MSc in Informatics Engineering
Internship
Final Report

YoubeQ Management Platform

Nuno Fauso Da Paixão Khan

nfkhan@student.dei.uc.pt

DEI Supervisor:

Fernando Barros

iNovmapping Supervisor:

André Santos

Date: 2 July 2013

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Abstract

System integration and distributed systems are essential concepts in our current web development era. When relying on these aspects, one must acknowledge the importance of having a system that is mature, secure, reliable and scalable.

The internship consists on the development of a management platform, called youbeQadmin, a web based application that integrates and manages youbeQ and Smarturbia, two existing applications developed by the company.

These applications, already exist as independent web applications , that are now partially integrated in one single management platform.

youbeqAdmin platform manages the youbeQ statistics and Smarturbia content (APPS). It enforces a distributed architecture while allowing the integration of these two modules/applications with the original web applications and with the possibility of easily integrating new modules into the platform in the future.

The development of youbeQadmin required significant modifications to Smarturbia, mostly to allow a REST communication between them and to add new features like changing the vehicle color or applying area limits for the user to explore the world.

Keywords

“youbeQ”, ”Smarturbia”, ”REST”, “Management”, “Web Development”,”DJANGO”, “Web Frameworks”, “youbeQadmin”

Index

Index of Tables

Index of Figures

Definition	Description
APACHE STRUTS	Web Framework
API	Is a specification intended to be used as an interface by software components to communicate with each other
COUCHDB	Open Source Non Relational Database
DJANGO	Web Framework
GIT	Version control system
JAVASCRIPT	Scripting Language
JSON	Lightweight data-interchange format
MVC	Architecture for building interactive applications
MVT	Architecture for building interactive applications
MYSQL	SQL Open Source Database
NO-SQL	Non-Relational Database
PHP	Scripting Language
PYTHON	Scripting Language
REST	Protocol specification for exchanging structured information
RUBY ON RAILS	Web Framework
SOAP	Protocol specification for exchanging structured information
SQL	Programming language for RDBMS
SVN	Version control system
XML	Extensible Markup Language
YII	Web Framework

Table 1: Definitions

Acronym	Description
API	Application Programming Interface
APP	Application
JSON	JavaScript Object Notation
MVC	Model-View-Controller
MVT	Model-View-Template
PHP	Hypertext Preprocessor
RDBMS	Relational Database Management Systems
REST	Representative State Transfer
SOAP	Simple Object Access Protocol
SVN	Subversion
XML	Extensible Markup Language

Table 2: Acronyms

1 Introduction

The internship consists on the development of a management platform, called youbeQadmin, which is a web based application that integrates and manages applications inside the companies scope. For now, these applications that require management are youbeQ and Smarturbia which are already deployed and available for the Internet users.

youbeQ and Smarturbia are using two products/services, Google Maps (web mapping service application and technology) and Google Earth (virtual globe, map and geographical information program), and since Google has provided an API to access this services from external sources, it has given the ability for third party companies to simply integrate their ideas and products with this services as well as others.

youbeQ as well as Smarturbia are the current main iNovmapping products/services, and these products are web applications that run on top of Google Earth/Maps API.

The main difference between them is the fact that youbeQ is the first Google Earth/Maps 3D social network where a user can meet other users online like other common social network except for the fact that it actually happens in a real Google Earth/Maps 3D environment.

Smarturbia is the second product that is well alike youbeQ except for the fact that at this point there is no “social” component making the application simply a means to travel inside Google earth with the companies own 3D models (cars, boats, airplanes).

youbeQAdmin will integrate the management of the applications into one single web based platform providing a faster and more reliable way to do all the operations required by the company. These operations include adding components, removing and editing them, having the desired effect on the original applications right after the server data replication.

youbeqAdmin platform manages the youbeQ statistics and Smarturbia content (APPS). It enforces a distributed architecture while allowing the integration of these two modules/applications with the original web applications and with the possibility of easily integrating new modules into the platform in the future.

The development of youbeQadmin required significant modifications to Smarturbia, mostly to allow a REST communication between them. The rest of the modifications were to add some new features to Smarturbia, like for example limit the area where the user can explore with a certain vehicle, or change the color of a vehicle.

The first part of the internship will focus around the requirements and design phase. During this period an elaborate State of the Art will be made in order to do a technology assessment to implement the requirements specifications.

The choice of the right tools and technologies for the development of this application, must be closely tied to the companies requirements and specifications as they are the the primary objectives. The other part of the internship will focus around implementation, verification and maintenance. During this period there will also be testing to validate the implemented code.

2 State Of The Art

This chapter shows the differences between Smarturbia and some other similar alternatives out there that also make use of the Google Earth API.

It also provides a study of Web Frameworks, and describes the advantages/disadvantages of using them for the development of youbeQadmin as opposed to not using one.

We also study the possibility of using ORM (Object Relational Mapping) for the development of youbeqadmin, if in fact we choose to use a Web Framework.

The annex for this chapter can be found at "*Annex A - State Of The Art"*.

For the state of the art, the only restriction imposed by the company was the PYTHON programming language and, if in fact a Web Framework was suitable for the companies needs after the study of the state of the art, DJANGO Web Framework [1] would be the main choice.

The use of open source technologies for the development of the application was the other restriction due to the companies current financial situation to purchase licenses and software beliefs.

Due to this restrictions, the Web Framework features will be according to DJANGO Web Framework features when comparing to developing the application without using a Web Framework.

2.1 Available Solutions

In this chapter we will do a brief description about available products on the Internet similar to iNovmapping's products (that use Google Earth/Maps API).

We will start by understanding what each one of the following presented products do, and in the end how are they different from iNovmapping's products (Smarturbia).

- **Smarturbia**
  - Smarturbia is a Web application that allows users to do virtual travelings around the world using the Google Earth/Maps API with a customized vehicle or standard vehicles (cars, bikes, race cars, boats, airplanes). Users can also create points of interest to be visited, delimit city areas and rate user/admin created APPS. Figure 1 shows a screenshot of this Web application.

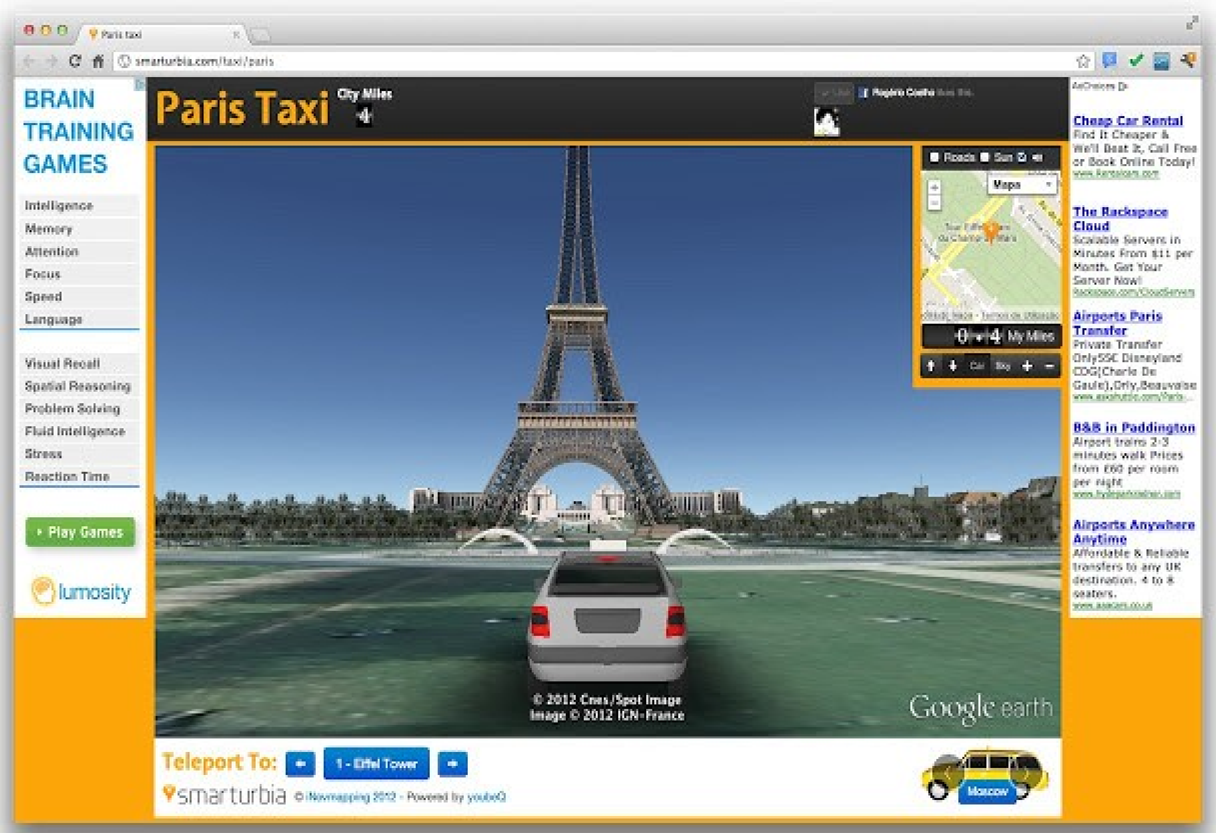


Figure 1: Smarturbia tour screenshot

- **GEFS (Google Earth Flight Simulator)**
  - GEFS (Google Earth Flight Simulator) is a real flight simulator Web application that uses Google Earth/Maps API to enable users to have a more realistic experience by traveling through an existing 3D virtual landscape. The flight models are complete enough to deliver a realistic flight simulation. Figure 2 shows a screenshot of this Web application.

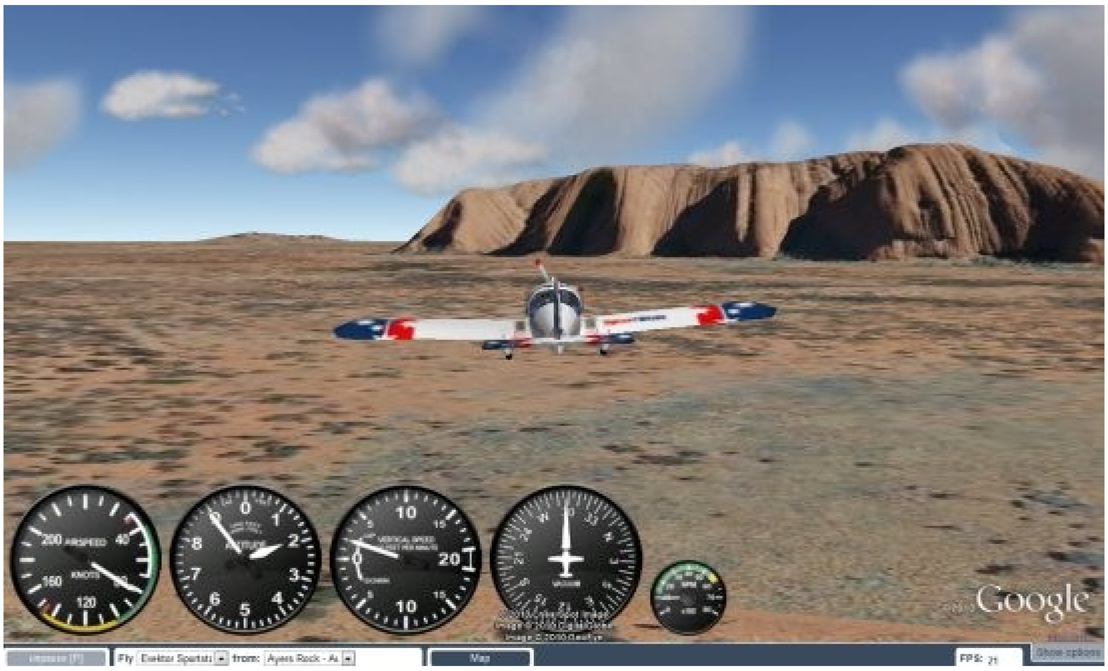


Figure 2: GEFS (Google Earth Flight Simulator Screenshot)

- **Planet In Action (A-Team Van)**
  - A-Team Van is a Web application based on the movie *"A-Team"* that uses Google Earth/Maps API to enable users to drive around the world guided by specific tasks such as reaching a certain amount of speed, performing a variety of jumps, crashing the van a certain number of times. Figure 3 shows a screenshot of this Web application.

