine many papers, time and effort is necessary to do this? We will talk more about this after.

So now all people have a smartphone, the hardware is really cheap (laptops, tablets, etc.) and open a website for an inexperienced person is very easy, and almost free, in some clicks. There are a lot of systems and tools about e-learning like Moodle¹, Chamilio², etc that allow managing a lot of things like classroom, activities, exams, etc., all have listened sometimes about it but they have a specific domain and are built to a specific way.

This area is more or less knowledge for the majority of new teachers and school stuff. Many centers use some tools like this to do a lot of tasks, but in spite of it, there are another a lot of tools much very heavy that are following handmade.

1.2 Why to build a new tool?

But is really necessary build to zero a new tool to do this? Well, if we want, could reuse some preexisting tool, maybe adding some module to do all things that we detect that fault, but this presents a lot of problems, that are closely related to the architecture and licenses of this software.

Because of this, is put a lot of effort in this project in the openness of code and architecture, because this is that makes possible that the platform can change, evolve in many forks and feedback herself.

1.3 Domain of the problem

The idea to build SMS arises from the need of a management system for a specific educational center in Granada that mainly speeds up all their internal management, avoid the spend of printed paper and save huge amounts of time.

Besides this, they want a system that was helpful for making better management decisions, in marking paths for future improvements. To achieve that the system would need as core a subsystem of relationships management, teachers that impart class to students, students that are enrollments in subjects and groups or courses, and a long etcetera. This only as the base of the system, because with this would only be modeled the kind of relations that the center has.

¹Moodle is a free and open-source software learning management system written in PHP and distributed under the GNU General Public License, know more at moodle.org.

²Chamilio is a free software under GNU/GPL licensing e-learning and content management system backed up by the Chamilo Association, know more at chamilio.org.

A part of this and like the minimal requirements the center need that the system provides a simple way to do their most heavy and paper, time and money cost, the attendance control (we can start to see some requirements that the system would have). Appart of this but related they need a system to save marks related with students and disciplinary notes. Only this three things done digitally may be a small internal revolution.

1.4 The cost of non-digitalization

Many times we do not perceive the cost of the processes in which we are involved because we do not pay directly this cost or because we are so used to it that is not evident.

1.4.1 Paper

How many papers are spendend by a normal teacher in a high?

If we think that an ordinary teacher have 8 groups of students (if it works in a full-time), this means 25 hours approximately at week, and each group have 25-30 student of average. In all, there are 200 students which a teacher have relation approximately every year.

Each educational center has an own control system but normally and like in this case, there are a lot of standard documents that each teacher manages to follow the evolution of their students (if it does their work fine

We are going to make the calculus of these through the summary of the mainly processes that they do.

Marks and nottations

To begin each teacher need to follow the evolution of each student in their subject quarterly, to do this they use an official document where they write all interesting annotations about the student, from marks of partials or final exams to annotation about homework or class issues.

If the common teacher (here "teacher") have 200 students and in the Spanish system we have three quarter this make up a total of 600 sheets of paper, more another 50 by differents uses that a teacher spends commonly. Usually, each center give to teacher a book with all these pages officially formatted to this task, so the cost is upper that only the paper and ink. If we multiply this number by 40 teachers approximately, we have 2400.

Tutorizing

Each time that a parent want to check the evolution of his son the mentor of the student need to do a simple but time and money expensive process.

So, if in a medium center we have 15 mentors and each one mentoring about 30 students, and technically each of this students needs a meet with their parents each quarter, this mean $15 * 30 * 3 * 10$ subjects that the student have and that need an inform is equal to 9000 pages. Including some pages necessary for the pedagogic control (parallel system of supervision in the center).

Delays

Delays record is another process which spends paper. It is a very inefficient whole of steps. To understand this better we are give you an example, when a student is late (whatever the reason) a teacher need write in a officially formatted page when it was late and why (according to it).

After the student will move to another place of the center to give this paper to an admin staff person that will notify this to their parents (because the delay is not justified or they did not know). This process means another 125 sheets per week between all students on average.

Class attendances

All weeks all teachers that impart class to a group write in a paper all attendance controls of a class (independently of their own records). This is done by all teachers. So, in on average a center have around 1000 students and around 25 students by course, more or less 40 courses and is necessary an attendance records paper by class, so is equal to 40 documents per week and if the course has 32 weeks, the result is approximately 1280 sheets of paper.

Another center communications

In addition to the above, there is another kind of process in which big amounts of paper is spent. When the direction of center wants to communicate something, in some special days when is necessary to do this or in some activities that simply require paper. In the most of the cases, the excessive cost is only because there are not another better way to do this, only for this reason. Sometimes some so easy as email or website or simple a share document would be save a lot of money but the culture and the dynamic of many places do difficult put into practice.

Summarize

With the follow constraint:

- A scholar course has 32 weeks at year on average (32-36 with 160-180 days).
- The cost by printed page is about 0.03 cents of euros.
- We are going to consider that a common center has 1000 students and 40 teachers of average.
- More or less 25 group of students.
- Each center has 15 mentors.

So, with this and preceding conclusion, we can obtain this costs table:

Process	Per week	Per year	Cost
Marks and annotations	–	26000	780 €
Tutorizing	–	9000	270 €
Delays	125	4000	120 €
Class attendances	40	1280	38,4 €
Others	–	1000	30 €
Total			1238,4 €

Table 1.1: Equivalent stuff cost in managment tasks.

As we can see, the cost is too much, independently of the environmental reasons.

1.4.2 Time and Money

Is not only the cost of the material resources but is also the time and their money equivalence what we are trying to save. If we count the hours of work that a common teacher dedicate to do this we can estimate the cost that all student management related work required.

Stuff	Amount	Hours/Month	At year	Price Hour	Total
Teacher	40	5	1800 h	15 €	27000 €
Admin	3	10	270 h	15 €	4050 €
Direction	4	10	360 h	20 €	7200 €
Total					38250 €

Table 1.2: Cost based on staff work hours.

This would be the cost if we think in the hours that all school stuff spend in this kind of tasks and that have not been spent in projects of the own

center. Can you imagine many projects that would be doing with this huge amount of hours of work by the staff of the center?

The problem is always the same, organization, control, and focused value. And the common pattern does not have a control over all and as thinks like this, the companies lost a lot of money and resources. The control of that happens in your business is the most important and obviously is the only way to can take decisions data based.

If we add the costs of the materials as paper joined to the teacher's work hours (strictly paid), we are talking about the incredible amount of approximately **39000 €** per year. A ridiculous number thinking in the cost of software implantation.

1.4.3 Value of business

Another thing that does go unnoticed is about the values of the data that big relational centers as school manage without any insight of their value. In a standard center with 1000 students is generated around 10.000 records per month, with an essential record schemas.

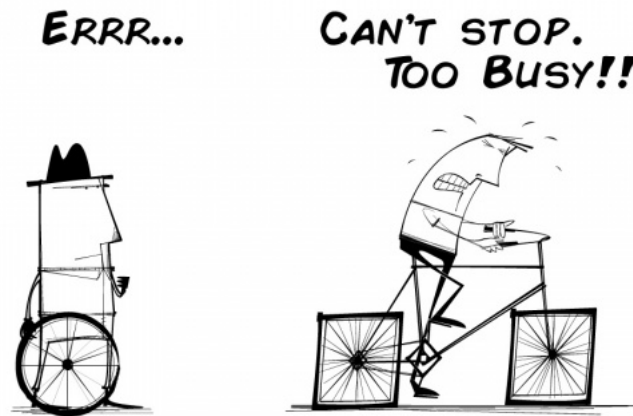
It is around 90.000 records per year and if we have a more complex system with a lot of data recorded the amount can be upper. So, what happens with all of this data. There is any value? Obviously yes, the exploitation of this data is not very clear at first because the most of people (centers directives included) not seen the center as a business, which actually, it is, only that our product is the quality of the teaching that we are offering to our students.

So, as in any manufactured process, there are a lot of steps that can be improved, only if are controlled and measured. With students is the same (without any wrong charm), there are a lot of steps, processes, and stages that could be improved to offer a better education. If we detect that by whatever reason a student have a bad trend in their evolution is very difficult to detect if we don't have a strong system to recollect data and be able to do a fast actuation (proactive in opposite of late traditional reactive behavior) will be very difficult fix the problem.

And as there are a huge amount of data is difficult that anyone can process this manually, and for this reason, is not easy too detect what part of our processes fail or need to be improved. All this problem can be easily solved with a powerful system that can be recollected, processes and give human-readable way all the conclusion obtained.

And this the other big mission of the kind of software like this, offer bunnisses

value of the data that help to manage.



1.5 State of Art

The developing of a new complete system have no sense if before have not been analyzed other existent solutions. As part of the research process has been analyzed some applications that (although is not a direct competition) represent the actual state of the art about the kind of software we are talking about.

First, we take a look at the main features that are important in our domain to see how many applications have covered them. These are a few requirements analysis but not very formal, we will discuss about this after.

In the following table, we are going to compare the rest of app with our idyllic software (with the features we hope it will cover).

Feature	classDojo	TeacherKit	A+	iDoceo	Additio	SMS
Attendance	✓	✓	✓	✓	✓	✓
Discipline		✓	✓	✓	✓	✓
Attitude		✓	✓	✓	✓	✓
Marks	✗	✓	✓	✓	✓	✓
Competencias	✗	✗	✗	✓	✓	✓
Rubrics	✗	✗	✗	✓	✓	✓
Message	✓	✓	✓	✗	✗	✓
patters	✗	✗	✗	✓	✓	✓
reports	✗	✓	✓		✓	✓
analysis		✓				✓
models and predictions	✗	✗	✗	✗	✗	✓
Centralizate and jerarqy	✗		✗	✗	✗	✓
Total	58.3%	80.5%	72.2%	72.2%	80.5%	99%

Table 1.3: Features

As we can see, there are some apps that are so focused on their specific domain that is very deficient in the rest. In spite of this are referents in their area, and their numbers of users are huge (only have been compared apps that have a good acceptance by users).

But is not only important the features of the software, because with time and effort any feature can be added. There are other things in the background that are really important too and that can be that to grow our app steadily or that crash when we have thousand of users. We are talking about something as data control, performance, scalability or centralización (we do not explain in detail each one because is out of his work but we think is also very important). And of course, all related to interface and user experience, that in most of cases can represent the difference between the successful and failure of our project.

So, based on it this is our comparative:

Feature	classDojo	TeacherKit	A+	iDoceo	Additio	SMS
Personalization	✗	✗	✗	✗	✗	✗
UI		✓	✓	✓	✓	✓
UX		✓	✓	✓	✓	✓
Price	✗	✓	✓	✓	✓	✓
Performance	✗	✗	✗	✓	✓	✓
Scalability	✗	✗	✗	✓	✓	✓
Data control	✓	✓	✓	✗		✓
Multiplatform	✗	✗	✗	✓	✓	✓
Centralization	✗	✓	✓		✓	✓
Total	44.4%	50.0%	16.6%	16.6%	44.4%	99%

Table 1.4: Architecture and design

Obviously, only in the hypothetical case of our app is fully developed will cover all the aspects considered, but before of the research is easy to understand that is not so difficult cover all with the correct design, architecture or design.

And we are going to try to give a simple and really little approximation of the way to achieve that.

So, **let's start!**

Chapter 2

Requirements

In this chapter we are going to describe which are the requirements that actually the system has to do that is expected from it. It will be used User Stories methodology (reference a book) which it will be talked a bit more in the next chapter. This tool will be useful to identify the requirements of the system by a human readable way. Removing all possible layer between design and user requirements.

User stories can be very minimal (unit) and very general. That means that we can do an user story that be a simple requirement or a very general issue. If the criteria to follow is to do very minimal user stories we could be hundred of them, so in this case, is better have upper abstraction level to have less user stories but with more content, at least when is the first time that the team uses this methodology.

2.1 Minimal user requirements

The managers of the center can't have a precise idea about which must be the functions of the software. They have a clear idea about what kind of process in their daily work there are the most heavy and consume more paper, that are which they want to digitalize, but they haven't think about navbars, icons, colors palettes, design of user interfaces or something like this. They have abstract ideas that will need dimensioning and shaping but which in synthesis offer the minimum system requirements.

2.1.1 General needs

How is talking about software, it will need run in somewhere, and they what to be cheap and easy to run. And at first there aren't any prerequisites. Maybe they prefer that this can run in devices like smartphones, tablets, laptops, but they don't have a good technical knowledges and is indifferents that be a native app, a web app or some strange artefact. For

other hand exists an essential requirements, must be run in any device for the same person, independently of the machine where it be (sometimes in laptop and sometimes in smartphone, for example).

As user, I want to use the app in any device.

Acceptance criteria:

1. Must be accesible from:
 - (a) Smartphones
 - (b) Tablets
 - (c) Laptop
 - (d) Standar Computers
2. The item must be inserted, readed, updated and deleted.

GN #1 - High priority

As user, I want to load in the system data from another databases and formats.

Acceptance criteria:

1. Must be accesible from:
 - (a) Smartphones
 - (b) Tablets
 - (c) Laptop
 - (d) Standar Computers
2. The item must be inserted, readed, updated and deleted.

GN #2 - High priority

As user, I want to get the database of the app to migrate if was necessary

Acceptance criteria:

1. Must be accesible from:
 - (a) Smartphones
 - (b) Tablets
 - (c) Laptop
 - (d) Standar Computers
2. The item must be inserted, readed, updated and deleted.

GN #3 - High priority

2.1.2 Academic Management System

As base of requirements process have been used user stories.

As manager, I want to save students to can to register them.

Acceptance criteria:

1. The item must be this fields.
 - (a) Name
 - (b) Surname
 - (c) Birthdate
 - (d) Card-id
 - (e) Location
 - (f) Address
 - (g) City
2. The item must be inserted, readed, updated and deleted.

AMS #1 - High priority

As manager, I want to save teachers to can to register them.

Acceptance criteria:

1. The item must be this fields.
 - (a) Name
 - (b) Surname
 - (c) Card-id
 - (d) Address
 - (e) Location
 - (f) City
 - (g) Birthdate
 - (h) Phone
 - (i) Image
 - (j) Gender
2. The item must be inserted, readed, updated and deleted.

AMS #2 - High priority

As manager, I want to save subjects to can to register them.

Acceptance criteria:

1. The item must be this fields.
 - (a) Name
 - (b) Description
2. The item must be inserted, readed, updated and deleted.

AMS #3 - High priority

As manager, I want to save classes to can to can to register them.

Acceptance criteria:

1. The item must be this fields.
 - (a) Course
 - (b) Word
 - (c) Level
 - (d) Description
2. The item must be inserted, readed, updated and deleted.

AMS #4 - High priority

As manager, I want to save relation between subjects and classes.

Acceptance criteria:

1. The item must be inserted, readed, updated and deleted.

AMS #5 - High priority

As manager, I want to save relation between teachers and the relation between subjects and classes.

Acceptance criteria:

1. The item must be inserted, readed, updated and deleted.

AMS #6 - High priority

As manager, I want to save relation between students and the relation between subjects and classes.

Acceptance criteria:

1. The item must be inserted, readed, updated and deleted.

AMS #7 - High priority

As we can imagine, a center have a lot of differents kind of relationships, and as obligatory part of the system it must be offer a simple way to do this.

Teachers imparts subjects to group of students (called classes), students are enrolled in subjects, and so on. These cases of use and all it can see below will be revised as *User Stories* in detail later.

Attendance Controls

One of most important process to digitaly, the amount of data tha is saved on papper and also imposible to manage before and much less do it an analysis. They need a simple way, a simple interface to do this that allow this save time (doing it, analysis and mangement it), paper and effort.

As teacher, I want to save attendance of students to can to register them.

Acceptance criteria:

1. The item must be this fields.
 - (a) Attendance
 - (b) Delay
 - (c) Justification
2. The item must be inserted, readed, updated and deleted, in a list of all students of the class.

AC #1 - High priority

2.1.3 Class Controls

A part of manage of classrooms they must offers a say to follow the students evolution in class, behaviour, positives and negativss and another lot of things like paper save, time and effort as section above.

As teacher or educator, I want to know most common disciplinary notes and data related.

Acceptance criteria:

1. The item must be this fields.
 - (a) Course
 - (b) Word
 - (c) Level
 - (d) Description
2. bla bla bla.

DN #2 - High priority

2.1.4 Marks

As teacher or educator, I want to record bad behaviour of a student.

Acceptance criteria:

1. The item must be this fields.
 - (a) Course
 - (b) Word
 - (c) Level
 - (d) Description
2. The item must be inserted, readed, updated and deleted.

MRKS #1 - High priority

As another of proccess that more paper consumes is the marks management. They want a system that simplity the proccess of insertion, analysis and management. Without any specially idea in mind are open to any good user interface that simplify this.

2.1.5 Disciplinary Notes

It also must provide a management of this kind of notes, in which a bad behaviour of a studnet is saved, managed and fully reported to the specific users inside of app, as tutors, pedagogue, etc, for example.

As teacher or educator, I want to record bad behaviour of a student.

Acceptance criteria:

1. The item must be this fields.
 - (a) Course
 - (b) Word
 - (c) Level
 - (d) Description
2. The item must be inserted, readed, updated and deleted.

DN #1 - High priority

As teacher or educator, I want to know most common disciplinary notes and data related.

Acceptance criteria:

1. The item must be this fields.
 - (a) Course
 - (b) Word
 - (c) Level
 - (d) Description
2. bla bla bla.

DN #2 - High priority

2.1.6 Simple and advance reports

Another part interesting for they is improve the reports that they obtains from their data. The on paper support do this almost imposible to big scale, an a important feature must be do this posible. Advanced reports about a lot of kind of items, like students, state of subjects, marks, etc, in seconds with some cliks. This will improve the take of decisions and will do better meetings with better decisions based on good and crontastable data.

An user stoire by subsystem: repot by marks, report by attendance control.. reports by.., etc ,etc etc....

As any person of center I want to have simple and advanced reports about the state of the student.

Acceptance criteria:

1. The item must be this fields.
 - (a) Change
2. bla bla bla.

SAR #1 - High priority

2.1.7 Autonomic Official System Connection

The national system of education in which this center is framed have a diferents informatic reginal system. In this case, in Andalucia, where the certer is it called SENECA (other in other places of the country). All public and semipublic educaitional centers need save data in this systems necessarily, and this haven't a simple public API where connet us. They must be done dirty way, making a maxed mode way to simplify the download and download of data that minify the interaction with the official system.

As a teacher, I need not interact manually with the official system.

Acceptance criteria:

1. The item must be this fields.
 - (a) Course
 - (b) Word
 - (c) Level
 - (d) Description
2. The marks of students must be upload automat

SNC #1 - High priority

2.2 System requirements

2.2.1 Modularity

As user, I want to use the app in any device.

Acceptance criteria:

1. Must be accesible from:
 - (a) Smartphones
 - (b) Tablets
 - (c) Laptop
 - (d) Standar Computers
2. The item must be inserted, readed, updated and deleted.

GN #1 - High priority

2.2.2 Communications

JSON

Acceptance criteria:

1. Must be accesible from:
 - (a) Smartphones
 - (b) Tablets
 - (c) Laptop
 - (d) Standar Computers
2. The item must be inserted, readed, updated and deleted.

GN #1 - High priority

STATUS

RESPONSE

Acceptance criteria:

1. Must be accesible from:
 - (a) Smartphones
 - (b) Tablets
 - (c) Laptop
 - (d) Standar Computers
2. The item must be inserted, readed, updated and deleted.

GN #1 - High priority

Chapter 3

Planification

3.1 Methodology and planification.

Eleccion de metodologias agiles basadas en Scrum para la gestion del equipo y en historias de usuario para el modelado del sistema.

Explicar que se usará SCRUM, el burndown, etc, ect...

Para 6 meses, a sprint por quincena, 12 sprints +1 cierre.

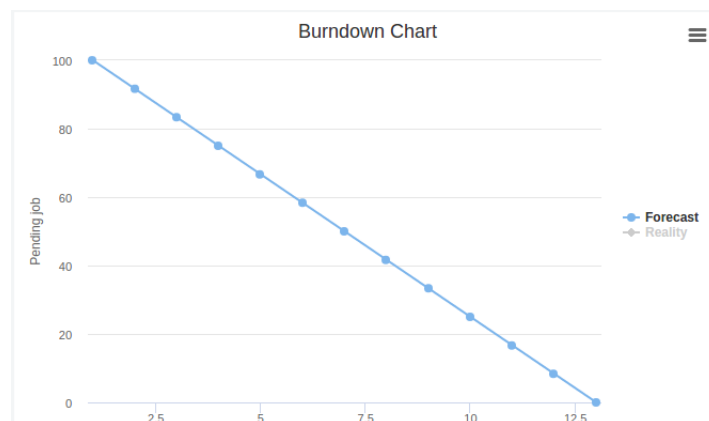


Figure 3.1: Students Control service parts.

HACER GRÁFICA DE BURNDOWN

Chapter 4

Tools, patterns and firsts decisions

Nowadays we can build almost any kind of software with a lot of different languages from which to choose, but as if that were not enough there are also a lot of architectures that are possible to follow, and beside of this it will be necessary choose where to deploy our software, or how doc it, or how to manage our work team, etc. A lot of options that it will be necessary to choose and that can make the difference between the success or failure of our project. So, what about of these decisions in this project? Take a look at it.

4.1 Architecture

Almost any kind software or tool can be built of different ways, independently of this behavior or the goals that it must achieve. Some ways are more specific to achieve some specific behavior and another.

4.1.1 Kind of software

Based on informal requirements was thought that the best option must be a **web app** for a lot of reasons. First of all because is the simplest way to offer an app that can run on almost any device (with a *simple*¹ browser).

4.1.2 Why microservices?

When we think about an app, of any type, the most of the time we think of a software running on a single machine, with more or less hardware

¹Actually a browser is one of the more complex pieces of software, but here is labeled as simple because is a software that the majority of devices like smartphones, tablet, etc, have by default and for the most nontechnical people isn't a complex software, nothing could be further from the truth.

available, where all related to the software are inside of the same machine.

Now, we hearing all the time about microservices, which are in synthesis the opposite of the traditional monolithic classical approach in software architecture, and seem like if your design is not based in this mean you are outdated or your design is directly wrong. Well, this is not totally true but neither false. The goal of this section is not describe the advantages and drawbacks of this approach, but yes justify why is their choice.

Microservices are actually a variant of SOA²,that simplifying a lot is the architecture of a software system composed of few different systems, working together to do the service which the software was design. Dr. Peter Rodgers introduced the term "Micro-Web-Services" during a presentation at Cloud Computing Expo in 2005 and as we can see in the following graphic (extracted using sotagtrends.com) community interest about this emerging technology has never stopped growing,

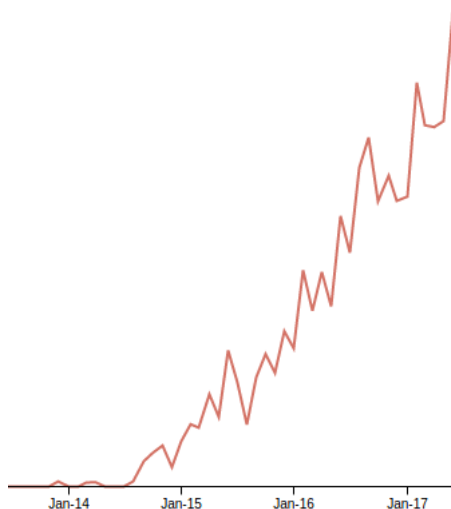


Figure 4.1: Search trend of "microservice" tag in Stack Overflow.

James Lewis³ and Martin Fowler⁴ are the major defensor of this architecture and this brings us to the canonical definition of microservices:

²Service-oriented architecture (SOA) was first described by Gartner in 1996 and is a style of software design where services are provided to the other components by application components, through a communication protocol over a network

³Principal Consultant at ThoughtWorks and member of the Technology Advisory Board

⁴British software developer, author and international public speaker on software development, specializing in object-oriented analysis and design, UML, patterns, and agile software development methodologies, including extreme programming.

"In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralised management of these services, which may be written in different programming languages and use different data storage technologies."

(Lewis/Fowler)

Language agnostic, scalable, size ajustables, independently, our system design splited in litle pieces with this boundaries do this architecture perfect for our goals, and this is why did this choice.

4.1.3 Why polyglot database?

With the mircroservices approach the system will be some very differents database to do some diferents things. So, in general we can see our backend like a black box when the data persists in a polyglot database. That means that the data is save in differents ways, using diferetns formats and diferents driver to manage this. There are a group of data that it need saved with certain relations, due to its nature, so a relational database seem perfect to do this, but this kind of databases (like MySQL) can be slowly or too heavy for other tasks of kind of processes, like data analysis.

4.2 Code strategy

Entire project will be stored in a single repository, using git as control versions system.

4.2.1 Version Control System

Git⁵ is the perfect tool to achieve maintain a more o less easy workflow between developers minimizing the risks, and the huge list of plugins to different software and IDEs do this perfect to work, dismissing another as Subversion, Mercurial, Fossil, or Bazaar.

The reason to use a single repository is that the mechanism to maintain the consistency between projects is not easily enough to do the develop agile. In spite of, git offers us a lot of tools that can improve enormously our

⁵Git is a free and open source, distributed version control system created by Linus Torvalds

workflow, as the Git Submodules⁶, or Git Hooks⁷ between another.

4.2.2 Hub

For another hand we have to do another choice, select the remote hub of our repositories or Git repository hosting service. The most popular for some years has been GitHub, a web-based Git or version control repository and Internet hosting service. It offers all of the distributed version control and source code management (SCM) the functionality of Git as well as adding its own features, launched at 2008 but there are other options as Bitbucket⁸, SourceForge⁹ and especially GitLab¹⁰ (with a big grow ultimately).

4.2.3 Develop workflow

Another of the main problem easy to find in the develop when are involved few people is the organization of the contribution of a repo, but if the project is composed of few subprojects, this can be worst. For this reason was choose the strategy to have an only repo with subfolders. So, took this decision remains to be defined how will work the repo in the different phases or the develop.

To do this, we will follow the standard, using branches to develop features and maintain a good state of the versioning of project.

So, as we can see in the picture, we are going to use *develop* branch to do the develop, where we are going to work in minimal changes and will do a fork when we want to do some representative change in the code, as new functionality or some big correction.

⁶Submodules allow you to keep a Git repository as a subdirectory of another Git repository. This lets you clone another repository into your project and keep your commits separate.

⁷Git has a way to fire off custom scripts when certain important actions occur. There are two groups of these hooks: client-side and server-side. Client-side hooks are triggered by operations such as committing and merging, while server-side hooks run on network operations such as receiving pushed commits

⁸web-based hosting service that is owned by Atlassian, used for source code and development projects that use either Mercurial (since launch) or Git launched at 2008

⁹another system launched at 1999

¹⁰launched at 2011, was written by Dmitriy Zaporozhets and Valery Sizov from Ukraine. The code is written in Ruby. Later, some parts have been rewritten in Go.

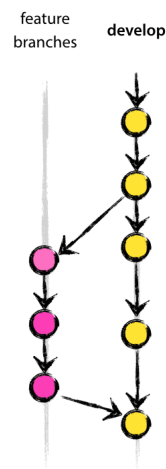


Figure 4.2: Workflow with features branches

For other hand when the all code to launch a final release we will fork the repo in a *release* branch to while the team continue with the develop in the mainly develop branch ot in a issue form of this another part of the team (or itself, is the same), will prepare the code to production, taking a look because there are some bug, fix it and when all are testend an correctly working put the project in exactly this verison in master (doing a merte) and this is when officially a new version of program will be launched.

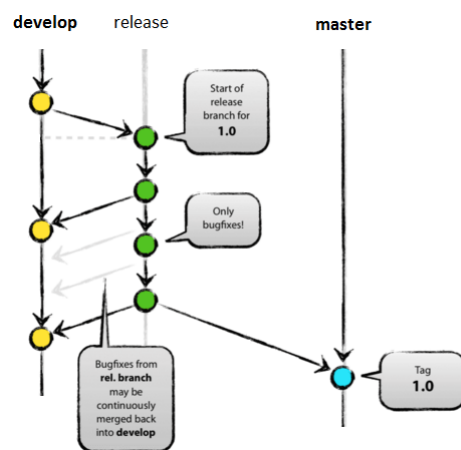


Figure 4.3: Workflow with releases branches.

There are some variations of this standard workflow but for this kind of project is perfect.

4.3 Languages and frameworks

As has been said before, almost any kind of software can be builded at infinite ways, and this start with what language can do this. This means it could be builded with Java, Ptyhon, C++, Go, Ruby, JavaScript, PHP, (only talking about backend) and with another list to frontend, in this deep list of possibilities, we choose the most flexible and powerfull of all of them, Python and JavaScript.

Python because is one of hte most simplest and powerfull languages nowadays, and JavaScript because is all an standard in the industry. Obviously the choise is based also in the fact of both languages are really supported of community, have a good learning curve and a lot of projects and systems are based in them.

JavaScript has been selected becuase Angular is writte in it. AngularJS is the most powerfull framework nowadays to buil fas, clean and powerfull web apps.

4.3.1 Communications

To exchange data between service we need to select a idiom which the services will talk, which them will exchange information. That's mean mainly select how will transform the objects anda data structures that servicres manage to be able sender across the net.

Data serialization format

We have some data serialization solutions, some very popular and another most specific of very focused domains. So, the most commons are Json, Yaml, Bson and ultimately MessagePack. Each have their owns benefits and drawbacks but the selection has been easy, JSON.

XML

Extensible Markup Language, is a markup language, defined v1.0 by W3C¹¹ This is an example:

```
<exam>
  <result>5.8</result>
  <type>Partial</type>
  <subject>Science</subject>
```

¹¹World Wide Web Consortium, founder by Tim Berners-Lee at 1994 at MIT (Massachusetts Institute of Technology) the consortium is made up of member organizations which maintain full-time staff for the purpose of working together in the development of standards for the World Wide Web.

```
<date>17-06-2018</date>
</exam>
```

JSON

JavaScript Object Notation, is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs Douglas Crockford originally specified the JSON format in the early 2000s; two competing standards, RFC¹² 7159 and ECMA¹³-404, defined it in 2013. The ECMA standard describes only the allowed syntax, whereas the RFC covers some security and interoperability considerations.

In spite of A restricted profile of JSON, known as I-JSON (short for "Internet JSON"), defined in RFC 7493, is not as popular as original. This is an example:

```
{
  "title": "The Picture of Dorian Gray",
  "author": "Oscar Wilde",
  "date": "July 1890"
}
```

YAML

Yet Another Markup Language is commonly used for configuration files, but could be used in transmission also, YAML 1.2 is a superset of JSON, which Latest release 1.2 (Third Edition) was published at (1 October 2009; 7 years ago), YAML was first proposed by Clark Evans in 2001.

```
---
invoice: 34843
date   : 2001-01-23
```

BSON

Binary JSON, is a computer data interchange format used mainly as a data storage and network transfer format in the MongoDB database. MongoDB represents JSON documents in binary-encoded format called BSON behind the scenes. BSON extends the JSON model to provide additional data types, ordered fields, and to be efficient for encoding and decoding within different languages.

MessagePack

¹²Request for Comments, is a type of publication from the Internet Engineering Task Force (IETF) and the Internet Society (ISOC), the principal technical development and standards-setting bodies for the Internet. were invented by Steve Crocker in 1969 to help record unofficial notes on the development of ARPANET

¹³Ecma is a standards organization for information and communication systems founded in 1961 to standardize computer systems in Europe.