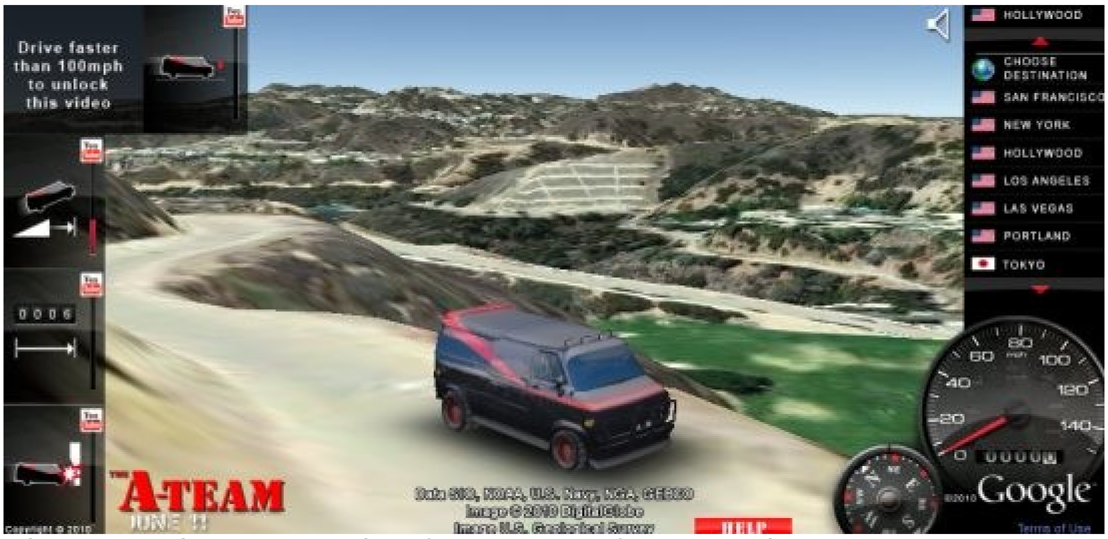


Figure 3: Planet In Action (A-Team Van) Screenshot

- **Sailing Alone Around the World**
  - Sailing Alone Around the World is a Web application based on the book "*Sailing Alone Around the World*" that uses Google Earth/Maps API to enable the user to experience the voyages of captain Joshua Slocum [2]. These are guided tours with a narrator impersonating Joshua Slocum and narrating his experiences. Figure 4 shows a screenshot of this Web application.

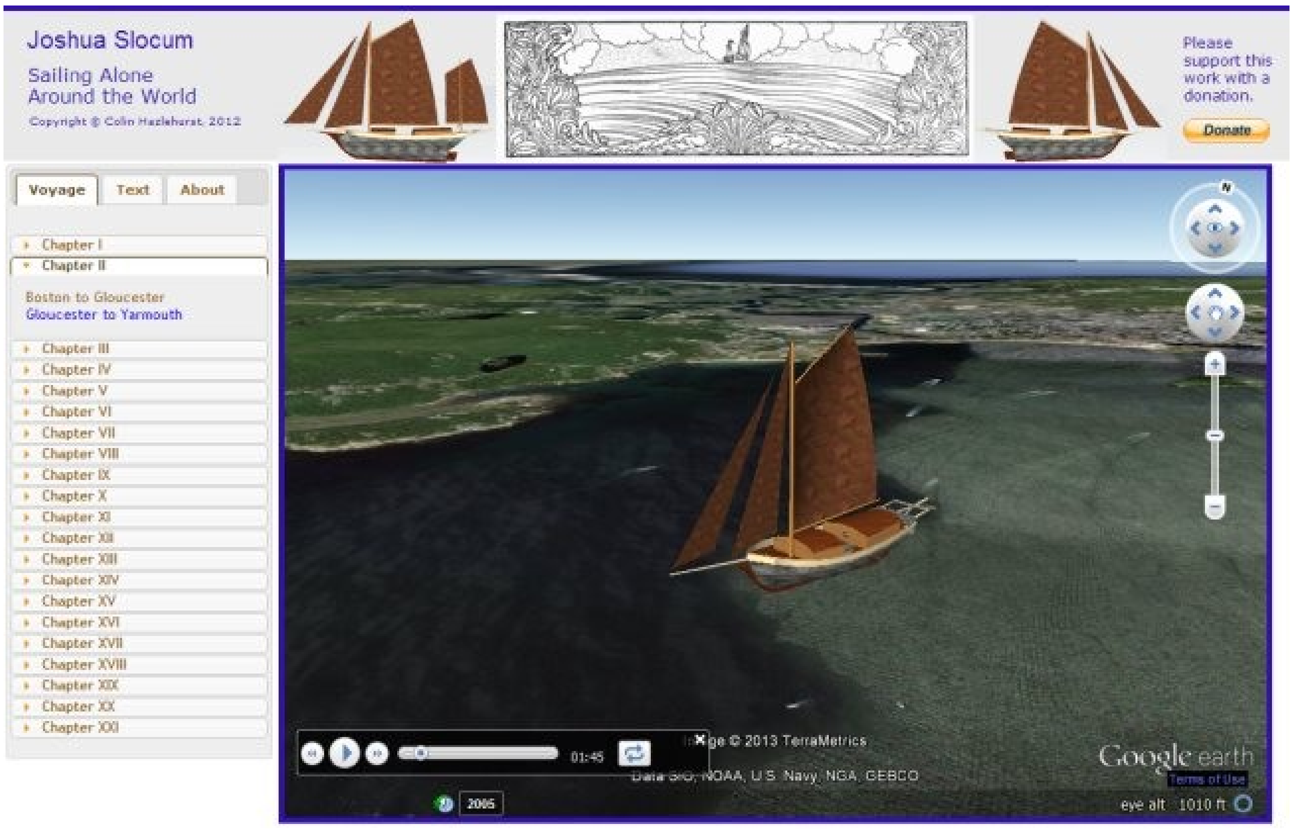


Figure 4: Sailing Alone Around the World Screenshot

When comparing iNovmapping's Smarturbia and the rest of the presented alternatives we can arrive at a few conclusions:

- All of these web applications use Google Earth/Maps API.
- All of them involve vehicles controlled by the user with the exception of "Sailing Alone Around the World" where the user only has a visual representation of the boat sailing but cannot actually control the boat.
- GEFS is a real flight simulator, so a user must be a flight simulator affectionate, this is not a web application for the conventional user.
- Sailing Alone Around the World is only for people interested on the voyage stories of Joshua Slocum.

- Each one of them uses only one specific vehicle, with the exception of Smarturbia that uses more than one vehicle.
- Only Smarturbia allows user vehicle customization, adding points of interest and delimiting areas to travel.
- Smarturbia and A-Team van seem to be ideal WEB applications for the average user.

2.2 Web Frameworks

Web frameworks [3] are frameworks that help developers build strong and reliable Web applications without concerning themselves with some minor/larger details that usually appear when developing applications without these frameworks.

“DJANGO is a high-level PYTHON Web Framework that encourages rapid development and clean, pragmatic design.” [1]

The idea of the framework is to alleviate that additional overhead that many times comes associated with the developers everyday tasks. An example would be a proper authentication / registration system that most dynamic websites/web applications tend to have. This authentication system would in fact take some amount of time to develop every time we decided to develop a website with it.

Other examples would be the fact that many frameworks provide libraries for database access, templating frameworks and session management and they often promote re-usability of code, which can be our own code in our current application or another application we have developed, or even third party code integration with our applications.

Frameworks are in fact strongly present in software development. Even if we decide not to use a Web Framework, we will definitely have to use another framework, even if we decided to write all of our own code from scratch it would be almost impractical to not use them.

“Tasks that usually would take you hours and hundreds of lines of code to write, can now be done in minutes with pre-built functions. Development becomes a lot easier, so if it’s easier it’s faster, and consequently efficient.” [4]

In this section we will be discussing its features, exploit its advantages and disadvantages and try to relate heavily to the idea of not using them and what that would mean.

2.2.1 Features

This section will introduce to features commonly offered by Web Frameworks.

The features are as follows:

- **Web Template System**
  - A web template system is a software used to produce dynamic web pages, this system uses a template engine which in turn is a software designed to process web templates and content information to produce output web documents. Templates play an important role when wanting to separate the presentation layer from the business logic layer, or even the Model Layer (in an MVC environment). This is done due to the fact that in a template there is no need to know the server-side programming language, just the HTML and the frameworks template system syntax which is in most cases quite simple and similar to plain HTML.
- **Caching**
  - The most common trade-off in dynamic websites is that each time a user requests a page, the Web Server makes calculations from the database actual query, to the business logic to provide the information for the template rendering in order to create the web page seen by the user. This is quiet expensive in terms of a processing-overhead perspective. For most web applications, this might not be an issue, but for high traffic web applications it is essential to reduce the overhead as much as we can.
- **Security**
  - This is an everyday concern in the development world, where developers must work on the authentication and authorization of the application. Some applications provide help with this matter, bringing along some authorization and authentication frameworks built in the Web Framework.
  - There are also other things we must worry about with our applications security such as SQL injection, cross site scripting (XSS), cross site request forgery (CSRF), click-jacking, etc. Mature frameworks should be able to help developers easily protect their Web applications views against this matters.
- **Database access, mapping and configuration**
  - To avoid configuration and mapping hassles, most frameworks provide an API that allows a quick and painless database setup. Developers are able to simply choose the name and type of the database that they want to use, enter the database credentials and its all ready for them to start using it.

  - Some databases provide ORM (object relational mapping) for the object mapping.
- **URL Mapping**
  - A URL mapping system, allows the use of regular expressions for pattern matching, increasing the simplicity of the site, and allowing better indexing by the search engines. This allows the URL to be easily read and written by the users and provides the search engines with better information about the applications structural layout.
- **Ajax (Asynchronous JavaScript and XML)**
  - This is a very common technology used in web development for asynchronous communication between the client and the server. The main idea here is to give the possibility to create web pages that look more responsive by exchanging small packets of data with the server so that the entire page doesn't have to reload every time we decide to put some new and dynamic content. Some Web Frameworks have a native Ajax library support built in, others have third party resources for easy Ajax support and integration.
- **Web Services**
  - Web Services are an indispensable feature in web development, due to this fact some Web Frameworks have native support, and others also support it by third party applications integration.