Byte array is an efficient binary serialization format, like JSON but faster and smaller.

```
{"compact": true, "schema": 0}
27Bytes

82 A7 compact C3 A6 schema 00
18bytes
```

And another aprox as zerorp, It builds on top of ZeroMQ and MessagePack

Protocol

Some introdfdsf dafdaf adsf ds fdsf da fadf dsdsfdfsdfs aad adsf ads adsf adsf afs

APIRest

Representational state transfer (REST) or RESTful Web services are one way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations. Other forms of Web service exist, which expose their own arbitrary sets of operations such as WSDL and SOAP

RPC

In distributed computing, a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in another address space (commonly on çanother computer on a shared network), which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction.

gRPC

gRPC is an open source remote procedure call (RPC) system initially developed at Google. It uses HTTP/2 for transport, Protocol Buffers as the interface description language, and provides features such as authentication, bidirectional streaming and flow control, blocking or nonblocking bindings, and cancellation and timeouts. It generates cross-platform client and server bindings for many languages.

We will talk a bit more about this in the Develop chapter.

4.3.2 Testing

The tool that we will use for any testing involved in the backend of the project will be PyTest. Launched at 2004 by Holger Krenel is one of the most complete suites for testing over Python. Is really easy to use and allow do some things that is very difficult with another frameworks as built-in *unittest* python package, somethings as the use of fixtures or the amount of plugin that it has.

Focused on the problem it will be used to make all unittest of the core of services, libraries and auxiliary programs and testing the entire functionality of the service when it has a role of black box.

4.3.3 Documentation

As is saided the tool Sphinx is used to build the doc of the service. That basically inspect the code files mixing this with all the files that we write (pure doc) to show this as a web based documetnation (easy to read and understand).

4.4 Storage

With the storage occurs the same, there is a huge list of options to choose. The fast answer in the most of the cases is: Why do not use a relational database, as MySQL for example? If we are thinking in a little system maybe a good choice, always that we data model is adjusted to this kind of system. But every people knows the deficits and the complexity of developing a system with this database. Mainly the performance when we are talking about million of arrays of objects stored. So, as we are talking before, our goal in this project is work with a lot of data, which apart are pure relational and another can be processed by another way.

If you have a logical data model like as this project have is not easy choice one of all database engine to model this. SQL systems is very powerful for some things but not for another (or not simply) and object oriented databases is really powerful and simple to develop with it when the data haven't a lot of relations (although can be modeled also). so, why choice one between them? Why do not both? This is the approach selected, build a system with a Polyglot Database, that means: much better select the better engine for any kind of data instead of trying to use the same for all.

And this approach joined with microservices architecture give as the first conceptually of our system, when each microservice of domain work with their data, using for this their own database engine, their own kind of data,

and until their own language if is precise.

So, focused on this project, we are going to use mainly two systems, SQL relational database system in a service (we will talk more about it after), and a NoSQL service, in this case, Google Datastore, a fast and lightweight engine to manage an object oriented database.

At the moment of to write this chapter had been evaluated MongoDB as the best choice, but the facilities of the platform selected (detailed after) was made that the G. Datastore was selected finally. Beside of this if the project are rebuild now have not doubt, Mongo will be selected after the experience (the reasons will be explained in more detail in the conclusion of the work).

4.4.1 Versions

Talking about Python has selected 2.7 version not for the best reasons. And is because the sandbox of Google App Engine do not allow another upper versions.

About Angular the selectioni Angular1.x. Seem a bit old, the new version 2.x is really popular now and the community is talking about the next two levels uppper version AngularJS 4.x.

The selection of the lower version is simple, a lot of interesting material about this, the most of problems that we can find already solved or with a lot of help and the maintenance.

4.4.2 Frameworks

Flask

At the moment to write this work has been used it but exists another that are been used in another project that have more potentia because have better performance, as Falcon (betweenr anothers).

4.5 Platforms

Today, with the explosion of the cloud is needed a lot of infrastructure that support all that all companies that want to migrate thier bunisses model to cloud need. the possiblite come from of the companies that offer solutions to do this, and another that after was introduced in the same bunisses. So, actually there are a lot of companies that offer some solutions to software (not necessary) companies to work with cloud to build their solutions and cloud computing easily.

There are a lot but only few are the big in this business. And always respond to some model IaaS (Infrastructure as a Service) AWS, G Computer Engine, Rackspace Cloud, vCloud PaaS (Platform as a Service) Google App Engine, Heroku, SaaS (Software as a Service) Google Docs, Salesforce, Dropbox, Gmail

Amazon Web Services

Launched at 2006 The most popular include Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3)

Microsoft Azure

is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. It provides software as a service (SAAS), platform as a service and infrastructure as a service and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems. Azure was announced in October 2008 and released on February 1, 2010 as Windows Azure, before being renamed to Microsoft Azure on March 25, 2014.

IBM BlueMix

is another platform.

Heroku

Is a cloud platform as a service (PaaS), founded by James Lindenbaum in 2007 at San Francisco, California, supporting several programming languages that is used as a web application deployment model. Heroku is said to be a polyglot platform as it lets the developer build, run and scale applications in a similar manner across all the languages. Heroku was acquired by Salesforce.com in 2010 for 212 million of dollars.

CloudFoundry

Is an open source, multi cloud application platform as a service (PaaS) governed by the Cloud Foundry Foundation, a 501(c)(6) organization. The software was originally developed by VMware and then transferred to Pivotal Software, a joint venture by EMC, VMware and General Electric launched at 2011

Google Cloud Platform

Is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube. Alongside a set of management tools, it provides, a series of modular cloud services including computing, data storage, data analytics and machine learning.

And is this which was selected.

4.6 Documentation

To do the documentation of the code as in the rest of sections, we have a few options to choose. So, as the backend is fully developed in Python we are going to use some python especially documentation framework and in this case Sphinx¹⁴ has been selected.

For another hand, to the front, especially talking about Javascript with AngularJS we do not have the need to a doc because the docstrings have been sufficient, and neither have been found any really good tool as Sphinx for them.

4.7 License

In this project, we are going to use GNU General Public License v3 (GPL-3) to license our code. There are a lot of licenses available, but it meets all our needs, this is a little summary of this:

Can	Cannot	Must
Commercial use	Sublicense	Include original
Modify	Hold liable	State changes
Distribute		Disclose source
Place warranty		Include License
Use patent claims		Include Copyright
		Include Install Instruction

Table 4.1: License characteristics summary.

¹⁴Sphinx is a simple and powerful Python-based documentation generator.

Chapter 5

Design

Once the requirements have been decided, is time to make the design of the system that will do all needed to cover them. First, we are going to take a look over the general architecture, the distribution of the domain of the services and how to delimit their behavior and communications with each other, by after, take a look to each one to know how to work in each domain and how have been solved all problems that this design present.

5.1 Responsabilites distribution

At the moment that we decided split the functionality of the system between services we found the problem that how to decide which is the domain

At the moment that we decided split the functionality of the system between services, we found the problem that how to choose which is the domain of each one. In the majority of the cases, we are beginning from a monolithic system that we want split, in this instance, is some like this but at first, it was thought to be splited.

In a common app (whatever kind of it) we have three well-defined layers, user interface, logic and storage, something as common MVC¹ A part of user interface make the necessary to present the info and offer the way to interact with this to the user while the logic layer do this possible doing the logic steps to work with data structures saved in some kind of storage.

There are some frameworks as Django wherein all this layer can be de-

¹Model-view-controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to and accepted from the user, user interfaces pattern, introduced by Trygve Reenskaug into Smalltalk-76 at Palo Alto Research Center in the 1970s

veloped together, is not strange their popularity, but if we use other as AngularJS this only help us in the user interface layer, not in the rest, but for other hands have another advantage.

So, that as it may, this would be the more simple schema to start to work in our domain, with the difference that we work with service instead of layers in a single program, but essentially, the concept is the same.

As we can see in the next picture, we will have a user interface, when will built all views and controllers that work with this to build a complete user interface, a logic section layer, here labeled as APIGmService (as a first approach) that will do the logic of the system, transform the data, restructuring and offering the interaction form UI to database. And at end, the database system, mainly an engine of the database and the drivers necessary to work with them.

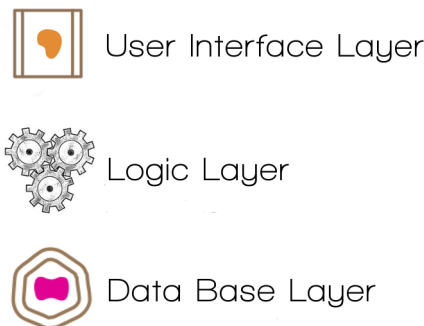


Figure 5.1: Basic layers in common app.

As we can imagine this work fine, is compact and stable when we are talking about the same code on the same machine but now we are going to split that in different services, maybe running at different places and maybe in different languages, so when we talk about divide the domain is about split the logic with the database access layer.

Next picture shows us perfectly what are talking about. This is an example of migration from monolithic to microservices from Nginx resources.

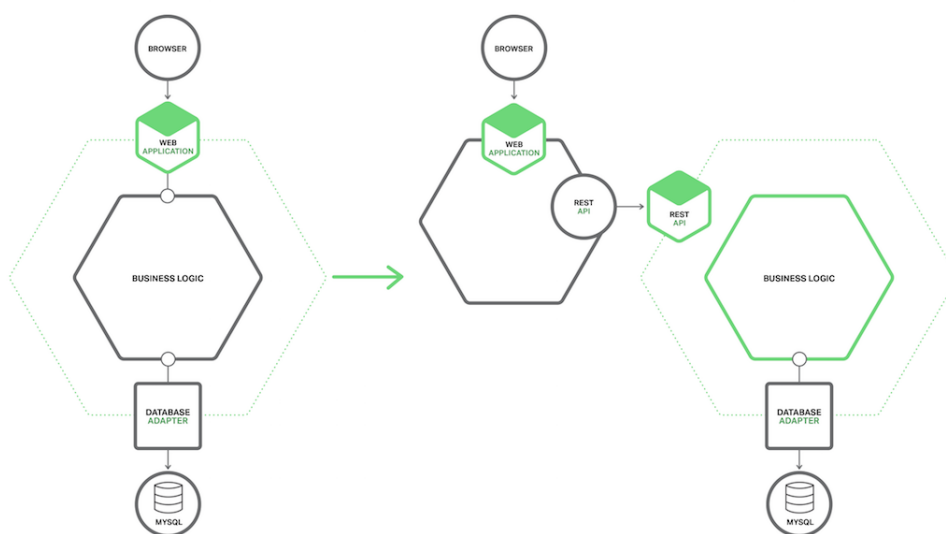


Figure 5.2: Monolithic to microservices, from nginx resources.

So, for us, each service will always be mainly the same components, communication protocol and logic layer while database layer will always be optional, a service can do some logic without need a database, imagine a simple service to do several calculations or consume the data from another service or act as third party services gateway, a social network, a payment service, whatever.

Coming back again to our problem we have decided to split all the back end in four services that will cover the domain of the problem, following a Domain Driven Development principles. Each one will be described with details after, for now, only some reasons that why is so. We need a logic part that controls any system call, some as the gateway that acts as a dispatcher of requests, that call to the service (maybe not only one) that can help it to resolve the requests and give back the info.

For another and, we need cover all requirements specified, and to achieve this we are going to design three services that split all logic issues, it will be **Teaching Database microService**, **Students Control** and **Analysis microService**. All together compose the backend of the app, together with other service that will enlarge the functionality of our project in the future.

So, as a big picture, the backend of our service would look like this, some services with approximately the same form (in the pictur only three, is the same).



Figure 5.3: Some as the first approach of the system.

Once that the architecture is clear and the services are well defined (not their behavior, only their domain) we must have clear how this services will be deployed in the platform or infrastructure chosen because the selection can modify the way to design the services.

So in our case, of all possibilities that we have already discussed before we only consider two as acceptable, Google and Amazon infrastructure, so as you know already, Google was the selected, so we need design our services to run inside of it and this mean that we need how configure our services.

As we always evaluate some options, we have done a simple design to see what will be the infrastructure and design required, first in Amazon Web Services.

In this case, as the code must run into the container we would need configure all our services as Docker Containers and to use the service Amazon EC2 Container Service to deploy them. And our gateway service could be deployed in another container or could be used a specific service called API Gateway Service with Lambda Service, both services was launched by Amazon when they saw that this pattern was really common in all projects, and they design this service to save efforts to developers (even though it have some drawbacks as for example that you do not try it in local, at least for now but is sure that will be solved).

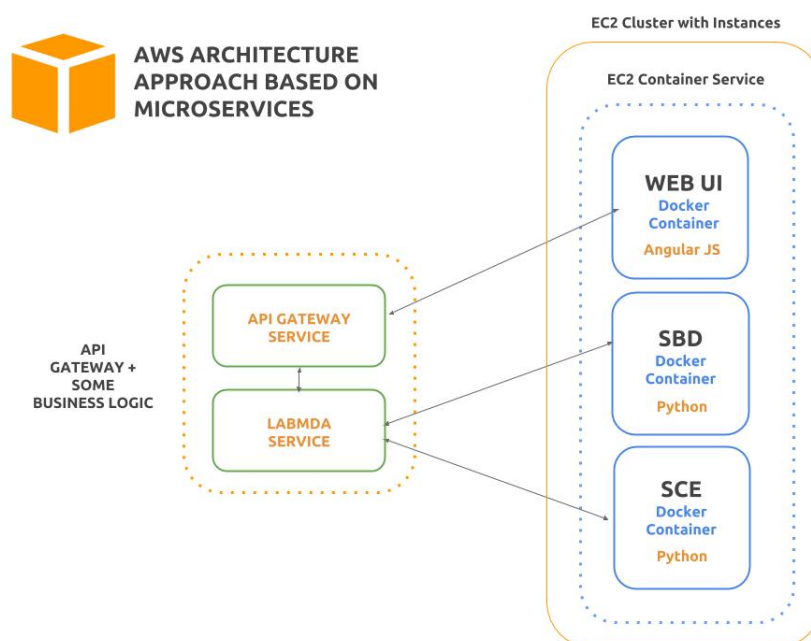


Figure 5.4: Approach in Amazon Web Service infrastructure.

The goal of this study was compare the facilities or drawbacks of each platform (doing a little bit deep study that only to take a look over the solutions each one offered), and the drawbacks of AWS for this kind of "amateur" project was too.

Come back to GAE, the deploy must be different because there are not docker container, we only have speciall sandbox to run our code with some special characteristics (as wee commented before). But, the architecture does not vary significantly between both architectures, as could not have been otherwise. If this not so, microservices and their platform independence would not have sense.

So, focused on GAE, the services must be deployed and developed (nothing else of some changes in the code) as modules (now called services by GAE) and this can (and must) put joined in GAE applications. So, as the platform are thought and to adjust this to our requirements we are going to launch two apps, one only with the user web interface service (front end) and another with the group of services that compound the backend, (here GAE app 2). You can see the distribution in the next picture.

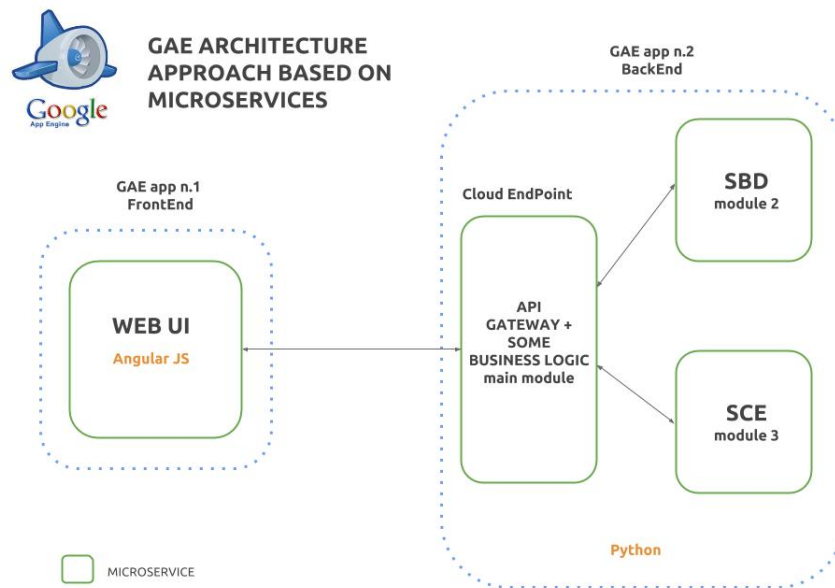


Figure 5.5: The same approach in Google App Engine infrastructure.

5.1.1 Be more realistic

As is easy to imagine, it is needed a lot of services to complete all requirements of the project, and presented here is only the most important because they define the core of the app and their structure but of course are necessary a lot more. Only thinking about security and authentication is easy to see that it is necessary other services to check this, but as is a standard service (not taking part in the design of the app) it will not be developed now and neither explained here. Their goals are to ensure the authentication of the user and provide the user authorization of different parts of the app. So, this will be **Authorization & Authentication microService**.

On another hand, there are two more functionalities that must be developed as soon as possible in the next iterations, a files storage, and a messaging service. Both are necessary for a more advanced stage of the deploy, not now, but essential in the final product. The next picture shows you all services, developed in this phase and the planned.

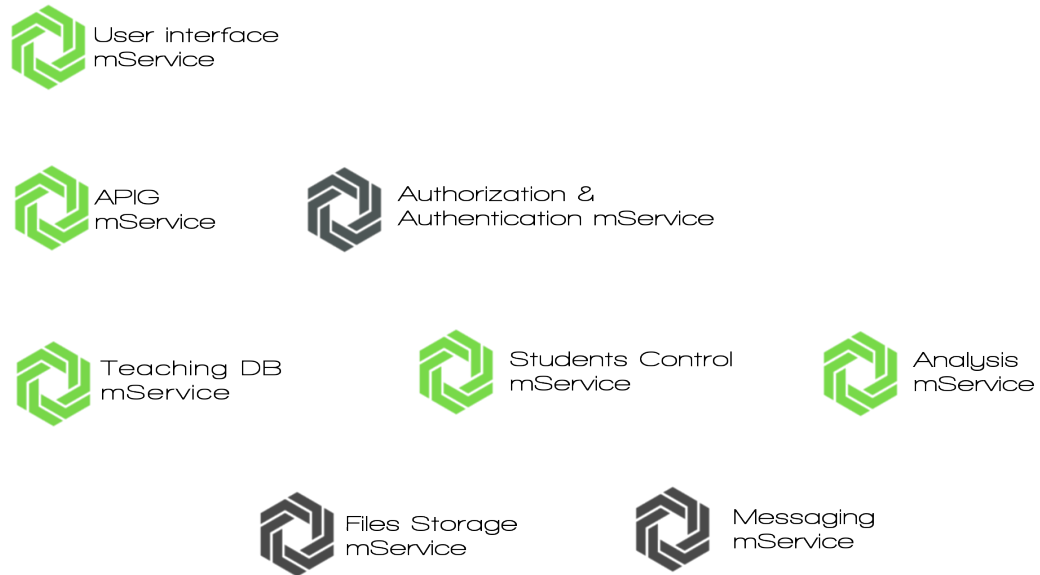


Figure 5.6: Group of microservices of SMS launched and planned.

Next picture shows us a more realistic snap of the deployment, with the services, technologies, tools and third party services consumed. We can see how SCmS need G Cloud Datastore, TDBmS Cloud SQL (for example) and all technologies related.

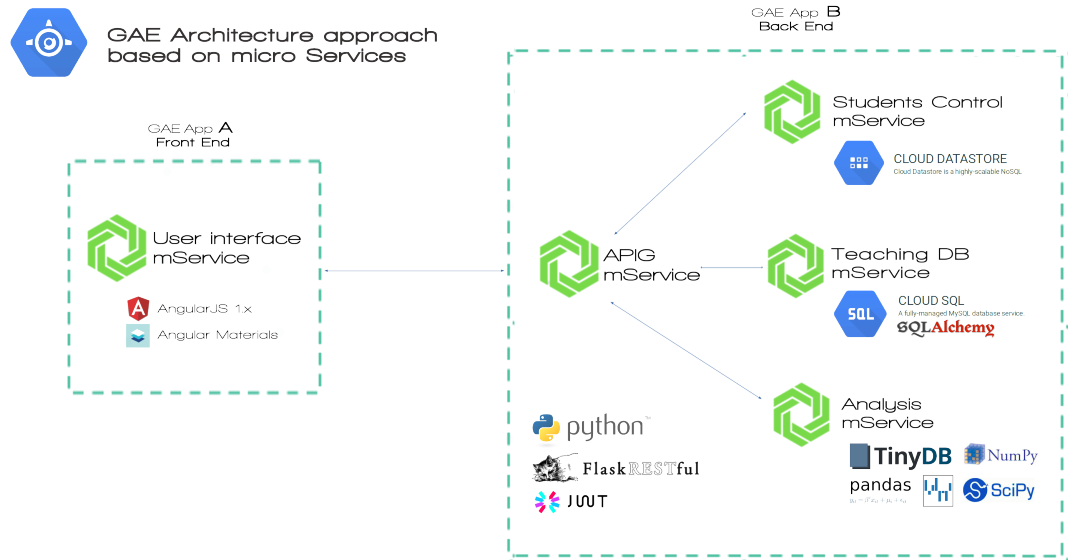


Figure 5.7: Final snap of the distribution of the services with their technologies.

And will be our roadmap to follow, without any service needed to obtain a pre-alfa product version that satisfy minimum requirements at this point.

5.2 API standar status codes

When we design an API over HTTP we are assuming that we are going to use standard status code of this protocol (HTTP/1.1 in particular) and is useful that from beginning all team knows what code will be used and the meaning. The next table shows us which is these, obviously, there are a lot of more, but this is most important for us.

Status Code	Meaning	Use/Behavior
200	Ok	Any request that returned content.
201	Created.	Not return content and the item has been saved.
204	Ok without body.	Call with success without content.
404	Not found.	Element not founded.
422	Unprocesable entity.	Bad body at request.
500	Server error.	Any unknown kind of server error.

Table 5.1: HTTP1.1 Status Code common used

And as is known, the numbers have a specific meaning and it will be useful to know just in case another will appear.

Class	Meaning
2xx	Successful
3xx	Redirection
4xx	Client Error
5xx	Server Error

Table 5.2: HTTP1.1 Status Code first digit meaning

5.3 Pesistence Strategy

When we are talking about persistency that means how the data that change their state in the database is managed. Obviously we are talking of the changes between created, deleted and the between states.

It has been identified three ways to do this:

- **Non persistence**

When an item is updated or deleted any information of the last state remains.

- **Half persistence**

When a data is logically erased, the data remains in the database but will never be accessible by the user. In this case it is useful if we want develop some mechanism to redo actions, in this case only deletion actions.

- **Complete persistence**

When not only the elements erased can be restored, all items can be restored in any point of the history of the system. This is the more complex mechanism to persistence but also the most powerful and useful for the users, because as in a simple text editor allow do an infinite redo over actions realized.

After study all options have been decided use the second approach, and to achieve this is necessary get some kind of metadata plugged to our objects stored in databases to control their state in all moment. To do this has been designed this metadata pattern for all objects saved in any service or database:

```

1  Metadata parameters
2  createdBy      INT,
3  createdAt      DATETIME,
4  modifiedBy     INT,
5  modifiedAt     DATETIME,
6  deletedBy      INT,
7  deletedAt      DATETIME,
8  deleted        BOOL,

```

With it we get a simple way to save logically deleted items, that allow us develop a simple way to recovery this if user want, and not only this, also a system to know who do the actions, who save, delete or update an item and also when, a lot of data very usefull in many ways.

5.4 API Gateway microService

This service has not owned API definition because at this point of development only takes the role of dispatcher (with a bit logic in some parts, but minimal) and their only goal is to copy the services API and clone it. Despite this, their task is very important in the system, talk with clients and with services and thanks to this nothing of them need to know about another.

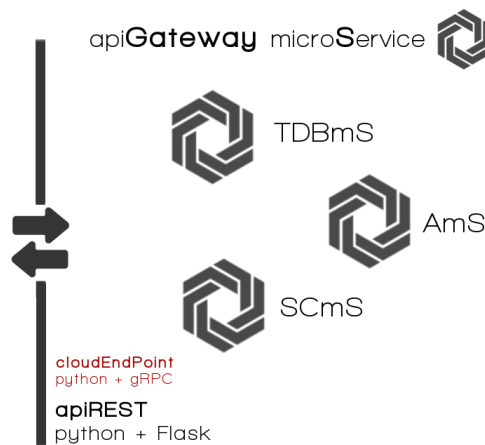


Figure 5.8: Api Gateway service concept.

As it can be seen in the figure above, at first it was designed with Google Cloud Endpoints technology, using Python and gRPC but, as we will explain in the next chapter, was not the best way (at least with the system that we can build) and finally was decided to build as the rest of them, using Flak and building a simple and well know API Rest.

5.5 Teaching Data Base microService

5.5.1 Domain and design

This mService offers the management of the teaching of the center. This means that persist in a relational database all relations between teachers, students, subjects, etc, and all resource availables to make this possible through an api.

This like the rest offers his resource through an api writed in Flask (follow the same architecture that all).

The engine to save all these relations is MySQL, for many reasons, mainly because is the best known engine and in which it has some experience and also because GCP offers as a cloud product Google Cloud SQL Databases. Until recently only offers MySQL but now (since March of 2017) they offer also PostgreSQL.

In the old version it was a little ORM that offers simple methods to access data transform this in SQL raw sentences. Now this library is only a wrapper of the SQLAlchemy to keep apart the api rest of the service to the database access layer.

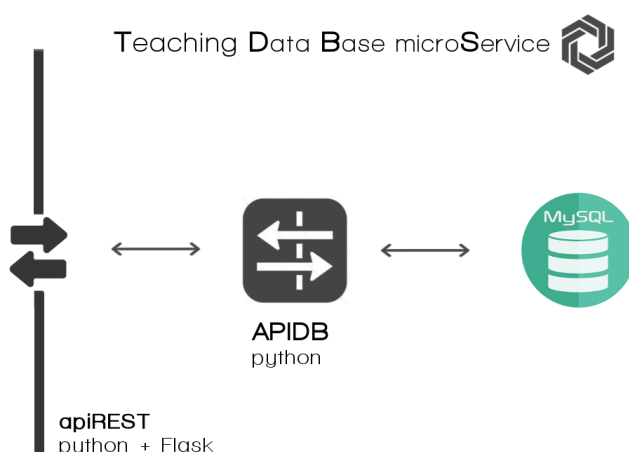


Figure 5.9: Teaching Data Base service parts.

5.5.2 Api definition

To get API definitions will be used RAML². In spite of to be more powerful, it only will be used to do a good definition of the APIs independently of the technology with after it will be developed, understandable for any developer.

In this case, the API definition of this service is like this, and as we can see, is very easy to understand, and we only are going to use the most simple version of this definition standard, there a lot of other things that we can express, but is not necessary as first approach.

500: description: Internal Server Error. When an unexpected error appears.

```

1  #%RAML 0.8
2  title: Teaching Data Base API
3  version: 1.0
4  baseUri: ---
5
6  /test:
7    description: For API testing purposes.
8    get:
9      description: Test the connection with the API.
10     responses:
11       200:
12         description: Ok. The API is running.
13       405:
14         description: Method not allowed. When it is not running.
15
16  /test_mysql:
17    description: For database testing purposes.
18    get:
19      description: Test the connection with the database through the
20        API.
21      responses:
22        200:
23          description: Ok. Database engine in running.
24
25  /entities/{kind}:
26    description: Collection of kind type management resource.
27    uriParameters:
28      kind:
29        description: Type of entity required.
30        type: string
31    get:
32      description: Return all items by kind passed.
33      responses:
34        200:
35          description: Ok. A list of item will be returned.
36        404:
37          description: Not found. The kind of item does not exists.
38    post:
39      description: To save item in the database.
40      responses:

```

²RESTful API Modeling Language (RAML) was launched at 2013 and is YAML based language for describing RESTful APIs


```

40      201:
41          description: Created. The item has been saved, no body will
              be returned.
42      400:
43          description: Bad request. Request have a syntax error.
44      404:
45          description: Not found. The kind of item does not exists.
46
47  /entities/{kind}/{entity_id}:
48      description: Item of specific kind management resource.
49      uriParameters:
50          kind:
51              description: Type of entity required.
52              type: string
53          entity_id:
54              description: Identifies of the item.
55              type: string
56      get:
57          description: Return an item.
58          responses:
59              200:
60                  description: Ok. An item will be returned.
61              404:
62                  description: Not found. The kind or item does not exists.
63      put:
64          description: To update item in the database.
65          responses:
66              204:
67                  description: No content. Updated success without return.
68              400:
69                  description: Bad requeest. Request have a syntax error.
70              404:
71                  description: Not found. The kind or item does not exists.
72      delete:
73          description: To delete an item.
74          queryParameters:
75              action:
76                  description: Use to specify some special action.
77                  Options:
78                      dd: delete all dependencies of the item
                          deleted.
79                          If you delete a student will be deleted
                          all
80                          relations with the rest of item in the
                          database
81                          of them.
82              type: string
83          responses:
84              204:
85                  description: No content. Item has been deleted.
86              404:
87                  description: Not found. The kind or item does not exists.
88
89
90  /entities/{kind}/{entity_id}/{optional_nested_kind}/{onk_entity_id}:
91      description: To delete relations between items. Only available for
              now
92
93          between kind as class of subject and
94          optional_nested_kind as student.
95      uriParameters:
96          kind:
97              description: The kind of item to manage.

```

```

96         type: string
97     entity_id:
98         description: The id of the kind.
99         type: string
100     optional_nested_kind:
101         description: The nested kind (one related with first).
102         type: string
103     onk_entity_id:
104         description: Identifier of the nested kind.
105         type: string
106     delete:
107         description: Delete the relation between items.
108         responses:
109             204:
110                 description: No content. Item has been deleted.
111             404:
112                 description: Not found. The kind or item does not exists.
113
114 /entities/{kind}/{entity_id}/{related_kind}:
115     description: Manage subcollections of items of a kind.
116     uriParameters:
117         kind:
118             description: The kind of item to manage.
119             type: string
120         entity_id:
121             description: The id of the kind.
122             type: string
123         related_kind:
124             description: The related kind to get all items.
125             type: string
126     get:
127         description: Return a list of items of a related collection of
128         another.
129         responses:
130             200:
131                 description: Ok. An item will be returned.
132             404:
133                 description: Not found. Some of ids do not exists.
134
135 /{rk_entity_id}/{subrelated_kind}:
136     description: To manage the relation nested with related kinds.
137     uriParameters:
138         rk_entity_id:
139             description: Related kind id
140             type: string
141         subrelated_kind: Subrelated kind id
142         description:
143             type: string
144     get:
145         description: To get all related items in nested related kind.
146         responses:
147             200:
148                 description: Ok. An item will be returned.
149             404:
150                 description: Not found. Some of ids do not exists.
151
152 /entities/{kind}/{entity_id}/report:
153     description: Retrieve the reports about items that have it.
154     uriParameters:
155         kind:
156             description: The kind of item to manage.
157             type: string

```

```

157     entity_id:
158         description: The id of the kind.
159         type: string
160     get:
161         description: To get a report from item.
162         responses:
163             200:
164                 description: Ok. An report will be returned.
165             404:
166                 description: Not found. Item does not exists.