For the specific case of DJANGO Web Framework [1], the highlighted features from its creators are as follows:

- **MVT (Similar to MVC)**
  - MVT is essentially the same as MVC [5] in terms of architecture. The only difference is the fact that the DJANGO developers did not agree with the MVC nomenclature (Model-View-Controller) and decided to rename it (Model-View-Template) where the Model is the same in both architectural patterns, but the MVT view and template are actually swapped around with the MVC view and controller, so the direct translation would be MVC-MTV.
- **Object-relational mapper (ORM)**
  - DJANGO gives the ability to write entire database models in pure PYTHON. It gives a free ORM Framework with a dynamic database-access API. Nevertheless it is still possible to write raw SQL when optimizations become needed.
- **Automatic administration interface**
  - DJANGO gives a “ready to use” administration interface. After the developer has defined the database models, he can easily import

them to the administration interface, and start using it, without the need to write the “standard” code to interact with the designed models.

- **Elegant URL design (REGEX)**
    - The URL design was already explained above, but there is a need to emphasize that the URL design in DJANGO is fairly easy to use, as it is listed in a particular URLS file and it also allows sub-URLS from within each URL.
- **Template system**
    - DJANGO comes with its own template language, and although its not Python, it still is very similar and brings the ability to have a layer separating the design, content and Python code.
- **Cache system**
    - DJANGO allows developers to choose between different types of cache (MEMCACHE, Database Caching, File system cache etc) giving complete power to the developer according to his needs.
- **Internationalization**
    - Internationalization is something very important in a mature framework, and its goal is to allow a single Web application to have its content in different languages and formats according to the applications targets.
- **Security**
    - DJANGO has most of the up to date security measures, some of them are cross site scripting protection, cross site request forgery protection, SQL injection protection, click-jacking protection, SSL/HTTPS, host header validation.

2.2.2 Advantages

There are in fact some big advantages when developing software with Web Frameworks, and we will present some of them below:

- **Reusable Code**
    - We can easily reuse our own code or third party code with new applications, in other words its just like connecting two different applications and make them work together to achieve a certain goal.
- **Rapid Development and with less code**
    - Web Frameworks help us work faster because we have a lot of tasks that may be already built in the Web Framework core, or we can easily install that code from third party sources. A good example is DJANGO-

registration module.

- **Security (big security implementations)**
  - The best security experts in the world have contributed to this open source project (DJANGO), so its safe to assume that the best security policies and implementations are being applied to each DJANGO version.
- **Organized application structure (MVC/MVT)**
  - DJANGO enforces an MVT (similar to MVC) architecture with its folder structure, and way of making the application files interact with each other, so we know how well organized our application is when we start developing it.
- **Scalable Applications**
  - If properly coded, DJANGO allows applications to be scalable. DJANGO applications within a project can be easily separated from each other to be deployed on different machines. This is one of the DJANGO philosophies, to provide potential scalable applications.
- **Easy database interaction (flexibility)**
  - We can easily choose the type of relational database engine that we want to use and if we use ORM models, everything on the programming side stays exactly the same, except if a certain database engine provides specific features that others do not possess.
- **Lower development costs**
  - The faster we develop our applications, the cheaper the development costs will be.

2.2.3 Disadvantages

Whenever we have advantages we definitely have disadvantages, and in this section we will also present some of them:

- **Additional Overhead**
  - There is a lot of unwanted/not needed code in DJANGO's core. This is due to the fact that depending on the application we develop, we might need some specific code or not.
- **Lost understanding on third party source code**
  - Its not easy to understand how another programmer developed his application without proper documentation. Reading the documentation can help us with this matter, but might not be enough for us to quickly add our own modifications to someone else's code.

- **Very steep learning curve**
    - Its not easy to start using Web Frameworks. The learning curve is steep as at the beginning we do not understand the framework's way of doing things until we actually start developing our own application. When migrating from a different framework, this process can become easier thou.
- **Inflexible when we want to change core functionalities**
    - Changing core functionalities of Web Frameworks might not be a good idea. We would have to understand it internally really well before we can do any change that wasn't meant to be done by its developers.
- **Configuration Cliff**
    - Easy configurations are easy to do in Web Frameworks. But complex configuration do require a better understanding of the specific Web Framework and a lot of documentation reading.
- **Framework Errors**
    - As every piece of software, we always encounter “bugs”. When encountering Web Frameworks bugs, we will probably have to wait for the Web Framework developer to fix the bug, or take a big risk and try to fix is ourselves (if it is open source).

2.3 Object-Relational Mapping (ORM)

Since the early days of computing, it became clear how important it was for the applications to retain the content of data structures not only in memory, but also in a non-volatile storage, like an Hard Drive. This process is called Persistence, and without it data exists only in memory and is lost whenever the application shuts down.

Object-relational mapping (ORM) [6], is a technique for converting data between incompatible type systems in object-oriented programming languages. The common primary feature of an ORM implementation (such as Hibernate for example) is mapping from the programming language classes to database tables (consequently from its data types to SQL data types). An ORM implementation will also provide data query and retrieval facilities.

The ORM implementation automatically generates the SQL code, and allows the developer to only focus in a class object-oriented point of view for the development of the application. The querying also tends to be much simpler then a raw SQL query, at least for the ORM implementations supported queries.

ORM has constantly been a target when it comes to performance. In this section, we will elaborate a bit more on the ORM performance issues, main obstacles for its adoption and a possible direction it could take to be able to eliminate this issues.

2.3.1 Performance

Before we proceed, in this chapter we must bear in mind that ORM is a fairly recent technology [6] , and although earlier implementations were clumsy and slow, there has been room for constant and substantial improvements. The popular and current ORM solutions, by contrast, provide features that save time for software developers and improve performance and normalization.

Some of the most significant an impressive improvements to ORM systems have been in the area of performance. Still, early criticisms about ORM have been around the fact that an ORM solution is often worst than raw SQL, performance wise. This is generally true, and it is due to the fact that ORM adds overhead (extra layers of abstraction) to our code.

Since this is a valid concern, and because of this modern ORM implementations use a variety of tricks to improve performance. A few examples are:

- **Caching**
  - Since ORM suffers from poor performance in relation to raw SQL solutions, caching the results of queries locally, applications communicate less with the database improving performance.

- **Lazy Fetching**
  - Decides whether to load child objects while loading the Parent Object. This is in fact an improvement, as the applications may not need the child objects but if it does, they will all be previously fetched and stored in memory for the applications use, subsequently providing less database accesses.
- **Dirty Checking**
  - Is an ORM feature that checks if an object has been modified or not, and determines if the object needs to be updated or not. As long as the object is in a persistence state, the ORM implementation monitors any changes to the objects and executes the SQL. Note that for dirty checking to exist, the object must exist in cache.

These tricks reduce the frequency with which the application has to connect and communicate with the database, becoming an optimized solution.

2.3.2 ORM Main Obstacles

Although ORM is a mature technology and has a vast number of increasingly advanced and readily available solutions, ORM has yet to see its adoption in everyday industrial operations. In this section we will show what are the main obstacles that are keeping ORM from being adopted by every developer/company.

- **The Learning Curve**
  - Since ORM is not a standard technology in the industry, not every developer knows how to use it. From the perspective of a software developing company, this may become a serious obstacle, as in order to effectively start using ORM, they would have to train their developers and this could be less cost effective.
  - Another reason is the fact that ORM is not natively available in every programming language, so this would mean having to install it in every machine in the company for the purposes of developing, testing and deploying.
- **Perceived performance limitations**
  - Many developers and managers have the idea that an ORM solution is slower than raw SQL. This is in fact true, but nevertheless this analysis is based on old ORM implementations. Old naive architectural ORM implementations, as well as new ones, will indeed be slower then hand-crafted persistence code, as they introduce the additional overhead of reading meta-data, reflecting on classes (if necessary), generating the SQL code, and so on. In a raw SQL this overhead would simply not exist.

  - However, modern popular ORM implementations like for example Hibernate (JAVA) do not have a naive architecture. Instead, they do introduce performance innovations like lazy checking and automatic caching to try and balance out the additional overhead they introduce. This improvements can also be done in raw SQL but the programming cost to do so is in fact high.
  - Regardless of this improvements, ORM still is slower then raw SQL. But if the ORM implementation is well architectured, it is not that slow and because of its advantages becomes plausible enough to be standardized in a company, and only resort to raw SQL for optimization tasks.
- **Sensitivity to architectural revisions**
  - ORM would work best once the mapping was completed and in fact no more architectural changes were made to the application, database or meta-data, ever again. Unfortunately this scenario is unrealistic, as in reality developers frequently want to make architectural changes late in the development cycle.
  - Sometimes this modifications can be modest (like for example just adding an attribute to an existing object(database), but other times it can be more complex then that like for example rearranging the structure of previously unrelated objects to make them related, as well as adding an elaborate inheritance hierarchy. This would implicate a change in all three components of the persistence system, the application, the database and the meta-data.
  - This operations are very difficult to automate, consequently most ORM solutions have unsatisfactory support for revisions of the architecture of a system after it has been mapped. Nevertheless this operations are also painful in hand-mapped solutions.
- **Accommodation of legacy systems**
  - Legacy systems are new applications that must use previously available databases, or a data source provided by a client or third party. In this case, the new application has no input in the matter of the legacy database architecture, having to use it exactly has they were designed. In insufficiently flexible automated persistence layers, this can be problematic. If the meta-data language is not especially inexpressive, it may even be difficult or even impossible to represent the mapping to the legacy database.
  - Problems also arise the other way around, in other words when an application that does not use ORM tries to use a database that was created and is managed by an ORM system. This happens, because many ORM products expect to have full ownership of their databases. This is because in particular, they may implement caching, transactional and delayed persistence strategies that make this

assumption. As an example we can imagine an ORM application that retrieves some information from the database. Later on the NON ORM application changes that information, and latter on when the ORM application tries to retrieve that object it may go directly to the cache, as it does not have the information that the object has been changed in the database making the data inconsistent and obsolete.

- **Limitations in expressing queries**
  - ORM does a good job at keeping track of the structural relationships between objects , and shuffling data between objects and databases. But unfortunately it does not perform that well when it comes to writing more complex queries. When this happens if then we might have the lock away ORM for a while and write raw SQL code.

2.3.3 Possible improvements for ORM

- **Improve awareness of ORM and ORM tools**
  - Unfamiliarity with ORM and its current improvements and methods may cause developers to avoid ORM. Making them aware of ORM's advantages (as well as disadvantages) may in fact help ORM become a standardized solution.
- **Continue to improve performance**
  - Most developers care more about performance than rapid development and comfort, making performance a very important weakness of ORM. And like every other technology, there is always room for improvement.
  - The largest improvements can be made by reducing the frequency on which the application communicates with the database, by making caching algorithms smarter and by somehow improving the performance of expensive procedures such as navigating object graphs.
- **Standardize persistence solutions**
  - This is probably the most important goal for ORM. When every developer can just know one ORM solution and apply the same knowledge and use it in another ORM solution, then the learning curve will definitely decrease. This will gain the developers attention.
- **Integrate transparent persistence into object-oriented languages**
  - In the future, ORM implementations could natively be integrated in all object-oriented programming languages eliminating the hassle of using third party libraries.