```

HERE-¿ UN POCO Más de TEXTO AQUI

5.5.3 Database logical design

Based on user histories and the domain of the problem the designe done based of this entity relation diagram:

The design follow some details that the domain presents, that is related (at least the more significantly) below:

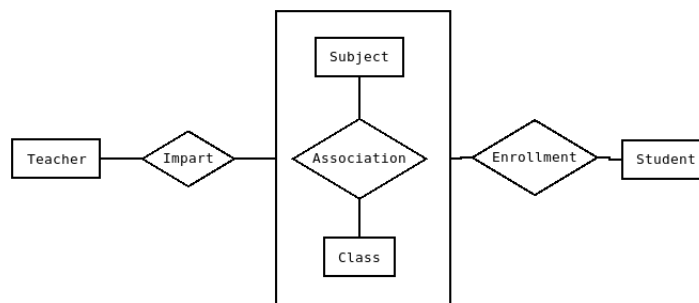


Figure 5.10: Entity-Relationship Diagram.

Data model

```

1  CREATE TABLE student (
2      studentId      INT NOT NULL AUTO_INCREMENT,
3      name            CHAR(100) NOT NULL,
4      surname         CHAR(100) NOT NULL,
5      dni             INT,
6      email           CHAR(120),
7      address         CHAR(100),
8      locality        CHAR(50),
9      province        CHAR(50),
10     birthdate        DATE,
11     phone            CHAR(50),
12     profileImageUrl  CHAR(200),
13     gender           CHAR(1),
14
15     #Metadata parameters
16     createdBy         INT,
17     createdAt         DATETIME,
18     modifiedBy        INT,

```

```

19     modifiedAt      DATETIME ,
20     deletedBy       INT,
21     deletedAt       DATETIME ,
22     deleted         BOOL,
23
24     PRIMARY KEY (studentId),
25     UNIQUE (dni, deleted)
26 );

```

```

1  CREATE TABLE teacher (
2      teacherId      INT NOT NULL AUTO_INCREMENT,
3      name            CHAR(100) NOT NULL,
4      surname         CHAR(100) NOT NULL,
5      dni             INT,
6      email           CHAR(120),
7      address         CHAR(100),
8      locality        CHAR(50),
9      province        CHAR(50),
10     birthdate       DATE,
11     phone           CHAR(50),
12     profileImageUrl CHAR(200),
13
14     #Metadata parameters
15     createdBy        INT,
16     createdAt        DATETIME,
17     modifiedBy       INT,
18     modifiedAt       DATETIME,
19     deletedBy        INT,
20     deletedAt        DATETIME,
21     deleted          BOOL,
22
23     PRIMARY KEY (teacherId),
24     UNIQUE (dni, deleted)
25 );

```

```

1  CREATE TABLE subject (
2      subjectId      INT NOT NULL AUTO_INCREMENT,
3      name            CHAR(100) NOT NULL,
4      description     CHAR(255),
5
6      #Metadata parameters
7      createdBy       INT,
8      createdAt       DATETIME,
9      modifiedBy      INT,
10     modifiedAt      DATETIME,
11     deletedBy       INT,
12     deletedAt       DATETIME,
13     deleted         BOOL,
14
15     PRIMARY KEY (subjectId),
16     UNIQUE (name, deleted)
17 );

```

```

1  CREATE TABLE class (
2      classId        INT NOT NULL AUTO_INCREMENT,
3      course         INT(1) NOT NULL,
4      word            CHAR(10) NOT NULL,
5      level           CHAR(20) NOT NULL,
6      description     CHAR(255),

```

```
7
8     #Metadata attributes
9     createdBy INT,
10    createdAt DATETIME,
11    modifiedBy INT,
12    modifiedAt DATETIME,
13    deletedBy INT,
14    deletedAt DATETIME,
15    deleted    BOOL,
16
17    PRIMARY KEY (classId),
18    UNIQUE (course, word, level, deleted)
19 );

1 CREATE TABLE association (
2     associationId INT NOT NULL AUTO_INCREMENT,
3     classId      INT,
4     subjectId    INT,
5
6     #Metadata parameters
7     createdBy    INT,
8     createdAt    DATETIME,
9     modifiedBy   INT,
10    modifiedAt    DATETIME,
11    deletedBy     INT,
12    deletedAt     DATETIME,
13    deleted       BOOL,
14
15    FOREIGN KEY (subjectId) REFERENCES subject (subjectId),
16    FOREIGN KEY (classId) REFERENCES class (classId),
17    PRIMARY KEY (associationId),
18    UNIQUE (subjectId, classId, deleted)
19 );

1 CREATE TABLE impart (
2     impartId      INT NOT NULL AUTO_INCREMENT,
3     associationId INT,
4     teacherId     INT,
5
6     #Metadata parameters
7     createdBy    INT,
8     createdAt    DATETIME,
9     modifiedBy   INT,
10    modifiedAt    DATETIME,
11    deletedBy     INT,
12    deletedAt     DATETIME,
13    deleted       BOOL,
14
15    FOREIGN KEY (associationId) REFERENCES association (associationId),
16    FOREIGN KEY (teacherId) REFERENCES teacher (teacherId),
17    PRIMARY KEY (impartId),
18    UNIQUE (associationId, teacherId, deleted)
19 );

1 CREATE TABLE enrollment (
2     enrollmentId INT NOT NULL AUTO_INCREMENT,
3     studentId    INT,
4     associationId INT,
5
```

```

6  #Metadata parameters
7  createdBy      INT,
8  createdAt      DATETIME,
9  modifiedBy     INT,
10 modifiedAt     DATETIME,
11 deletedBy      INT,
12 deletedAt      DATETIME,
13 deleted        BOOL,
14
15 FOREIGN KEY (associationId) REFERENCES association (associationId),
16 FOREIGN KEY (studentId) REFERENCES student (studentId),
17 PRIMARY KEY (enrollmentId),
18 UNIQUE (studentId, associationId, deleted)
19 );

```

5.5.4 Way to access to raw data

While at first of develop the main strategy to follow was write all by hand, finally the point of view has been changed to follow the use of standard tools and avoid reinvent the wheel.

So, if in the first stages of the project the access to raw data through the engine was hand made, using an own simple library that worked like as a simple ORM³, the evolution of it and especially the problems found and the unmaintainability of code have made that now the approach turned to use a good tool as ORM like SQLAlchemy⁴.

The changes in the specifications of the api while the develop and the maintenance of the performance of the queries when is written hand made in raw SQL isn't a good idea. Before the develop of this mService it was easy to understand that only in few projects is justified the use of raw sql sentences and drivers without ORM (by easy that it was).

³Object-relational mapping or **ORM** is a technique for converting data between incompatible type systems using oriented objects programming to do this, as for example access to database systems

⁴

5.6 Students Control microService

5.6.1 Domain and design

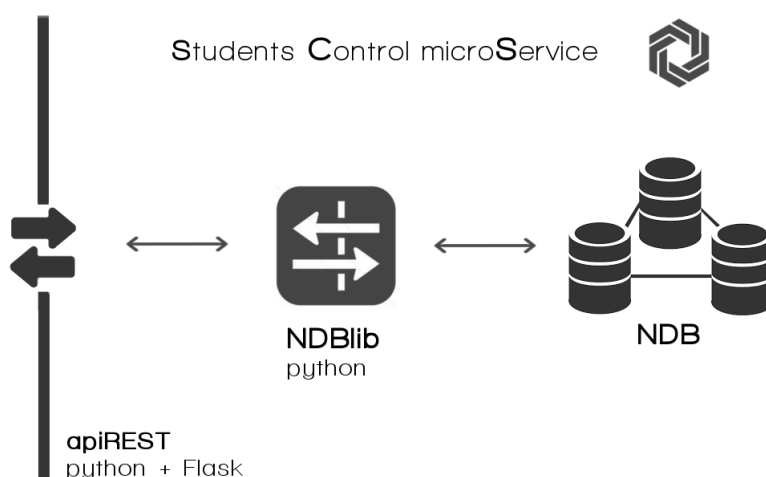


Figure 5.11: Students Control service parts.

5.6.2 Api definition

How was commented before, the design of the resources is based on getting an user interface with minimal requirements. So, there are a lot of ways to get a collection of data to present it on the interface, and we must decide who do the more complex part of this compose, and obviously, we want to save time to the user device. To better understand it, is best an example. If we want to get a list of attendance controls that have been done and if we only request the raw list we only will obtain a list with all data but without any reference to the data of teacher or data of the subject.

So, to can offer by the user interface all data that it need, is necessary to develop the fill of this data before to send. And all this complexity must be a backend issue, indecently of the request or the mechanism that this use. We can see an example of this in *ac resource* when is called with *GET* method.

Attendance Controls API Subsection

The sub API which we can manage all related with attendance controls, that are records about the attendance of the students to class and something as if the students attends with delay, if this is justified, if gone with or

without the uniform, etc. As with the previous, this is the behavior designed RAML based.

```

1
2  #%RAML 0.8
3  title: Attendance Controls API
4  version: 1.0
5  baseUri: ---
6  /ac:
7    description: The resource to work with attendance controls
      database saved.
8    get:
9      description: Get a list of all attendance controls
10     responses:
11       200:
12         description: Ok. A list of items will be returned.
13         body:
14           application/json:
15             example: |
16               [
17                 {
18                   "acId": 4785074604081152,
19                   "association": {
20                     "associationId": 13,
21                     "class": {
22                       "classId": 1,
23                       "course": 1,
24                       "level": "ESO",
25                       "word": "A"
26                     },
27                     "subject": {
28                       "name": "Science",
29                       "subjectId": 1
30                     }
31                   },
32                   "createdAt": "2017-03-12T13:20:23.906080",
33                   "createdBy": 1,
34                   "students": 2,
35                   "teacher": {
36                     "name": "Jhoan",
37                     "profileImageUrl": "/imageservice/29372929.jpg",
38                     "surname": "Mathew",
39                     "teacherId": 1
40                   }
41                 }
42               ]
43
44     post:
45       description: To save an item of "ac" kind into service.
46       responses:
47         201:
48           description: Created without response. The item was added to
            database will not
            returned body.
49       body:
50         application/json:
51           example: |
52             {
53               "students": [
54                 {
55                   "control": {

```



```
57         "delay": 0,
58         "assistance": true,
59         "uniform": true,
60         "justifiedDelay": true
61     },
62     "studentId": 113
63 },
64 {
65     "control": {
66         "delay": 0,
67         "assistance": true,
68         "uniform": true,
69         "justifiedDelay": true
70     },
71     "studentId": 213
72 }
73 ],
74 "teacherId": 23,
75 "association": {
76     "associationId": 13,
77     "classId": 24,
78     "subjectId": 17
79 }
80 }
81
82 400:
83     description: Bad request. Request have a syntax error.
84 404:
85     description: Not found. The kind of item does not exists.
86
87 /{ac_id}:
88     uriParameters:
89         ac_id:
90             description: Attendance control identifier
91             type: string
92     get:
93         description: Return an item of the attendance controls
94             collection.
95         responses:
96             200:
97                 description: Ok. An item will be returned.
98             404:
99                 description: Not found. The item required does not exists.
100     put:
101         description: Save a new attendance control in the database.
102         responses:
103             204:
104                 description: No content. The server has fulfilled the
105                     request but does not need to return an entity-body.
106             404:
107                 description: Not found. The item required does not exists.
108     delete:
109         description: Resource to delete items of this type.
110         responses:
111             204:
112                 description: No content. The server has fulfilled the request but
113                     does not need to return an entity-body.
114             404:
115                 description: Not found. The item required does not exists.
116
117 /schema:
118     description: To know the schema of this kind of object.
```



```
39         ]
40
41     post:
42         description:
43         responses:
44             201:
45                 description: Created without response. The item was added to
46                             database will not
47                             returned body.
48             body:
49                 application/json:
50                 example: |
51                 {
52                     "studentId": 5,
53                     "teacherId": 42,
54                     "classId": 3,
55                     "subjectId": 21,
56                     "dateTime": "2000-12-03 10:30",
57                     "kind": 1,
58                     "gravity": 5,
59                     "description": "A little problem with new boy."
60                 }
61             422:
62                 description: Unprocessable Entity. Probably because of the
63                             payload
64                             sendes has not correct format.
65
66 /{dn_id}:
67     uriParameters:
68         dn_id:
69             displayName: Disciplinay Note ID
70             type: integer
71     get:
72         description: Return an item of the disciplinary notes
73                     collection.
74         responses:
75             200:
76                 description: Ok. Successful requests. An item will be
77                             returned.
78             404:
79                 description: Not found. The item required does not exists.
80     put:
81         description: Save a new disciplinary note in the database.
82         responses:
83             204: No content. The server has fulfilled the request but
84                 does not need to return an entity-body.
85             404:
86                 description: Not found. The item required does not exists.
87     delete:
88         description:
89         responses:
90             204: No content. The server has fulfilled the request but
91                 does not need to return an entity-body.
92             404:
93                 description: Not found. The item required does not exists.
94
95 /schema:
96     get:
97         description:
98         responses:
99             200:
```

```

95      description: Ok. The item schema will be returned.
96      body:
97          application/json:
98              example: |
99                  {
100                      "gravities": [
101                          {"code": 1,"meaning": "mild"},
102                          {"code": 2,"meaning": "low"},
103                          {"code": 3,"meaning": "medium"},
104                          {"code": 4,"meaning": "high"}
105                      ],
106                      "kinds": [
107                          {"code": 1, "meaning": "Bullying"},
108                          {"code": 2, "meaning": "Disrespect"},
109                          {"code": 2,"meaning": "Gender violence"}
110                      ]
111                  }
112
113      404:
114          description: Not found. The item required does not exists.

```

Maks API Subsection

As the final part of the simple student control that we do we still have the marks section. With this we would record all exams marks that students achieve, that is another big part that was required. As with the previous, this is the behavior designed RAML based.

```

1
2  #%RAML 0.8
3  title: Marks Subserction API
4  version: 1.0
5  baseUri: ---
6  /mark:
7    description: The resource to work with marks database saved-
8    get:
9      description: Get a list of all marks
10     responses:
11       200:
12         description: Ok. A list of items will be returned.
13         body:
14           application/json:
15             example: |
16               [
17                 {
18                   "createdAt": "2017-03-12T23:12:53.603021",
19                   "createdBy": 1,
20                   "enrollment": {
21                     "classId": 4,
22                     "enrollmentId": 42,
23                     "subjectId": 5,
24                     "teacherId": 8
25                   },
26                   "markId": 6016527627190272,
27                   "marks": {
28                     "final": null,
29                     "firstEv": 5,
30                     "preFirstEv": 3,
31                     "preSecondEv": 8,
32                     "secondEv": 9.

```

```
33         "thirdEv": 10
34     },
35     "studentId": 5
36 }
37 ]
38
39 post:
40   description:
41   responses:
42     201:
43       description: Created without response. The item was added to
44         database will not
45         returned body.
46     body:
47       application/json:
48         example: |
49           {
50             "studentId": 5,
51             "enrollment": {
52               "enrollmentId": 42,
53               "classId": 4,
54               "subjectId": 5,
55               "teacherId": 8
56             },
57             "marks": {
58               "preFirstEv": 3,
59               "firstEv": 5,
60               "preSecondEv": 8,
61               "secondEv": 9,
62               "thirdEv": 10,
63               "final": 9
64             }
65           }
66     422:
67       description: Unprocessable Entity. Probably because of the
68         payload
69         sendes has not correct format.
70
71 /{mark_id}:
72   uriParameters:
73     mark_id:
74       displayName: Mark ID
75       type: integer
76   get:
77     description: Return an item of the marks collection.
78     responses:
79       200:
80         description: Ok. Successful requests. An item will be
81         returned.
82       404:
83         description: Not found. The item required does not exists.
84   put:
85     description: Save a new mark in the database.
86     responses:
87       204: No content. The server has fulfilled the request but
88         does not need to return an entity-body.
89       404:
90         description: Not found. The item required does not exists.
91   delete:
92     description:
93     responses:
```

```

91      204: No content. The server has fulfilled the request but
          does not need to return an entity-body.
92      404:
93          description: Not found. The item required does not exists.
94
95      /schema:
96          get:
97              description:
98              responses:
99                  200:
100                     description: Ok. Successful requests. An item will be
                          returned.
101                  404:
102                     description: Not found. The item required does not exists.

```

5.7 User Interface microService

5.7.1 Domain and design

As has been mentioned, we are going to use AngularJS to build the user interface. This means that the files of the app would be load from a service to user devices and from there it will make all requests to the server, so the single purpose of the service is to serve all app files. The design follows bellow schema.

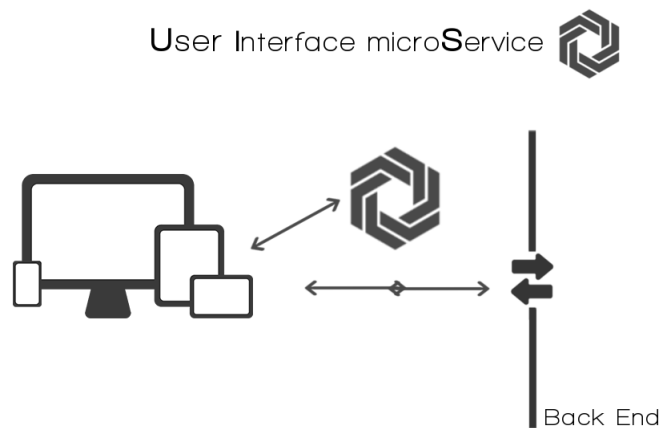


Figure 5.12: User interface service logic.

About design

All the design of the app is based of Java Script files. And the structure will be this, a section o Students Control where will placed all sections related, as Attendance Controls, Discipline, and Marks section and another section, the biggest, with all related with the teaching as can be all sections that the domain has, teachers, students, subjects, classes, and the rest of

items related.

The most of the parts need the same javascript files, controllers to define the logic and services to interact with the server. Moreover as the idea is work as each logical part as independly modules they will be built as modules as well.

This is the wireframes that give an approach of the design that we want for our interface. There are a lot of parts, sections and detail that will filled houndred of pages, so this is some examples of the simple a common parte as work with list, in this case of teachers.

About distribution

The design of the components of the interface will follow Google Material Design Philosophy⁵, it is a design language developed by Google.

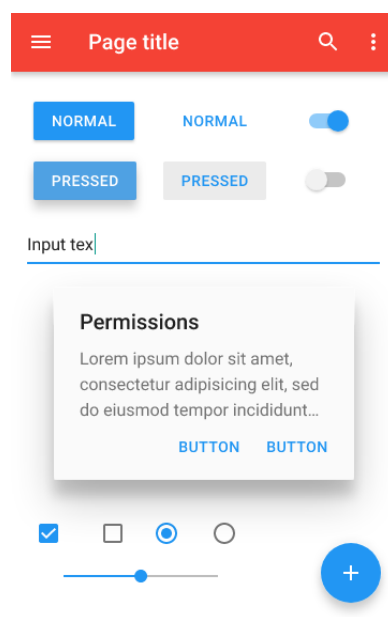


Figure 5.13: Some typical Material Design UI components.

To obtain this characteristic design there are a lot of options, but easiest is to use some CSS Framework existing instead of try to implement our own css components. There are some options availables, as *Materialize*, *MUI*, *Material UI*, *Polymer* (that is much moret), we could talk about it hours, and *Angular Material*, that is which we are selected. Why? because is not only a css kit, is a group of Angular directives that allow us to use their css components as directives, some very powerfull and simple, that increase the

⁵Google announced Material Design on June, 2014, at Google I/O conference

speed of develop when their behaviour is well understood.

Indenpendly of the style guide that we use we need design the interface, their parts, the position of the elements and how interact this among them and amoung several screens of our app. To do this design most common way is use a system of wireframing, that is as a way to compound user interfaces using very abstract form, puting the focus in the position and general design instead of the details, colors, or somethings like this. There are some tools to do this, all do basically the same, and we are going to use Pencil (An open-source GUI prototyping tool that's available for all platforms).

We would need a dozen of pages to show all sections of app. Bellow it can see some example of design with two samples of teachers section, that it will follow in the rest of the design, that possibly will suffer some change in the way.

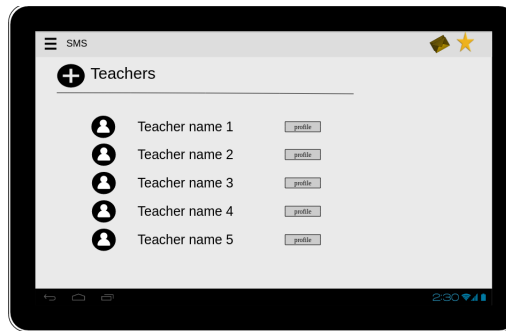


Figure 5.14: Teachers List view wireframed.

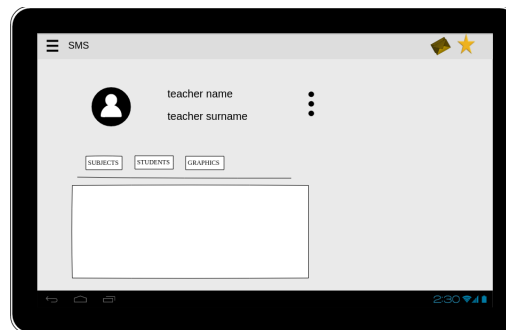


Figure 5.15: Teacher profile view wireframed.

The rest of them have been handmade (on paper) and it was believed unnecessary to include all digital wireframes here.

5.8 Analysis microService

The goal of this service is easy, we need a centered way to can retrieve reports of all of our relevant kind of items in the app, as courses, classes, teachers and mostespecially, of students. If we think only in simple reports maybe is not necessarily an independent service that offers that (only this is enough interesting the services to manage the relations keep their simplicity) but if we think in report that included statistics and data analysis as regresion, trends and alert detection in students behavior the scenario changes completely.

To do this we need to use some specific tools, libraries, and techniques which have little to do with the domains of the rest of services. Moreover of this, to achieve some of the conclusion that this service will obtain will be necessary cross data between services, so externalize this in another actor in the system is more reasonable.

5.8.1 Domain and design

Logically, this service follow the same design of the rest. A library that wraps the complexity of the logic is accessed by a standard API (as the rest as well). The main difference is that the database used in this case is TinyDB⁶, a lightweight document oriented database, written in pure Python (without external dependencies).

This was chosen for a simple reason, we need some storage system to save the data calculated to avoid do them again for each request, and as is a service that not store a big amount of data is a good option and really easy to implement. The picture below show the logic of the service.

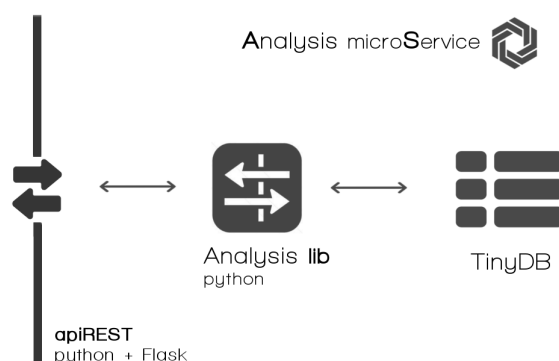


Figure 5.16: Teacher profile view wireframed.

⁶More info about the system here: <https://pypi.python.org/pypi/tinydb>

Our goal with this service is can get obtain more complex data that the services can offer us to can offer to the user interface good data blocks to build powerful graphics. As in the domain that we are the students have been evaluated by competencies in addition to particular subject skills and knowledge the app must offer some way to can check these competencies at general scope (independently of the subject) and this kind of information is perfect to use (among others) *polar charts as these*:

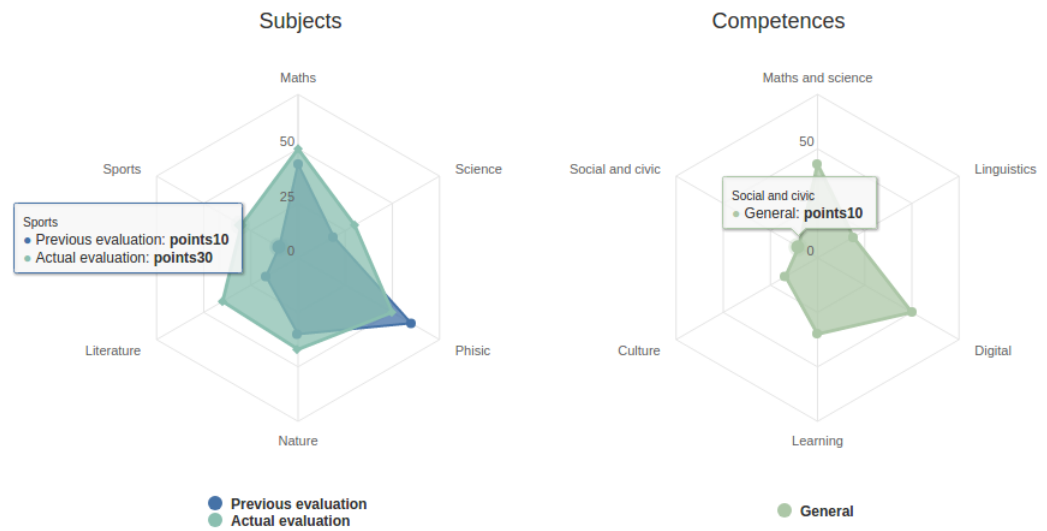


Figure 5.17: Subjects and competences polar charts.

In these, we can see as a specific student has their competencies covered and what it need to improve and the same with the subjects, not only in this course or semester, also comparing with previous of futures using simple regression equations. We can imagine the power of this kind of shots of data by the educators of the center, that would require a lot of work if they would do handmade.

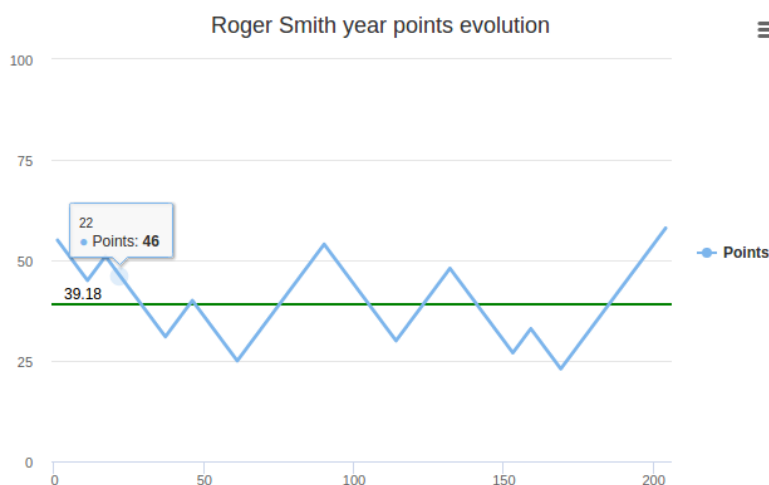


Figure 5.18: Subjects and competences polar charts.

Not is the goal review all possibilities of data flow that it can be exploited with all items and their relations, but the above picture shows another interesting bunch of data well presented to be analyzed by an educator. As we save all things that a student do in the center, homework, marks, behavior, delays, etc. We can build a points system that in a range of 0 to 100, we can measure the efficiency of the student per week. That means that if a student does all their homework, all their exams well, without any delay, etc. It will be 100 points in this week and the opposite if it does not do anything.

It is mighty because in a look we can know how the student is, how is their evolution and the best interesting (using simple regression) which is their trend, to be able to make decisions proactively, not when will be too late, improving drastically the way to understand the student tracking systems. **And it is only the begin.**

5.8.2 API Definition

In this case, the API is very small (not meaning nothing) because is not few relations between items inside, we only need a way to request reports to the system, nothing else.

```

1  #%RAML 0.8
2  title: Attendance Controls API
3  version: 1.0
4  baseUri: ---
5  ams/{kind}/{entity_id}/trend:
6    description: Simple resource to get general reports.
7    uriParameters:
8      kind:
9        description: Item type related with the reports.
10       type: string

```

```

11     entity_id:
12         description: Item identifier.
13         type: string
14     get:
15         description: Return the data block with the report of the item.
16         responses:
17             200:
18                 description: Ok. A report will be returned.
19             404:
20                 description: Not found. The item or type required does not
21                     exists.
22     ams/{kind}/{entity_id}/{report_type}:
23         description: To get other type of reports of more specific one.
24         uriParameters:
25             kind:
26                 description: Item type related with the reports.
27                 type: string
28             entity_id:
29                 description: Item identifier.
30                 type: string
31             report_type:
32                 description: The kind of report required.
33                 type: string
34     get:
35         description: Return the data block with the report of the item.
36         responses:
37             200:
38                 description: Ok. A report will be returned.
39             404:
40                 description: Not found. The item or type required does not
41                     exists.

```

5.9 Auxiliar tools

5.9.1 Provisioner

A provisioner is simply a tool used to provide a system to insert test data, to avoid to do this hand-made, in our services.



Normally this work without knowledge of the internal parts of the back-end, only working with their API gateway (or API if we want only fill the database of service), understanding it as a black box. this is only an approach, we can design our provisioner to fill data directly in the databases,

which will require established a connection with them without the interaction of the services or their connection libraries. This is faster but on some occasions, we do not want to do this, because another of their goal is to check at the same time the correctly of all parts of APIs involved in the data insertion.

So, the execution this kind of program required that the system has fully launched. This will be the most used tool when we want to try some feature that required a lot of data in the system.

How does it work? Easy, only need simple parameters as entry and based on some rules it will do all calls to the API gateway to insert all data required, besides to save all operations in a register or log file because is necessary to check this in a debug operation.

5.9.2 Data base helpers and testing tools

To all those repetitive tasks as clean the database, create it, populate or something like this, reinstall the engine or purge of the system we will need some scripts to help us and will be necessary develop this.

About the testing tools, side server will be enough with the normal testing using pytest (we could talk a lot of it) and over user interface, we would be using Selenium⁷, in words of their creators, textitit is for automating web applications for testing purposes but is certainly not limited to just that.

Thinking of this pair of details, we would have covered all design decision taken, ready to start do develop, crossing the fingers to get all in time.

⁷Know more at: <http://www.seleniumhq.org/>

Chapter 6

Develop

As talk about all details that are been solved or managed would take hundred of pages in this section we only are going to revise some part of developing that are been interesting or have required some bit specific design or decision process and are interesting to explain.

6.1 Api Gateway microService

As was discussed above, the principal goal of this service is to offer the abstraction of the whole system, so it will not be very complex because all functionality is actually implemented when this service is built.

Their unique task is to receive the requests, to construe this and do the call to the service that must answer it and return the data, maybe modified. In common case and in the point of the develop are the service only reply the queries and return the responses without more logic interaction, but in the future, this is not only that this will do, because is the perfect place to implement authorization and authentication with ACLs¹ exploiting the fact that through this service all calls go.

Another situation which this service would transform the response of a service or compound a response with the response of few of them is when must be offered a resource that is the composition of data arising from several services. For example, when a user profile info is required, the call will be redirected to Teaching Data Base mService that will return a simple data block. So, this data block does not have the image of the user, because this is stored in another different service. Now the role of gateway take sense because will be it who will retrieve the image from other service, will insert it in the data block (by some way) and will return the data block complete.

¹ACL of Access Control List specifies which users or system processes are granted access to objects, as well as what operations are allowed on given objects

It also provides some possibilities to have a dynamic and not locking develop process. How? Easy, because at first the images can be extracted in execution time from some library that put example images (or something like this) without locking the APIGmS and also UImS and so, this service of data storage could be developed after, in another phase of priorities. So, one more time, the architecture help us to develop team domain based without any blocking.

6.1.1 Cloud Endpoint spike

As we are working with GAE and GCE the first approach is use their technology, and if in any resource that you can read about this they are talking about Google Cloud Enpoints must be some good reason. This technology is based of an own version of RPC protocol developed by Google and used in their own intern architecture, called gRPC.

Basically is another implementation of a Remote Procedure Call system, but according to Google, really fast and nice, and is true, but has some drawbacks as for example that can you select other common technologies, as in our case.