- **Improve automated support for architectural revisions**
    - When a developer decides to restructure its database, ORM should have better algorithms for identify this changes and providing support for automating architectural revisions.
- **Better accommodate legacy systems**
    - Shared databases across applications are a common thing nowadays. ORM should be able to play nicely with ORM and non ORM systems that want to use those databases, by means of configuration for the object-relational mappers to work alongside legacy and other applications that might want to use it.
- **Support expressive, powerful queries**
    - Complex queries are a technical and conceptual problem for many ORM solutions. Future querying solutions should avoid the raw SQL fall-back by providing an easier way for complex queries.

2.4 Conclusion

After finishing the study of the State Of The Art, it was possible to have a more enlightened idea on what are Web Frameworks, and how can they be used to help us achieve a complete and mature Web Application with less development time.

DJANGO has in fact proven to be a mature solution as it offers most of the common features of Web Frameworks as well has some own particular advantages over some other Frameworks.

At the beginning the company was a little reluctant due to performance issues, but after this study, and after knowing that optimizations can easily be added with DJANGO at a later stage (if in fact they are needed) the decision to adopt DJANGO Web Framework for the development of this project was taken.

3 Project Planning

This section refers to the project planning for the internship.

First we will see the software development process used in iNovmapping. Afterwards we will go through the planned tasks for the duration of the internship, as well as presenting some GANTT diagrams.

All the annexes for this chapter can be found at *"Annex B - Software Requirements"*.

3.1 Software Development Process

youbeQadmin follows a hybrid approach between the WATERFALL model [7] and the AGILE development methodology [8].

This is given to the fact that all the classic WATERFALL steps are followed, in other words:

- Firstly the team **analysis**, then determines and prioritizes requirements / needs.
- After, in the **design** phase business requirements are translated into IT solutions, and a decision taken about which technology (PYTHON, JAVA or MySQL,etc) is to be used.
- Once processes are defined, code **implementation** takes place.
- The next stage evolves into a fully tested/verified solution for implementation and **testing** for evaluation by the end-user.
- The last and final stage involves evaluation and **maintenance**,with the latter ensuring everything works fine.

This approach seems efficient in theory, but in practice some problems may arise and this is an issue in this specific project where the platforms to be managed are still in development and new features are being introduced, making it impossible to close the requirements phase.

That is why the AGILE development also comes into play by allowing the quick implementation of new features that came along making it a hybrid approach.

The Software Development Process for the youbeQAdmin is presented in Figure 5 :

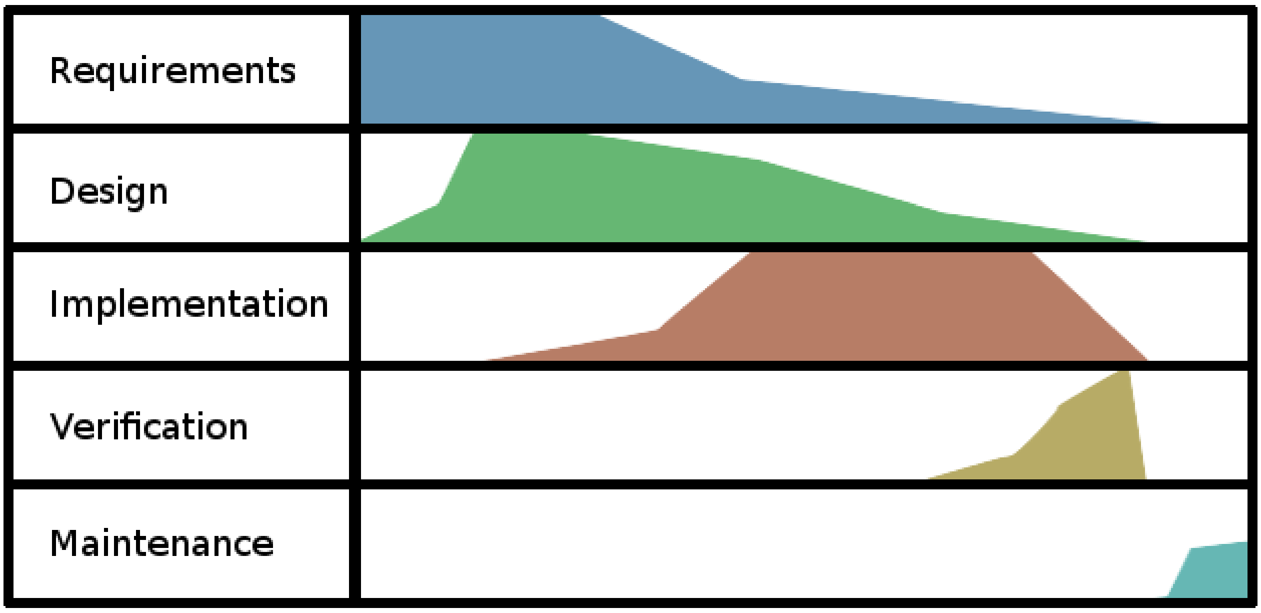


Figure 5: Software Development Model in iNovmapping

We can easily acknowledge that all phases start even if the previous one isn't completely closed. But we can also notice a reduction of the previous phases in favor of the new ones.

- **Project Execution Control**
  - There are regular meetings with the project supervisor, to monitor the project related subjects, such as its execution and possible deviations from the original planning.

- **VERSIONING**
  - The project uses GIT, a distributed revision control system available for VERSIONING of documents and software. This repository stores all the documentation and code related to the project.

3.2 Planning

3.2.1 First Semester

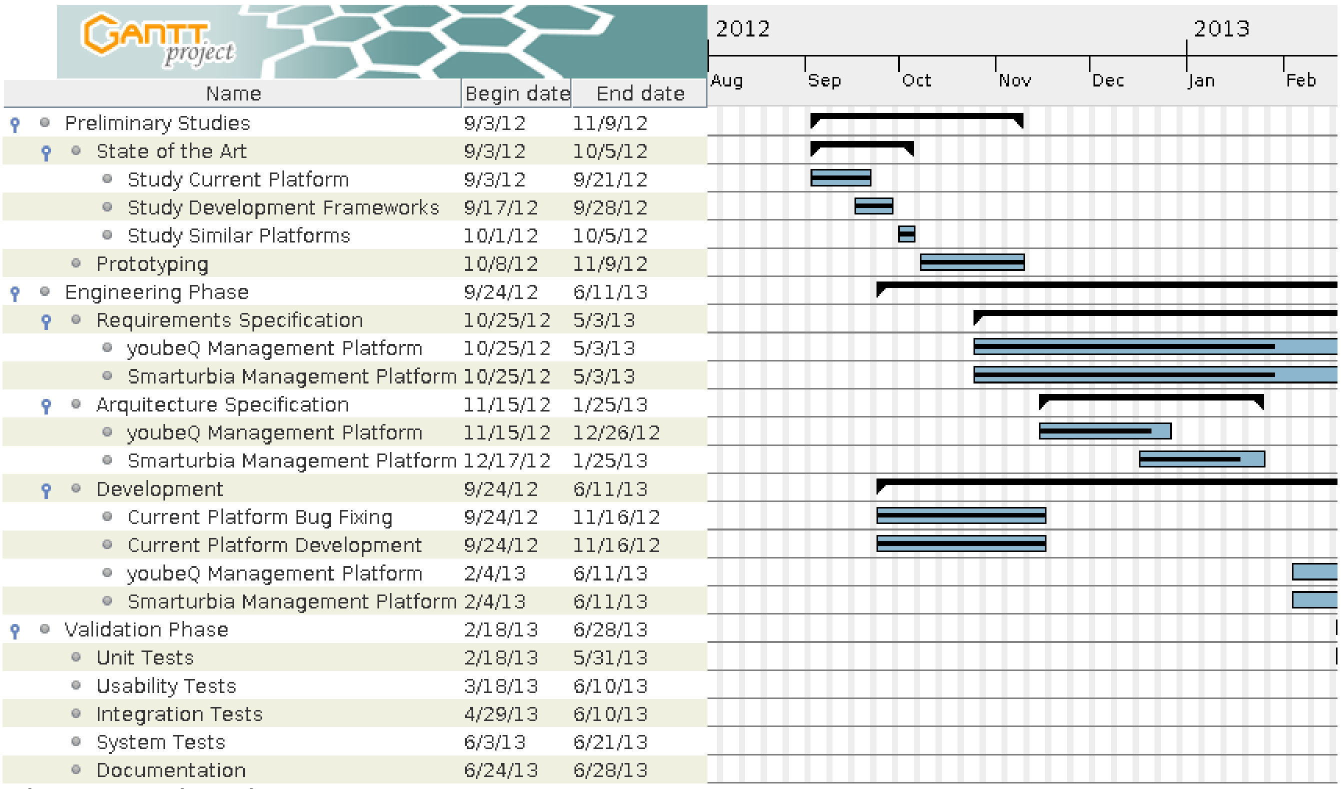
The first semester had a research component, consisting of the state of the art. This was the basis for the creation of all the documentation for the project, such as software requirements and architecture.

The state of the art involved the study of Web Frameworks, more specifically DJANGO and ORM (Object Relational Mapping). It also involved a comparison between Smarturbia and other similar alternatives.

Afterwards the software requirements were specified for the two modules as well as the main system.

Regarding the architecture, some UML diagrams where specified to better understand the overall system. This diagrams include, component, use cases, activity diagrams and class diagrams.

The work done during the First Semester corresponded to the requirements phase , design phase and part of the implementation of a prototype. Figure 6 shows this GANTT diagram.