This is an example of the spike related, it can see in the first phases of the project.

```

1  ...
2
3  import endpoints
4  from protorpc import messages
5
6  ...
7
8  class Alumno(messages.Message):
9      nombre = messages.StringField(1)
10     apellidos = messages.StringField(2)
11     id = messages.StringField(3)
12
13 class ListaAlumnos(messages.Message):
14     alumnos = messages.MessageField(Alumno, 1, repeated=True)
15
16 ...
17
18 @endpoints.api(name='gateway', version='v1')
19 class GatewayApi(remote.Service):
20     """Gateway API v1."""
21
22     @endpoints.method(message_types.VoidMessage, ListaAlumnos,
23                       #path=nombre del recurso a llamar
24                       path='alumnos/getAlumnos', http_method='GET',
25                       #Puede que sea la forma en la que se llama desde
26                       la api:

```



```

26         #response = service.alumnos().listGreeting().
           execute()
27         name='alumnos.getAlumnos')
28
29     def getAlumnos(self, unused_request):
30         ...
31         students_list = []
32         ...
33         # Logic to get the students list from another service with maybe
           more logic.
34         ...
35         alumnosItems.append(Alumno( id=idAlumno, nombre=nombreAlumno.
           decode('utf-8'), apellidos=apellidosAlumno.decode('utf-8') ) )
36         return ListaAlumnos(alumnos=alumnosItems)
37     ...

```

This is only an example, but the complete file has about 2000 lines in the first approximation of the service. Now take a look to the *Flask* version, an example bellow, having exactly the same functionality of previous code segment.

```

1 @api_gateway.route('/students', methods=['GET'])
2 def get_students():
3     ...
4     students_list = []
5     ...
6     # Logic to get the students list from another service with maybe
       more logic.
7     ...
8     return students_list

```

The reasons to select Flask instead of it? Is not necessary define the schema of the objects, this is sufficient reason to dismiss their use, but sound enough nice to considere to another applications in the future (as almos any technology researched).

6.1.2 As a simple dispatcher

This is an example of the behavior dispatching a simple request and answering exactly the same response from the service.

```

1 @tdbms_segment_api.route('/entities/<string:kind>', methods=['POST',
   ])
2 def post_entity(kind):
3     response = requests.post(url='http://' + str(modules.
       get_hostname(module='tdbms')) + '/entities/' + str(kind),
       json=request.get_json())
4     response.headers['Access-Control-Allow-Origin'] = "*"
5     return make_response(response.content, response.status_code)
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

7     response['profile_image'] = requests.get(url='filesServicesUrl
    ...')
8     ...
9     return make_response(response, 200)

```

And the next natural step will be built a library that works as a general customer of any service in the system, to be used in any place where be needed make a call to any service. This will to work loading dynamically at execution time the api of each service to know what resources are available and turned this as Python methods.

```

1
2     @tdbms_segment_api.route('/entities/<string:entity_id>', methods=['
    GET'])
3     def get_entity(entity_id):
4         ...
5         response = service['teachingDBmS'].getEntity(entity_id)
6         ...
7         response['profile_image'] = services['filesStorage'].get(file_id
    )
8         ...
9         return make_response(response, 200)

```

With this new tool, the code of the whole project will be reduce 10% at least.

6.2 Teaching Data Base microService

6.2.1 Optional subjects, the “class” Table.

In the domain of the problem can be exists optative subjects and is needed search a way to implement this because has a specific details that aren't like the rest.

The details. The studies plan forces in certain courses to select one or several optional subjects. For example, if a student has enrollment in 2°ESO (independently of the group, A, B...) the law and consequently the studies plan force to the student to choose between some optional subjects. So, maybe this subjects exists only in this optional case as *“rare subjects”* but in other cases this are only normal subjects but that in this course are offer like optional.

A simple example of this is French subject, it in some courses like 3ESO and 4ESO is obligatory but it in Bachiller (the upper level) is optional because the students can be select if they want make the final exam with this second language or select another like English or Greek or Latin p.e.

To obtain this we decide develop a simple solution without change the original database logic schema. So, how we have an entity that save the classes

and it have three attributes, course, word and level mainly we going to add three more to this special cases, optative, groupNumber and subgroupNumber. Like this special cases haven't word param when they have value word don't have and when the item have word (A, B, C...) then don't have this special attributes.

Maybe this don't be the best solution, but is a simple in the point of develop. Obviously like we can't have two autoincrement values in the same table definition in MySQL we will need control this programatically, but is something that we can assume to get our goal easily. But we found a problem.

The same advantage that offer *UNIQUE* to deleted cases now is problematic here. While there works fine because this clause does not include to items that have fields to null and allow to exists without conflict in this case if we saved a optative group obviously would to be "*WORD*" field to null, and if we do this we can have two groups exactly equals, for example:

```
1 1 <null> ESO 1 1 1 0
2 1 <null> ESO 1 1 1 0
```

This could happens without conflict, and this should be impossible. For this reason we decide to use the same field *WORD*, with a special naming, because this never will be used by general groups, that help us to specify that the group at issue is an optative and also specify the group and subgroup.

This is something like this: *OPT_n_m* Where *n* will be the group number (must be increased handmade) and *m* the number of the subgroup (also increased handmade). Obviously this is only a simple approach to a first solution, will be improved in next iteration of the service, and we always follow the same philosophy, we prefer explicit way to do something as implicit, more understandable by any new developer.

This way to solve this problem with MySQL engine presents some problems. As all is managed by mysql but there are not way to do this automatically with the own mechanism of mysql is necessary to create some dispatcher to the insertion and the deletion to maintain the consistency. When a new optative group is introduced must be checked that exist the previous (group and subgroup, can not exists group 4 if 3 does not exists), and the same way should not be possible delete the group 4 if the number 3 exists yet.

Note that is important understand that a lot of detail of consistency control can be relegated to UI, because although there is not allowed by the logic of interaction from the API must not be allowed neither, to prevent inconsistencies malicious induced.

Because of this not only the security, also the consistency must be ensured in any action that user can do.

6.3 Students Control microService

6.4 Analysis Control microService

6.5 User Interface microService

6.5.1 Save processes flows

The logical process of saving data is apparently easy but it hides some details when we talk about update existing data. These and how to solve the problem that it presents will define how the user will use our interface. And spite of this is a design phase issue is in the develop moment when this appeared and is why it is explained here.

Focused on the problem, we have an object load in the interface, as a complete profile info of a student, and we want to update their data (change or add some new data) and this is not trivial because will define the signature between the API of service and the user interface. So, basically, we have found three ways to do this so we are going to analysis each one to choose the best, even though there are combinations. When the object is loaded in our interface we modify some field and push the save button if all the form requirements are satisfied then the complete object is sent without any else requirements.

1. Common save button

The object is sent always, independently if it really has suffered changes. Save button is always available while the requirements of the form are satisfied.

Disadvantages: The object is always sent, even if is not necessary because it has not suffered changes, hence a lot of bytes will be sent without sense.

Advantages: Is the simplest and faster implementation for the user interface and the server do not need to check anything only override the object updating their metadata.

2. Automatically saved

The object is sent each time as it suffered any change, save button not exists. Their behavior is similar to an online text editor where all changes are saved implicitly.

Disadvantages: The calls needed are huge if the user to do a lot of changes. Is necessary another parallel mechanism to maintain the possibility to do undo because if all changes are saving at the moment in the server there are not any way to do a simply undo action. A number of calls needed are huge.

Advantages: Is really futurist because the user does not to need to push any button after of update their data. All forms become more clear.

3. Totally checked

Is a more efficient way that the rest. In this case, when the object is loaded in the interface a copy is done and saved also, always, independently of the interaction of the user. With this, we achieve to get a copy of the object without any modification when the user is updating some field.

So, each way that user change some field of the form the logic check if there are any difference between both objects (original copy and the modified) and only when this difference exists and the requirements of the form are satisfied the save button will change to enable. This way if the user cancels the update of the object or after of thousand modifications leave the object exactly with the same data the send button will not enable because there are not any to update in the server.

Disadvantages: The computing requirements in the user device it will high because it will be needed hundred of checks in a simple interaction.

Advantages: The server is not involved until the object is really updated. It supposes a user experience that gives to interface more intelligent aspect.

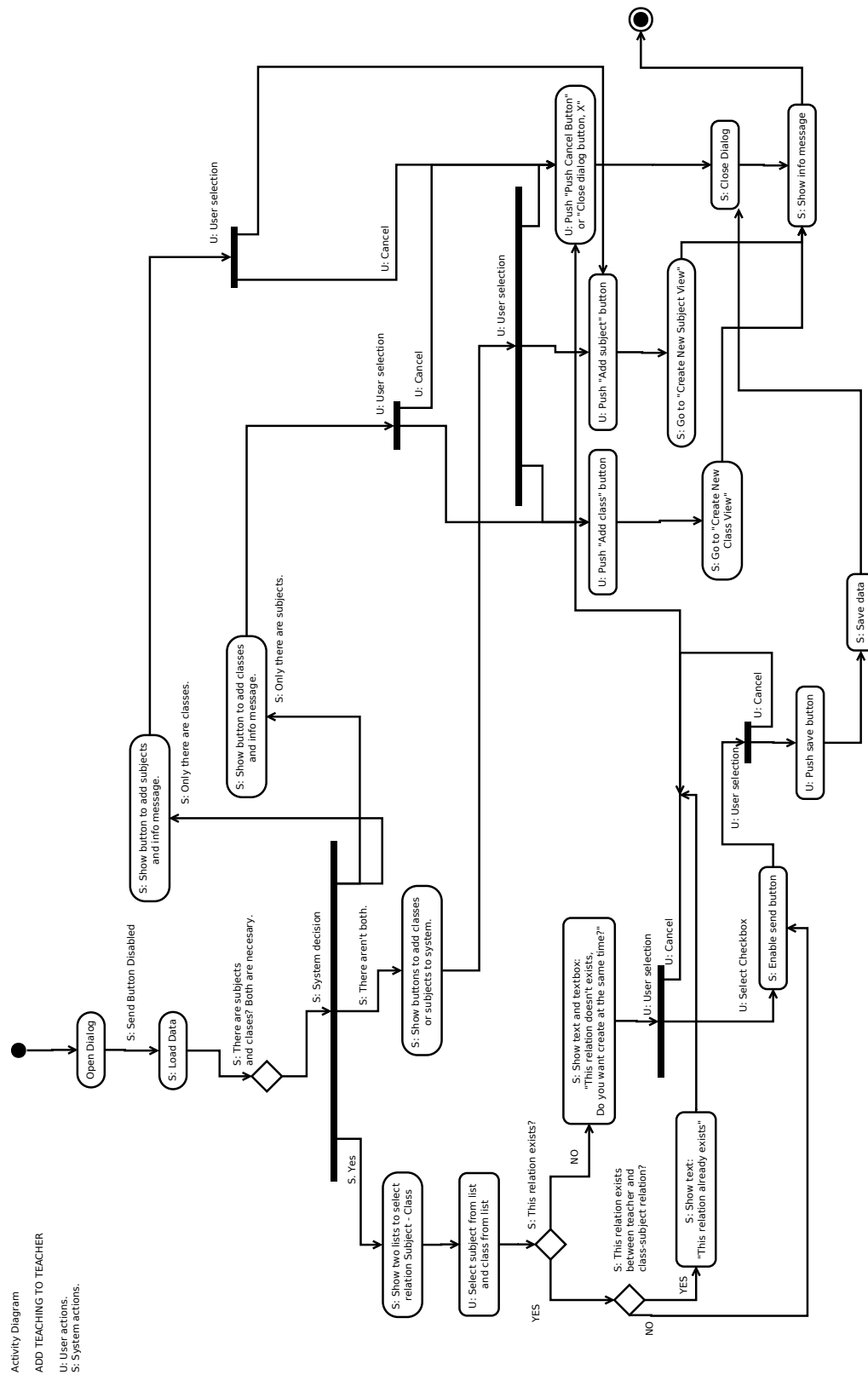
As is easy to imagine the option chosen is the third because the reason explained above. Down below is shown a piece of code which we have developed the feature of object copy. When we can to see how is used a variable to save the state of the object, after the object is saved and watches is enabled with it to detect any possible change that will modify the state of save button, all after of the student object has been returned.

studentProfile.js

```
1 ...  
2 function loadData() {  
3     vm.student = StudentsService.get({id: vm.studentId}, function () {  
4         ...  
5         vm.studentOriginalCopy = angular.copy(vm.student);  
6         $scope.studentModelHasChanged = false;
```

```
7         $scope.$watch('vm.student', function (newValue, oldValue) {
8             $scope.studentModelHasChanged = !angular.equals(vm.student
9                 , vm.studentOriginalCopy);
10             if ($scope.studentModelHasChanged)
11                 vm.updateButtonEnable = true;
12             else
13                 vm.updateButtonEnable = false;
14         }, true);
15     }, function () {
16         console.log('Student not found')
17         ...
18     });
19 }
20 ...
```

So, this behaviour is replied in all places where this kind of interaction appears.



Teachers list

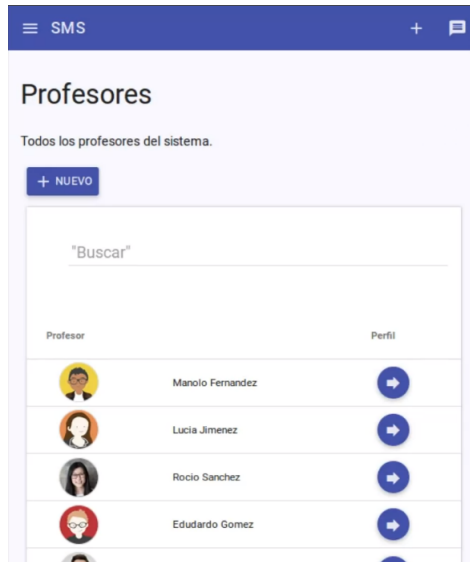


Figure 6.1: A figure

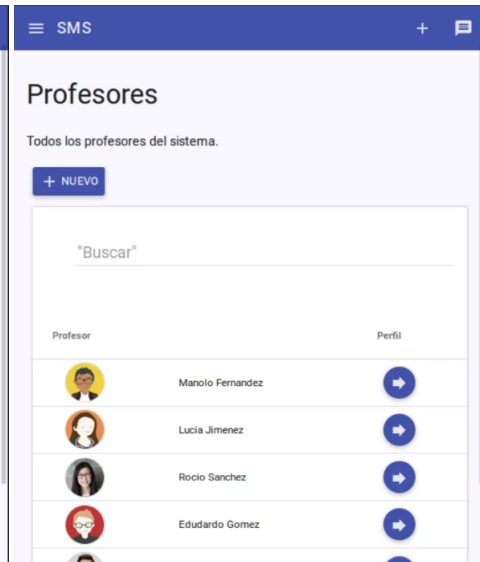


Figure 6.2: Another figure

Teacher profile

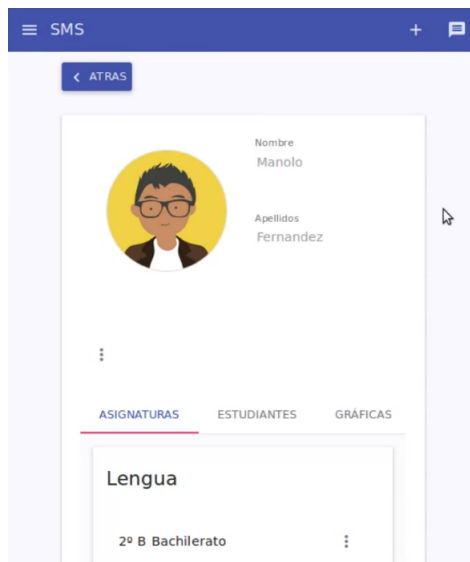


Figure 6.3: A figure

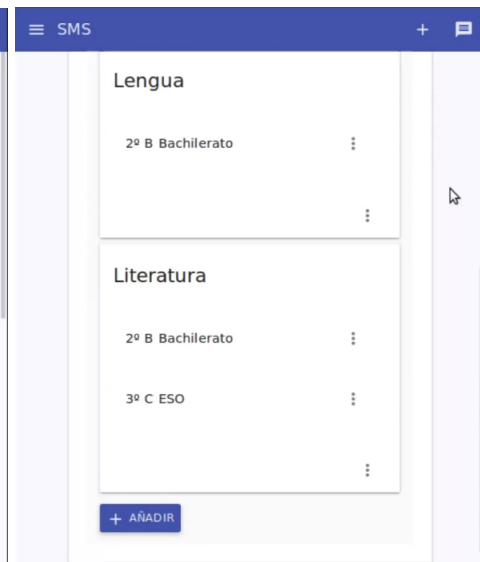


Figure 6.4: Another figure

Updating a teacher profile (note that this is the spanish version).

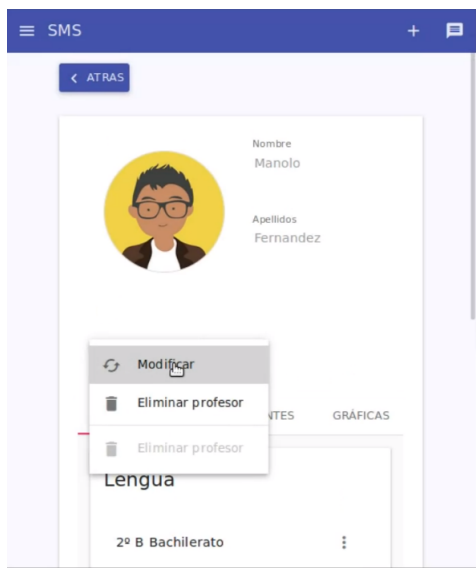


Figure 6.5: A figure

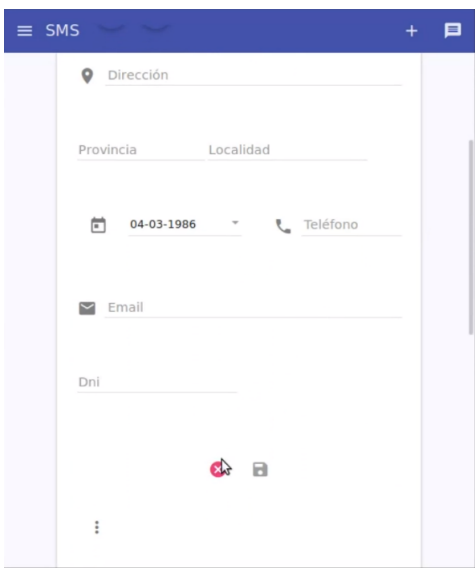


Figure 6.6: Another figure

Students lists and graphics.

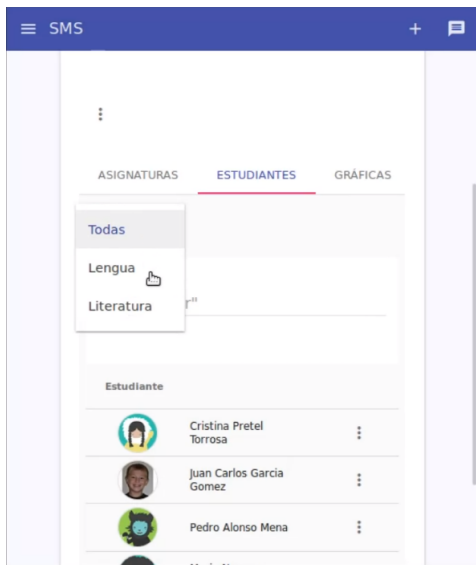


Figure 6.7: A figure



Figure 6.8: Another figure

6.6 Analysis microService

PONER EJEMPLOS DE COMO LAS ESPECIFICACIONES DE RUTAS EN LA PARTE DE DISEÑO SE CORRESPONDEN CON LA IMPLEMENTACIÓN EN FLASK

Chapter 7

Conclusions

Only a few months might not sound much but is enough to obtain some conclusion about technologies, patterns and ways to develop.

This work can not finish without a reflexion about what have been the mainly drawbacks and locks in the develop, design and evolution of the entire project, trying to propose solutions and overall, learning a lot about all.

7.0.1 Scheduling

The scheduling of development is very easy to do, in the hypothetical case that all goes fine, but as is normal, there are a lot of factors hard to control that besides are unknown at the beginning.

If we join this with an inexperienced scrum master and technologies absolutely new for the team (without a good spikes issues to select and try these) the result can be catastrophic. Despite this, the result is not as wrong as could be, and with all predictable fails, it has not been a bad finish.

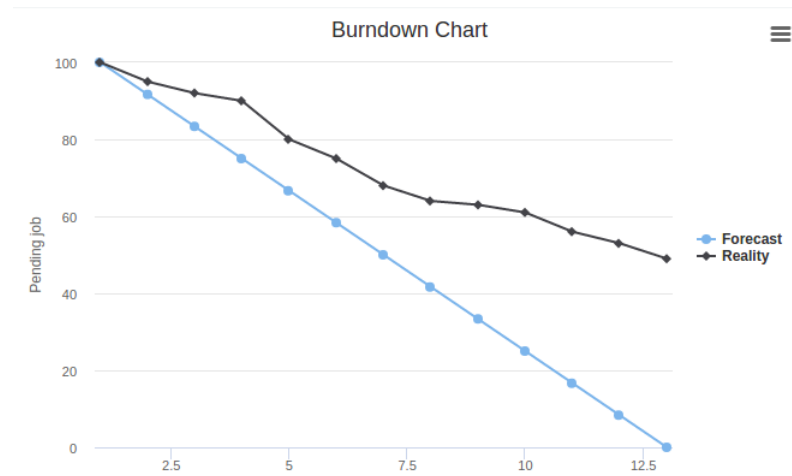


Figure 7.1: Students Control service parts.

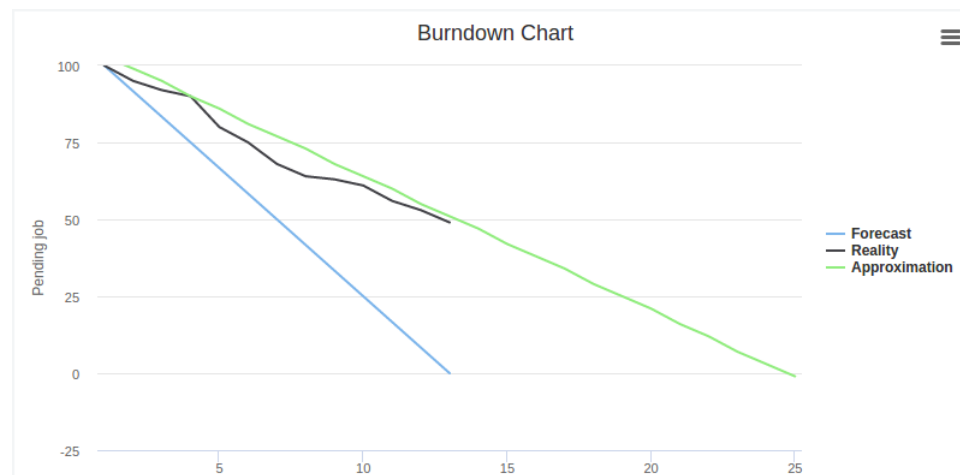


Figure 7.2: Students Control service parts.

7.0.2 Technologies and frameworks

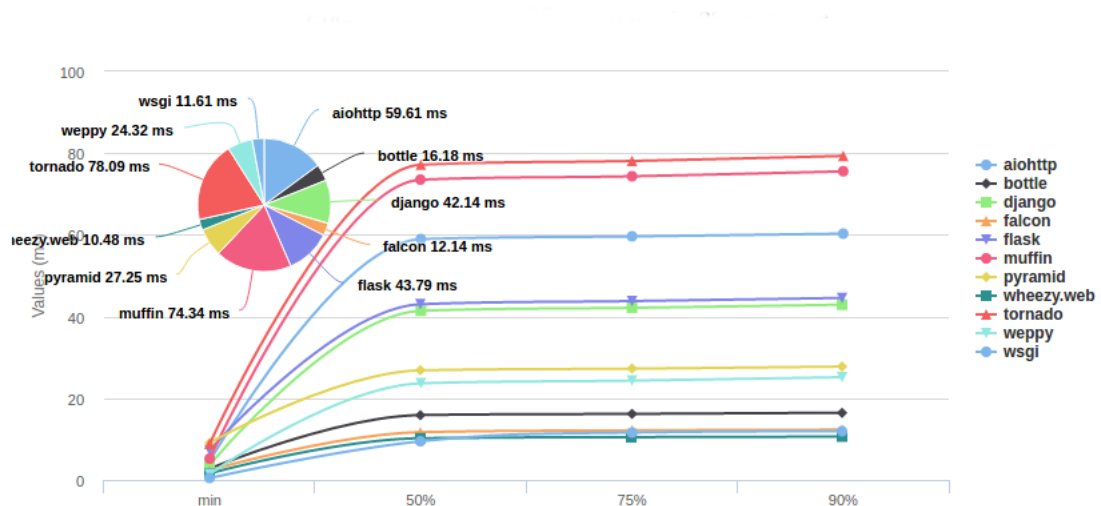
Of one side, we have the difficulties with the technology choice. In this case can be said that use AngularJS and Python has been literally perfect, beyond that the typical novice errors with the languages and their learning curves. So, in general, without a doubt, they will be chosen as technologies again.

About the platforms or technologies the point of view changes. If you are an expert developer using platforms like Google App Engine can be really interesting, because you are evaluated the rest of the options, but when you

don't have any practice, in my view, is not a good option. Moreover when the learning curve is so soft.

As in many new technologies is easy to do the first steps, but develop some bigger is another thing, especially when we are not talking about frameworks and languages standard as C++, Java or PHP. So, before to select one is really justify the spike of some of these.

About the API frameworks, Flask was a good selection, because their ease of use and lightness does it perfectly. However, when we analyze the performance, there are other solutions also in python faster. An example of this is Falcon¹. As can be seen in the next picture, extracted from py-frameworks-bench² project, we can see how Falcon has better performance than Flask. In this graphic is measure the response in ms of encode an object to JSON and return the response.



And Falcon would be our selection today (even more having some practice in company).

About Sphinx, is a good choice, but it has something that could be improved and is the requirements of that the project must have. To use Sphinx your project needs to be *executable*, understanding this as all imports must work. Many people do not know why Sphinx have this restriction, so if you only want doc, independently of if you structure is correct or not, well, this is impossible in the actual version, we hope that change this at any moment.

¹Defined as: "A very fast, very minimal Python web framework for building microservices, app backends, and higher-level frameworks". More info at: falconframework.org

²<http://klen.github.io/py-frameworks-bench/>

About the platform

In spite of Google App Engine is a good tool it has some drawbacks that some are very obvious at first and others that can go unnoticed until you are working with it. the sandbox restrictions as use python only in their version 2.7 at first is not a problem but when you discover that the most of the libraries that you need works better in 3.x, or some are deprecated in 2.7, out of maintenance is not a good signal, especially when you have some block and the help of community is focused on upper versions. It happened in the standard version of App Engine sandbox, to solve this the team of it launched App Engine Flexible Environment, where you can run any code but the configuration is not trivial and it although is configurable by docker file is a strange mix between the auto maintained and scalable isolated sandbox and the standard containers developer managed.

So, taking advantage of all services of the infrastructure of GCE, another architecture that we will develop in the future will run only into docker containers, using the services required (SQL, DataStore or another and running over compute layer, not with google app engine.

About the database

About database, without doubt, MongoDB is nowadays the better solution that can be used in a project with low relational requirements, because their learning curve is so good that is easy to have a good prototype of database layer soon, and the resources of the community greatly assist.

7.0.3 Design

About the user interface

There are any that has been critical in the develop and is that the assumption as a good idea that all interface always is loaded when a user enters to the app, but after some tests have been seen that is not the best way. For this reason, many teams that work with Angular choose an alternative, use a library specially designed to load the javascript files of the app needed in each part of the interface.

So, if a user is in the teaching section don't need all files of the reports section, and this files only will be loaded when the user insert in this section. To do this, the teams use the Require.js³ library as the standard dynamically loader and their use will be the next step to do the load of the app faster.

³Literaly from their website ,requirejs.org: *"RequireJS is a JavaScript file and module loader. It will improve the speed and quality of your code."*

About the compression of APIs

Is easy to notice that the API of Students Control microService is very semantic but very big also (become more complex to maintain and change) while of the teaching service API is very compact and little but less semantic, because the meaning of their resources is not clear and therefore their behavior neither. It was made on purpose it to check in the development which approach was more problematic or simple to explain or update.

And the conclusion is that to maintain the coherency between the domain drive development of the service and the expressivity of the API the customers do not need to know the internal work of the service, but need understand the logic behind of the items, so, in the future, it will be moved to an approach more human readable (in spite of all disadvantages, talking about code and maintenance).

7.0.4 The develop process

Can be really interesting if you use any methodology as SCRUM and the developer's team work together. But when the team is a single developer and the project only have sporadic contributions, all is more difficult to follow. Other techniques can be used but SCRUM, eXProgramming or something like this not works fine to only one person.

So, independently, is a good point to start to practice. For another hand, is noticed that the sporadic contributions as in a Hackathon or in a simple day are difficult too because, as is normal, the people require a time to understand the project, the technology and all related with it.

7.0.5 Opportunities

Take part of some software contest is the better decision that any software student can take. Visibility, networking, new friendship are only some benefits that can achieve.

Thanks to enrolling this project of the contests that the Open Source Department of the University of Granada with JJ Merelo as principal organizes was offer a job in a related software company with the technologies and patterns used. So, if any people think that the participation, the contribution, and involvement is not useful, is absolutely wrong. In all cases, this attitude front the students only return benefits.

7.0.6 Future of the project

At the beginning of the develop, the idea behind of this was put in production the result in a few months in beta mode in a school center of Granada, but now, the jobs opportunities referred above have been done that the project go to the another plane, less important, because the ideas and the philosophy are developed just now with another really good engineers in a company, building a privative software, architecture, and new related tools.

So, independently the license of the code has not changed, and the develop can go ahead with any developer or group of them that want. For another hand the continuous evolution of the technologies do this issue a bit difficult, and actually, it is another learned lesson about the innovate software using three party technologies, we will never have the safety that the technology never will change. If you are working with C++ or even Java with you own infrastructure the changes are minimal through the months, with third party technologies and support you need be at day with all changes and update almost all your software each year. So, in this cases is difficult that the continuity of the project will be ensured, at least without the original designer inside the new developer's teams. But, anyway, is only a point of view, with the software nothing can be assumed.

Independently, the code is open, to learn, to review, and maybe to help someone, so for this part, we are happy.

7.0.7 Open Source

Other conclusions obtained in the solution development are related to open source and the viability to survive on this. Many time in the college is easy to hear that the open source is a good way to start and is true, but not if you want to work of this. Work in open source projects is really interesting, for the community, for the workflow, for build some useful to the community and by a huge list of advantages. But this is possible only when for one hand you are working in a company and some of the projects of this are decided to be open, independently of the reason, community, better visibility, etc. or when you are a student and have the opportunity to free amounts of code, as this case. But for another hand, thinking to build a company, more o less big based on an open source solution is very very rare and complex, mainly to younger and inexpert software developers.

Obviously in the most cases, always there are some exceptions that are wonderful examples that project with an amazing grow, and a really amount of code that any developer must have would be open, always open, because there are any developer that can learn alone, without the community (in

any of their forms) and be in the obligation to contribute, to give back the favor.

7.0.8 Closing

This project has been a great opportunity to learn a lot about amount of things, but especially about myself, has been another opportunity to know how to deal with new challenges, how to work a first really subtle approach to project management and the most important, to know which my bigger faults. It have helped me to understand that this is only the begin, the beginning of the way that only can be covered if you do not stop to learn never, absolutely never.

So, **let's start!**