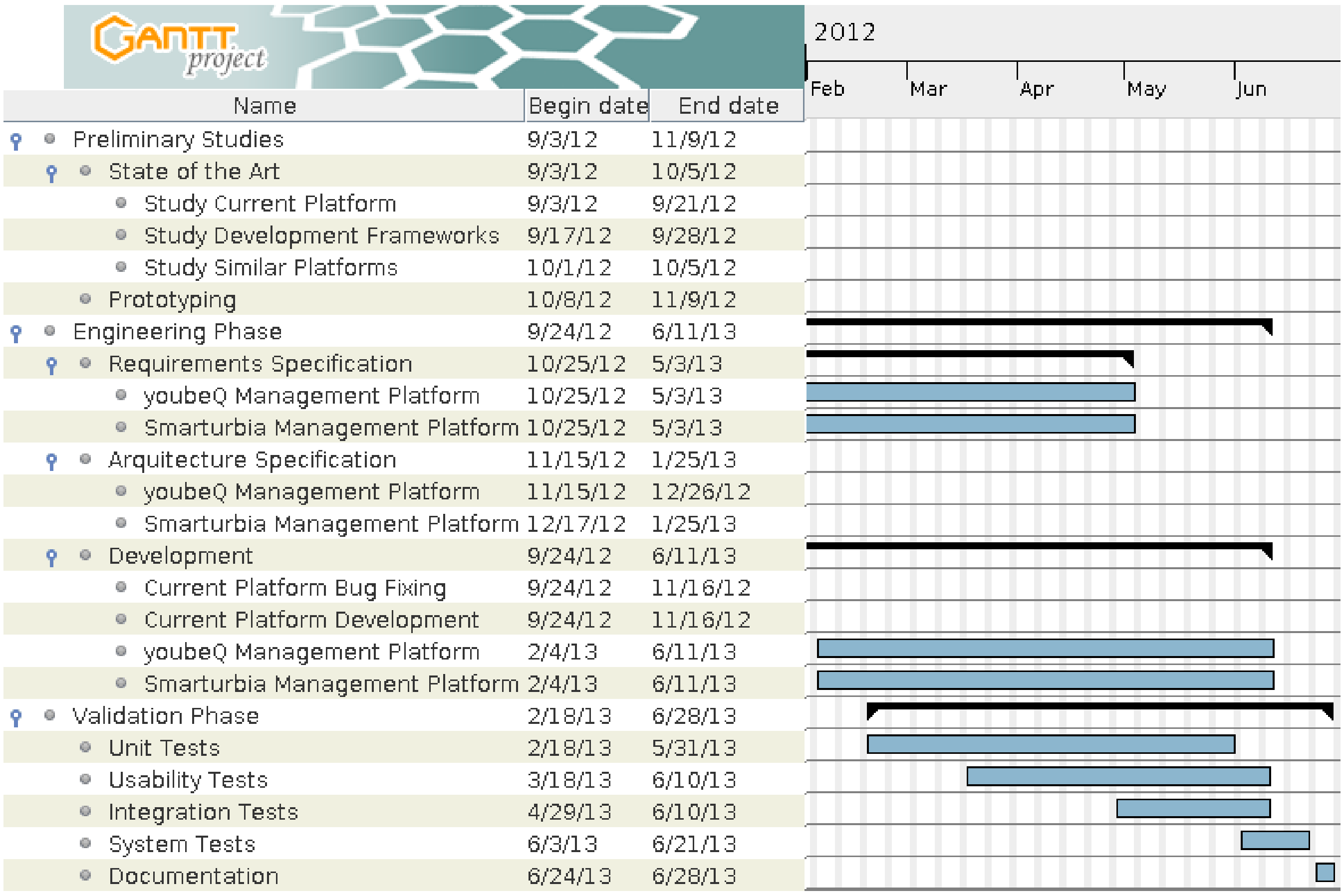


Name	Begin date	End date
Preliminary Studies	9/3/12	11/9/12
State of the Art	9/3/12	10/5/12
Study Current Platform	9/3/12	9/21/12
Study Development Frameworks	9/17/12	9/28/12
Study Similar Platforms	10/1/12	10/5/12
Prototyping	10/8/12	11/9/12
Engineering Phase	9/24/12	6/11/13
Requirements Specification	10/25/12	5/3/13
youbeQ Management Platform	10/25/12	5/3/13
Smarturbia Management Platform	10/25/12	5/3/13
Arquitecture Specification	11/15/12	1/25/13
youbeQ Management Platform	11/15/12	12/26/12
Smarturbia Management Platform	12/17/12	1/25/13
Development	9/24/12	6/11/13
Current Platform Bug Fixing	9/24/12	11/16/12
Current Platform Development	9/24/12	11/16/12
youbeQ Management Platform	2/4/13	6/11/13
Smarturbia Management Platform	2/4/13	6/11/13
Validation Phase	2/18/13	6/28/13
Unit Tests	2/18/13	5/31/13
Usability Tests	3/18/13	6/10/13
Integration Tests	4/29/13	6/10/13
System Tests	6/3/13	6/21/13
Documentation	6/24/13	6/28/13

Figure 6: Planning 1º Semester

3.2.2 Second Semester

The goal for the second semester is to finish the specification of the architecture for both modules, and start the development. The second semester will correspond to the implementation phase , verification phase and maintenance phase. Figure 7 shows this GANTT diagram.



Name	Begin date	End date
Preliminary Studies	9/3/12	11/9/12
State of the Art	9/3/12	10/5/12
Study Current Platform	9/3/12	9/21/12
Study Development Frameworks	9/17/12	9/28/12
Study Similar Platforms	10/1/12	10/5/12
Prototyping	10/8/12	11/9/12
Engineering Phase	9/24/12	6/11/13
Requirements Specification	10/25/12	5/3/13
youbeQ Management Platform	10/25/12	5/3/13
Smarturbia Management Platform	10/25/12	5/3/13
Arquitecture Specification	11/15/12	1/25/13
youbeQ Management Platform	11/15/12	12/26/12
Smarturbia Management Platform	12/17/12	1/25/13
Development	9/24/12	6/11/13
Current Platform Bug Fixing	9/24/12	11/16/12
Current Platform Development	9/24/12	11/16/12
youbeQ Management Platform	2/4/13	6/11/13
Smarturbia Management Platform	2/4/13	6/11/13
Validation Phase	2/18/13	6/28/13
Unit Tests	2/18/13	5/31/13
Usability Tests	3/18/13	6/10/13
Integration Tests	4/29/13	6/10/13
System Tests	6/3/13	6/21/13
Documentation	6/24/13	6/28/13

Figure 7: Planning 2º Semester

4 Proposed Approach

4.1 Requirements

This chapter presents a summary of the Requirements Specification for the youbeQadmin application and its modules.

The complete Requirements Specification can be found in *"Annex B – Software Requirements"*.

youbeQadmin as well as its modules follow a set of functional requirements, these being:

- Registration
- Authentication
- Error Handling (Registration, Authentication, Server, Connection)
- User Management (Create, Edit and Delete)
- Application Management (Create, Edit and Delete)

The next modules also extend the previous functional requirements, as well as adding a few of their own.

- **youbeQ Module of youbeQAdmin:**
  - Statistics Management (Add servers, edit servers, remove servers, alerts, server configuration)
  - Real Time Monitoring
  - Manage Publicity
- **Smarturbia Module of youbeQAdmin:**
  - APP Management (Create, Edit, Delete)
  - Basic Info Management
  - Category Listing
  - User APP Rating

Aside from the functional requirements, we also have the following non-functional requirements:

- Performance Requirement (Support at least 10.000 users)
- Data Integrity Requirement (Consistent Database, FIFO Database Updates)
- Usability Requirement (Web Usability)
- Interface Requirement (Social Networks external API's)

- Operation Requirement (Need resource such as web servers, databases)
- Security Requirement (Secure user authentication, Password Encryption, HTTP's, user permissions)
- Portability Requirement (Multi-platform)
- Interoperability Requirement (APP integration)
- Fast Development and easy third party module integration

4.2 Requirements Status

In this section, we will see the requirements for the project including their priorities and if they were completed or not.

An incomplete requirement basically means that the company decided to reduce their priority level. All of these incomplete requirements should be implemented in the future, but for the purpose of this internship they are not that important.

The full details of each requirement is in *"Annex B – Software Requirements".*

Requirement	Code	STATUS
REGISTRATION	YOUBEQADMIN-SRS-00010	COMPLETE
REGISTRATION ERROR HANDLING	YOUBEQADMIN-SRS-00020	COMPLETE
AUTHENTICATION	YOUBEQADMIN-SRS-00030	COMPLETE
AUTHENTICATION ERROR HANDLING	YOUBEQADMIN-SRS-00040	COMPLETE
USER MANAGEMENT: ADD USER EDIT USER DELETE USER	YOUBEQADMIN-SRS-00050 YOUBEQADMIN-SRS-00051 YOUBEQADMIN-SRS-00052 YOUBEQADMIN-SRS-00053	COMPLETE
APPLICATION MANAGEMENT DIFFERENT ACCOUNT TYPES	YOUBEQADMIN-SRS-00060 YOUBEQADMIN-SRS-00061	COMPLETE
ACCOUNT CONFIGURATION	YOUBEQADMIN-SRS-00070	NOT COMPLETE
USER PROFILE INFO	YOUBEQADMIN-SRS-00080	NOT COMPLETE
EDIT USER PROFILE INFO	YOUBEQADMIN-SRS-00090	NOT COMPLETE
CONNECTION ERRORS HANDLING	YOUBEQADMIN-SRS-00100	COMPLETE
SERVER ERRORS HANDLING	YOUBEQADMIN-SRS-00110	COMPLETE
APP MANAGEMENT CREATE APP EDIT APP REMOVE APP TRY APP	YOUBEQADMIN-SRS-00150 YOUBEQADMIN-SRS-00151 YOUBEQADMIN-SRS-00152 YOUBEQADMIN-SRS-00153 YOUBEQADMIN-SRS-00154	COMPLETE
INFO MANAGEMENT	YOUBEQADMIN-SRS-00160	COMPLETE
AREA LIMITS	YOUBEQADMIN-SRS-00170	COMPLETE
CREATE AREA	YOUBEQADMIN-SRS-00171	COMPLETE
EDIT AREA	YOUBEQADMIN-SRS-00172	COMPLETE
REMOVE AREA	YOUBEQADMIN-SRS-00173	COMPLETE
KMZ/KML MODELS	YOUBEQADMIN-SRS-00180	COMPLETE
CREATE KMZ/KML MODELS	YOUBEQADMIN-SRS-00181	COMPLETE
EDIT REMOVE KMZ/KML MODELS	YOUBEQADMIN-SRS-00182	COMPLETE
POIS MANAGEMENT	YOUBEQADMIN-SRS-00190	COMPLETE
ADD POI	YOUBEQADMIN-SRS-00191	COMPLETE
EDIT POI	YOUBEQADMIN-SRS-00192	COMPLETE
REMOVE POI	YOUBEQADMIN-SRS-00193	COMPLETE
POIS CONNECTION	YOUBEQADMIN-SRS-00200	NOT COMPLETE
EDIT POI ICON	YOUBEQADMIN-SRS-00201	NOT COMPLETE

TOP APPS	YOUBEQADMIN-SRS-00210	COMPLETE
CATEGORY LISTING	YOUBEQADMIN-SRS-00220	COMPLETE
USER RATING	YOUBEQADMIN-SRS-00230	COMPLETE
USER ADD RATING	YOUBEQADMIN-SRS-00231	COMPLETE
USER EDIT RATING	YOUBEQADMIN-SRS-00232	COMPLETE
USER COMMENTS	YOUBEQADMIN-SRS-00240	NOT COMPLETE
USER ADD COMMENTS	YOUBEQADMIN-SRS-00241	NOT COMPLETE
USER EDIT COMMENTS	YOUBEQADMIN-SRS-00242	NOT COMPLETE
ADMIN REMOVE COMMENTS	YOUBEQADMIN-SRS-00243	COMPLETE
USER REPORT APPS	YOUBEQADMIN-SRS-00244	NOT COMPLETE
AUTOMATICALLY REMOVE APP	YOUBEQADMIN-SRS-00245	NOT COMPLETE
MANAGE PUBLICITY	YOUBEQADMIN-SRS-00470	NOT COMPLETE
STATISTICS MANAGEMENT ADD SERVER EDIT SERVER REMOVE SERVER	YOUBEQADMIN-SRS-00100 YOUBEQADMIN-SRS-00101 YOUBEQADMIN-SRS-00102 YOUBEQADMIN-SRS-00103	COMPLETE
ALERTS	YOUBEQADMIN-SRS-00110	COMPLETE
ALERTS THRESHOLD CONFIGURATION	YOUBEQADMIN-SRS-00111	COMPLETE
ALERTS ADD EMAIL TO NOTIFY	YOUBEQADMIN-SRS-00112	COMPLETE
ALERTS DELETE EMAIL TO NOTIFY	YOUBEQADMIN-SRS-00113	COMPLETE
REAL TIME MONITORING	YOUBEQADMIN-SRS-00130	COMPLETE
STATISTICS REQUEST STATISTICS PARSE STATISTICS SAVE STATISTICS	YOUBEQADMIN-SRS-00140 YOUBEQADMIN-SRS-00141 YOUBEQADMIN-SRS-00142 YOUBEQADMIN-SRS-00143	COMPLETE
USERS STATISTICS	YOUBEQADMIN-SRS-00400	COMPLETE
JOURNAL STATISTICS	YOUBEQADMIN-SRS-00410	COMPLETE
STAMPS STATISTICS	YOUBEQADMIN-SRS-00420	COMPLETE
CHAT STATISTICS	YOUBEQADMIN-SRS-00430	COMPLETE
TELEPORTS STATISTICS	YOUBEQADMIN-SRS-00440	COMPLETE
DEMOGRAPHICS STATISTICS	YOUBEQADMIN-SRS-00450	COMPLETE
USERS LIST STATISTICS	YOUBEQADMIN-SRS-00460	COMPLETE

Table 3: Software Requirements

4.3 Use Cases

In this section we will present youbeQadmin use cases. Figure 8 shows the use case for youbeQadmin.

- **Actors List**
  - The actors of this application are the following:
    - Administrator – Has full access to the youbeQ Admin platform
    - User – Has restricted access to the youbeQ Admin platform
- **youbeQ Administration**
  - We have an actor that is the Administrator, with super user powers and permissions . This actor uses the youbeQ Administration, which in turn uses other modules like authentication and registration, and has modules that extend it, inheriting its attributes and behavior.

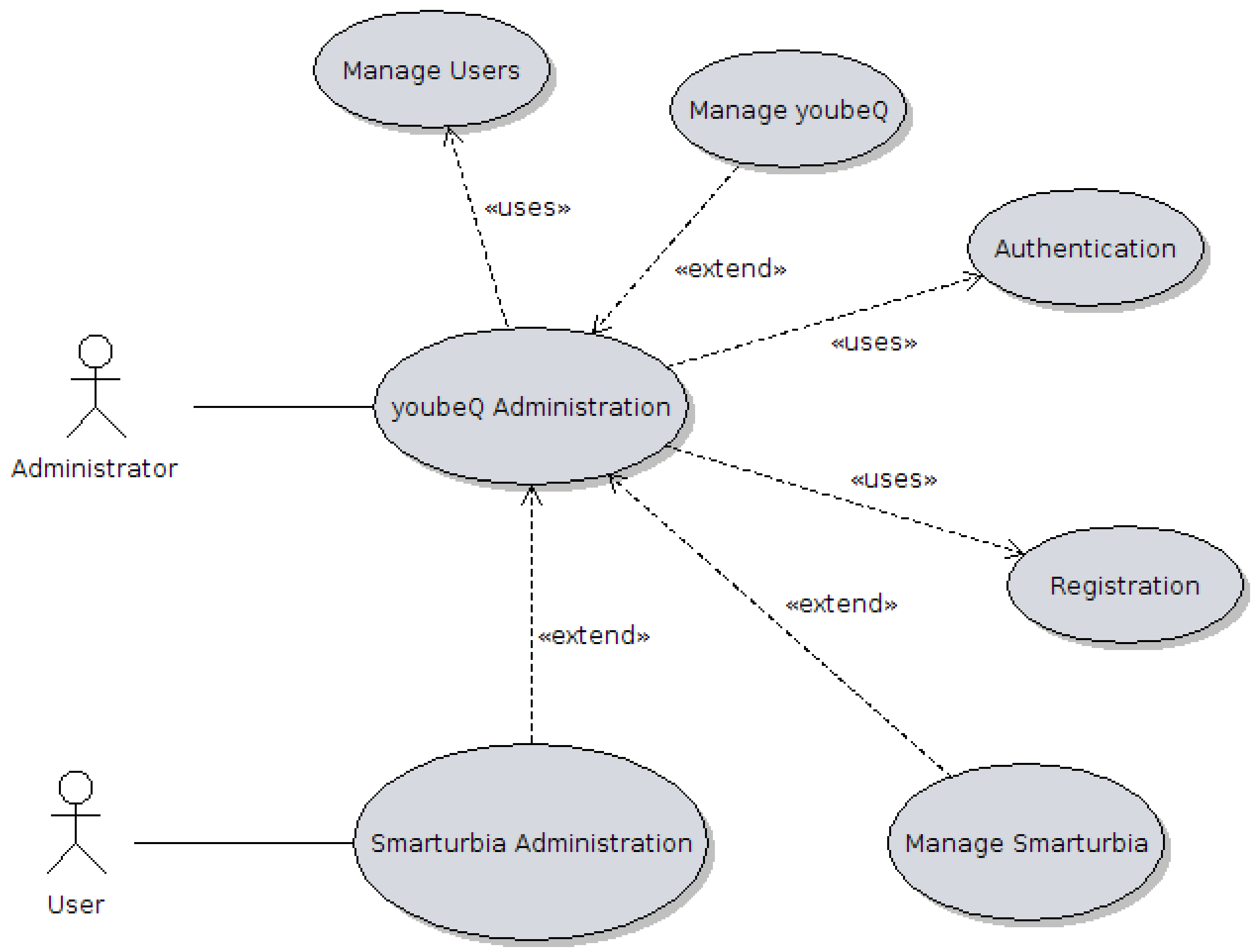


Figure 8: youbeQadmin Use Case (Full View)

- **Smarturbia Administration**
  - The other actor is a normal user with privileges granted by the administrator. It extends the youbeQ administration attributes and behavior but it has limited access to the views it can access and which attributes it can edit.
  - **Registration**
    - The user can choose between normal registration or social network registration (Facebook). He is able to register himself if he isn't already registered, otherwise he will be properly notified. An administrator can only be registered by another administrator using DJANGO's administration site also built in with DJANGO's Framework.
  - **Authentication**
    - The user or administration can / should authenticate themselves in order to keep using youbeQ administration / Smarturbia administration features. This authentication like the registration can also be normal, or social network authentication.
  - **Manage Users**
    - There is an administration interface with the purpose of managing users. The administrator can manage all user profile details, account details and permissions.
  - **Manage Smarturbia**
    - This is a submodule that extends youbeQ Administration. It inherits all the registration, authentication and user management capabilities. Apart from this capabilities Smarturbia can manage APPS (also know has cities in Smarturbia) giving features for APP creation, editing and removal. There are two use cases for this module, Figure 9 for the administrator, and Figure 10 for the normal user.

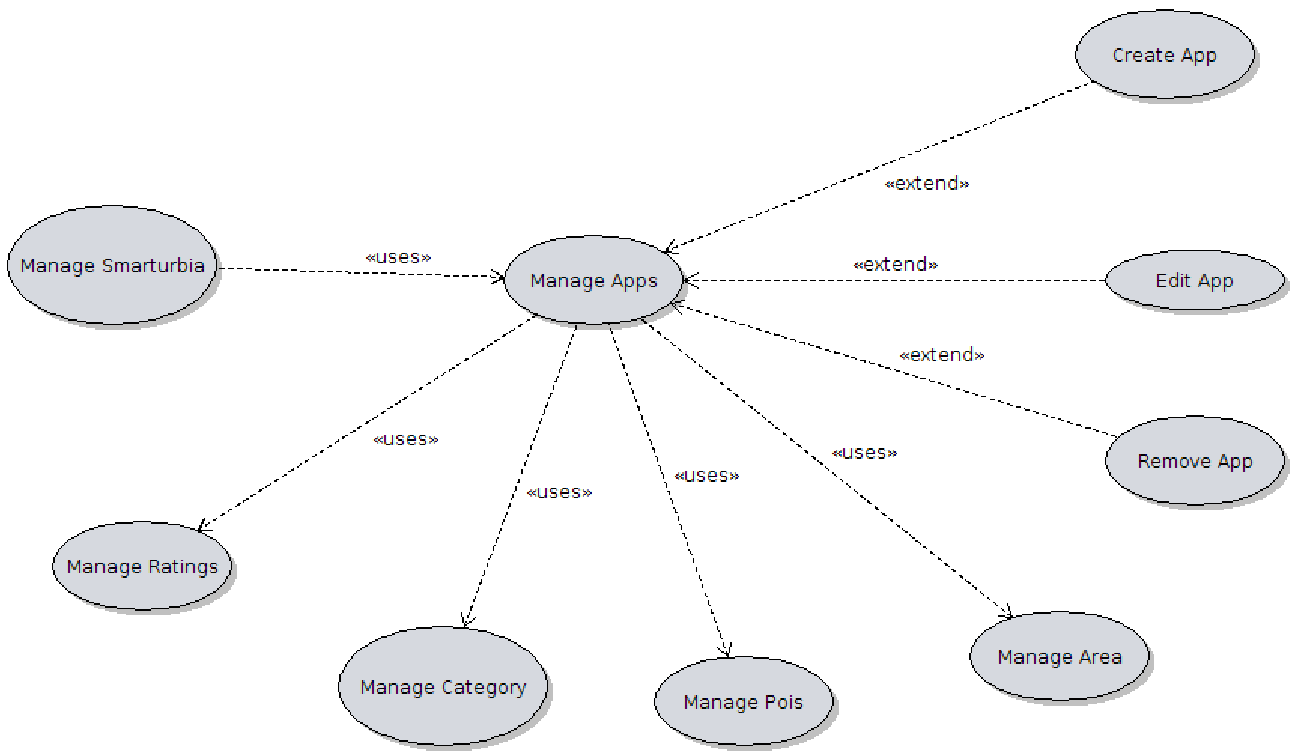


Figure 9: Smarturbia Administrator Use Case

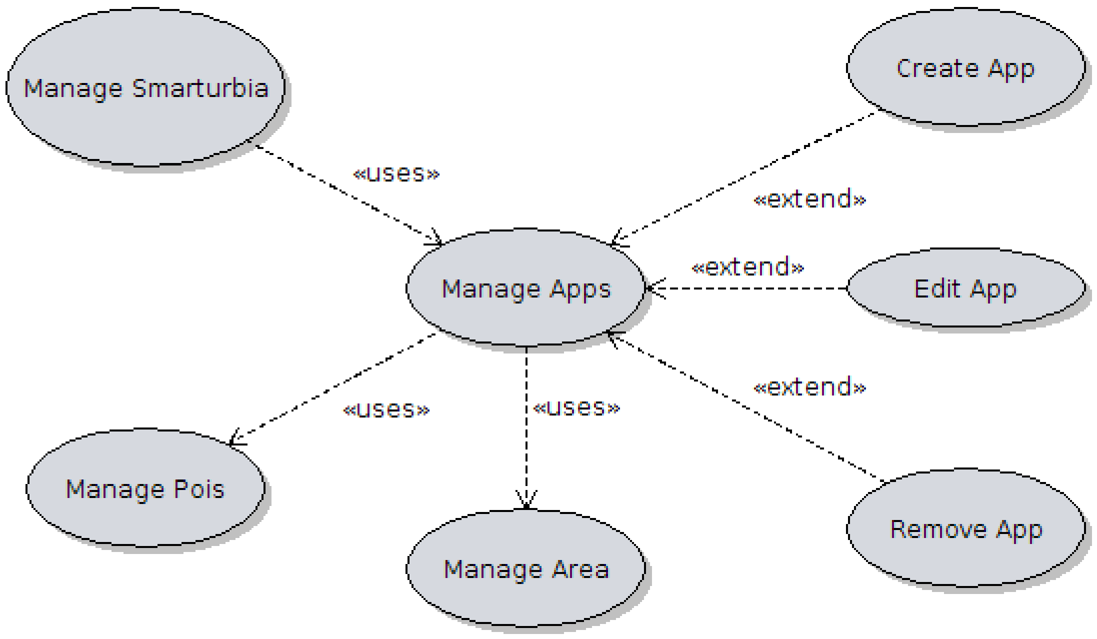


Figure 10: Smarturbia User Use Case

- **Create APP**
  - A user or administrator can create a new APP, proving its details (some of them required), city limits, area name and points of interest.
- **Edit APP**
  - A user or administrator can edit a APP, proving its details (some of them required) or city limits or area name or points of interest. Please note that a normal user can only edit a city that he created.
- **Remove APP**
  - A user or administrator can delete APPS. Please note that a normal user can only delete APPS that he created.
- **Manage Area**
  - A user or administrator can add area limits. Only administrators can edit and remove areas.
- **Manage Category**
  - Administrators can add categories, edit and remove them.
- **Manage POIS**
  - A user or administrator can add, edit and remove points of interest, when creating or editing an APP.
- **Manage Ratings**
  - A user or administrator can add ratings, but only administrators can edit ratings of APPS.

Manage youbeQ

- This is a submodule that extends youbeQ Administration. It inherits all the registration, authentication and user management capabilities.
- **Manage Publicity**
  - **Add Publicity Area**
    - The administrator can create a new publicity, providing its details (some of them required), limits and area.
  - **Edit Publicity Area**
    - The administrator can edit a publicity, providing its details (some of them required) or limits or area.

- **Remove Publicity Area**
    - The administrator can remove a given publicity.
- **Purchase Publicity Area**
    - The user can purchase/rent a given publicity area.

- **Statistics**
    - youbeQ Administration provides us with statistics for youbeQ's Messages, Journal, Places , Registrations, Login, Teleports, Data Visualization (2D and 3D) and filter content by Data, Date Interval, Gender, Age and Age Interval. It also gives statistics for the Smarturbia registered users, last login, average APPS per user.
  - **Server Management Statistics**
    - Server Management Statistics extends the statistics module and is specifically designed to provide statistics about the servers. We can also add servers, edit and remove them inside this module.
  - **Statistics Management Add Server**
    - Statistics Management Add Server extends the Server Management Statistics module and is specifically designed to add new Servers in order to get their statistics.
  - **Statistics Management Edit Server**
    - Statistics Management Edit Server extends the Server Management Statistics module and is specifically designed to edit Servers in order to get their statistics.
  - **Statistics Management Remove Server**
    - Statistics Management Remove Server extends the Server Management Statistics module and is specifically designed to remove Servers in order to get their statistics.
  - **Notifications**
    - In order to get the statistical information and alerts, with need this notification modules that will manage all content related to providing statistical information.

- **Real Time Monitoring**
  - This module is dedicated to providing information about memory usage, CPU usage and online users. It shall be used in conjunction with the statistical and notification modules.

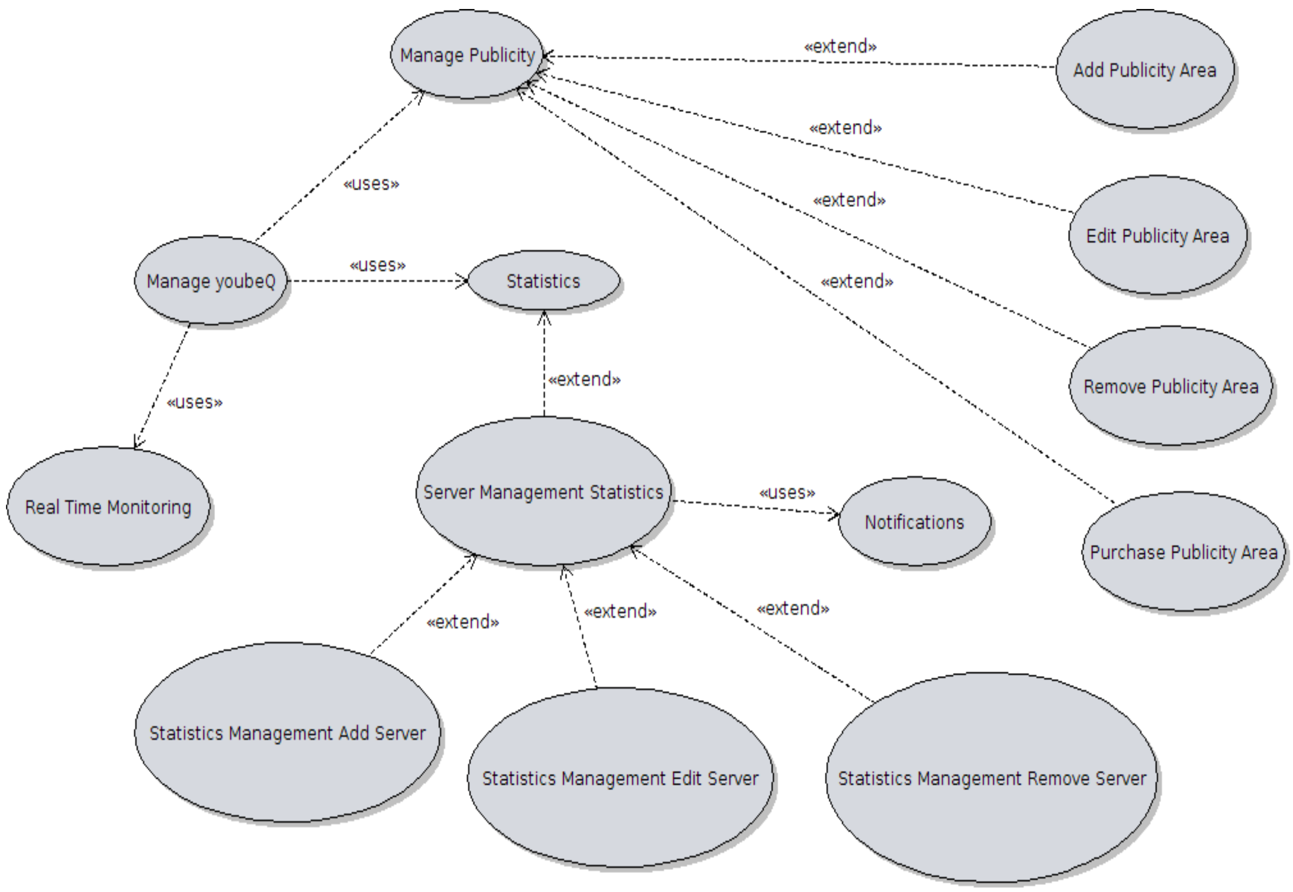


Figure 11: youbeQ Use Case Module

4.4 Architecture

All the annexes for this chapter can be found at "*Annex C - Software Architecture"*.

The managing platform will be divided into two different modules, youbeQ and Smarturbia.

Its concept centers around the idea of having a web based application that manages all of youbeQ and Smarturbia content's enforcing a distributed architecture while allowing the integration of this two modules/applications.

Nevertheless at the time the company decided to still keep their already existing applications re-enforcing the need for integrating the new platforms with the old ones.

Some of its contents have been migrated to the new platform others were simply redesigned and upgraded.

DJANGO uses ORM, which basically means that our programmed classes will be directly and automatically mapped to our database entities.
This brings great comfort to the programmer because he doesn't have to worry

about all the “low level” connections to the database and database access.

We simply instantiate an object from a model class already mapped to our SQL database, and we can immediately start filling its attributes or making queries.

4.4.1 Component Diagram

This is the component diagram, and represents the full system. It shows the REST interfaces for the communication between the modules as well as the MySQL connectors for the communication of certain modules with their databases.

In figure 12 the blue color represents the full implementations created during the internship, the green modules represent the modules that where changed during the internship period.

Finally the orange color represents modules that where simply used to achieve certain purposes and were not internally modified.

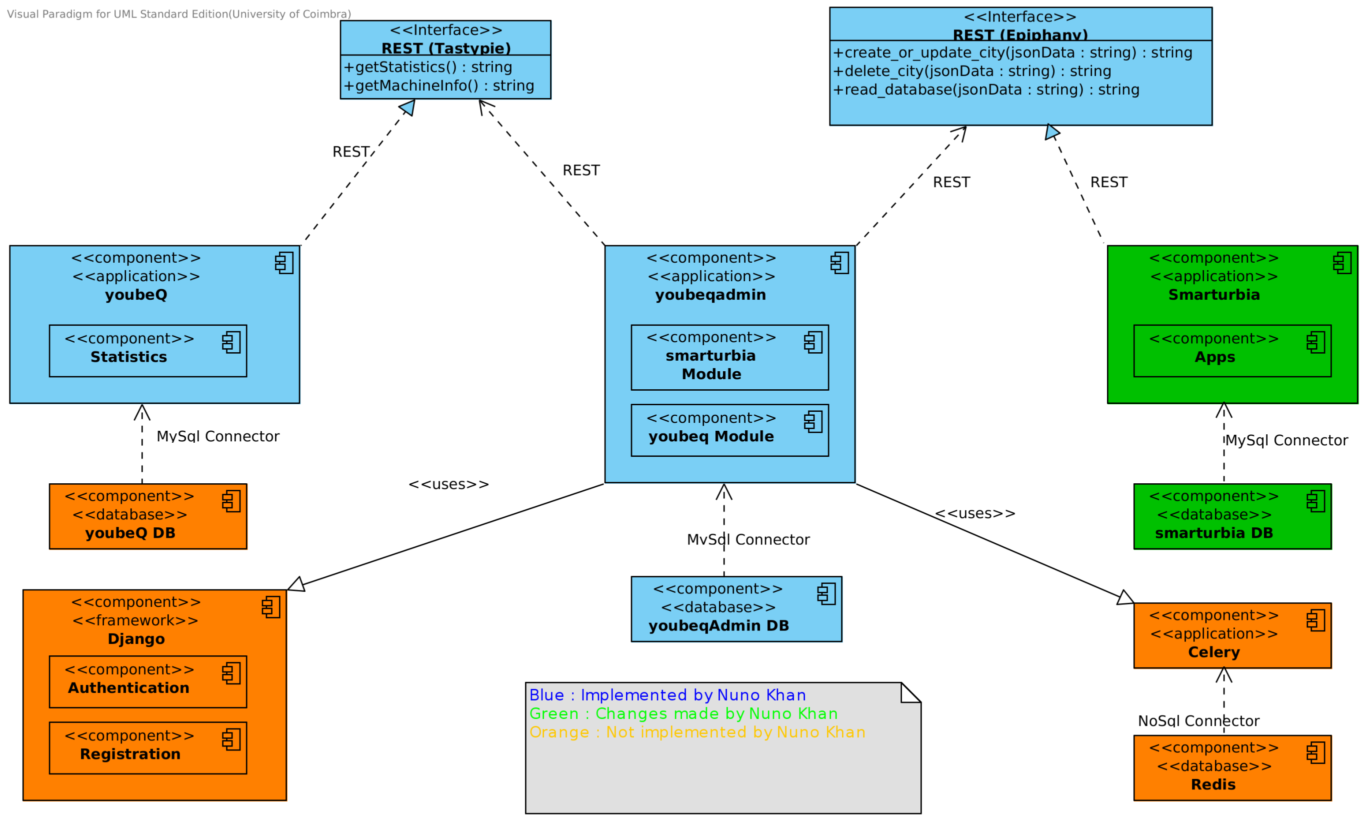

Visual Paradigm for UML Standard Edition(University of Coimbra)
<<Interface>> REST (Tastypie)
+getStatistics() : string
+getMachineInfo() : string
<<Interface>> REST (Epiphany)
+create_or_update_city(jsonData : string) : string
+delete_city(jsonData : string) : string
+read_database(jsonData : string) : string
REST
REST
REST
REST
<<component>> <<application>> youbeQ
<<component>> Statistics
<<component>> <<application>> youbeqadmin
<<component>> smarturbia Module
<<component>> youbeq Module
<<component>> <<application>> Smarturbia
<<component>> Apps
MySql Connector
<<component>> <<database>> youbeQ DB
<<uses>>
MySql Connector
<<uses>>
MySql Connector
<<component>> <<database>> smarturbia DB
<<component>> <<framework>> Django
<<component>> Authentication
<<component>> Registration
<<component>> <<database>> youbeqAdmin DB
<<component>> <<application>> Celery
NoSql Connector
<<component>> <<database>> Redis
Blue : Implemented by Nuno Khan
Green : Changes made by Nuno Khan
Orange : Not implemented by Nuno Khan

Figure 12: Component Diagram

In order to take full advantage of DJANGO's concepts, we integrate our code with third party DJANGO modules.

Although DJANGO is a complete and stable framework, its users have the ability to install third party submodules also called “reusable APPS”. This submodules allow us to have, in the majority of the time, a well written and tested reusable code where we can just “plug” into our existing project and we can immediately start using it. Nevertheless we can always implement our own code if we wish to or even modify open source third party code.

Below in table 4 is the list of dependencies for this project. All the details for each module are in Annex C - Software Architecture.

Framework Modules						
DJANGO-registration 0.8	DJANGO-tastypie 0.9.11	DJANGO-celery	South 0.7.5	DJANGO-social-auth 0.7.9	DJANGO_ extensions 0.9	Pillow 1.7.7

Table 4: DJANGO Framework Modules

4.4.2 youbeQ Module Architecture

The main task of the youbeQ module is to get the statistical information from the main youbeQ application through a RESTFULL API.

The youbeQ application has a DJANGO written module that will provide the RESTFULL API resources to be consumed only by the youbeQ module from the youbeQAdmin project.

This DJANGO module called youbeQ_django sole purpose is to compute in real time the statistical information provided by the software functional requirements, and supply this information as a resource in the RESTFULL API to be consumed, as well as keeping track of the historic data of previous requests in a local database.

The youbeQ module will also give the ability to add specific machines for statistical monitoring as well as other requirements specified in "*Annex B – Software Requirements*".

The other task of the youbeQ module is to be responsible for the user area purchasing, including the publicity management (publicity creation, edition and deletion) over the purchased area. The areas are created only by the administrators.
Each area can have many publicities purchased by different users.

- **Activity Diagram**
  - **YoubeQ Module (Full View)**
    - This activity diagram in Figure 13 shows the youbeQ module architecture. From the start point we have four available options (publicity and statistics). Each one of this options have their own sequence diagram (see annex C) as this is only a full overview of the system. After that we go back to our main options chooser or we terminate by reaching the final state.

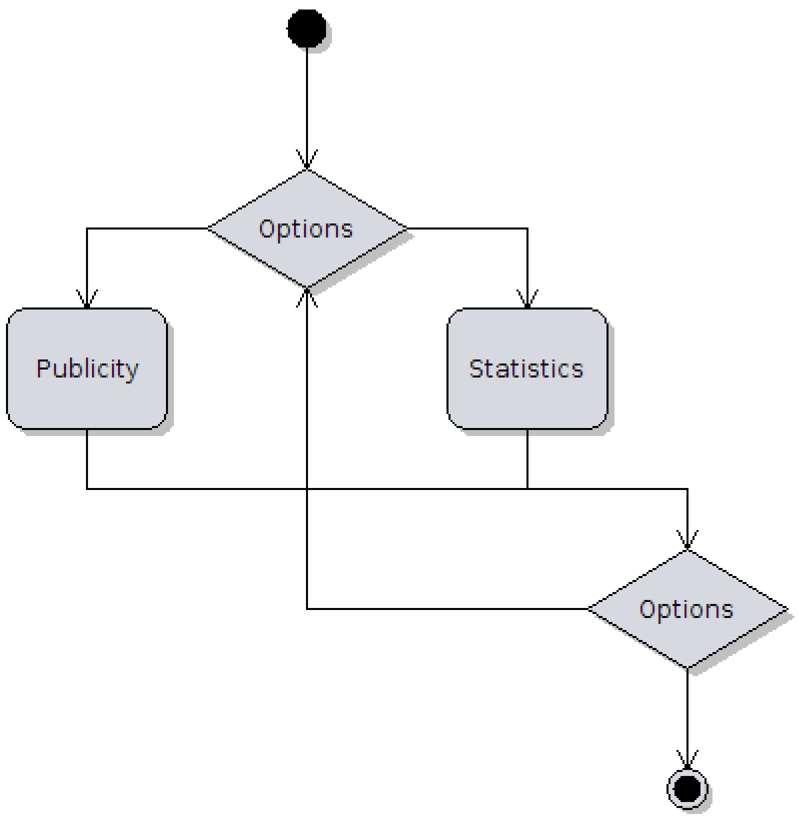


Figure 13: youbeQ Module (Full View)

- **Class Diagram**
  - This class diagram in Figure 14 represents the ORM abstraction for youbeQ module. It also shows the existing relationships between the classe entities before being mapped into tables.

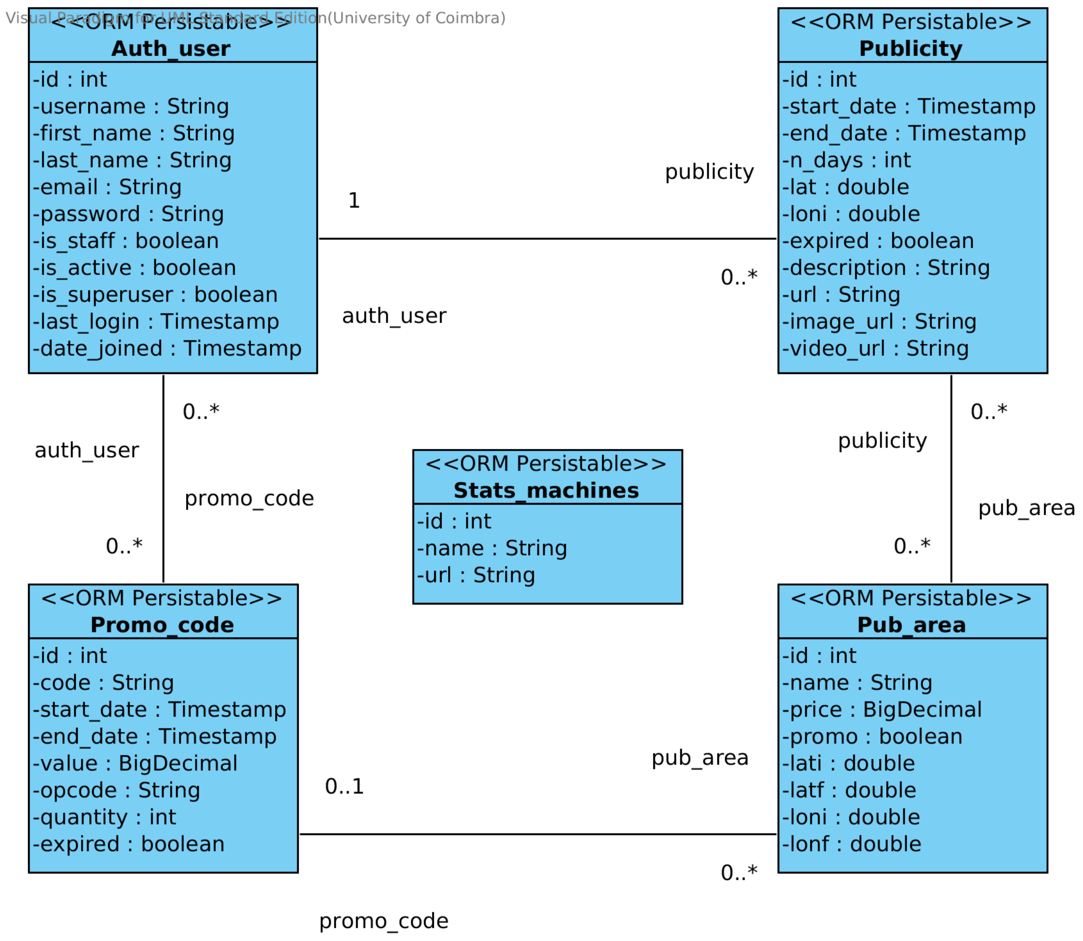


Figure 14: youbeQ Class Diagram (Partial View)

4.4.3 Smarturbia Module Architecture

The Smarturbia module is responsible for the creation of new APPS (cities), new categories for the APPS, and rating/commenting of the APPS. It also allows the user to edit the APPS and remove them, as well as editing the rating and comments for the APPS.
Note that only the administrators have permission to create new categories.

Every change occurring in the Smarturbia module will also be replicated to the Smarturbia APP server in order to become visible to the users. This is also done using a RESTFULL API, communicating through JSON messages.

The replication is accomplished using Celery which is "...an asynchronous task queue/job queue based on distributed message passing" [9]. This way we ensure that all the replication is done asynchronously, allowing the user to continue using the management platform simultaneously.

This replication clearly asks for a FIFO implementation in order for the data to be consistent in both databases.

- **Activity Diagram**
  - Inside the Smarturbia module we find different options that aren't related between themselves. After choosing any of the available options (with the exception of "Try APP") the page gets redirected back to the main Smarturbia page.
  - If we choose "Try APP" instead, we get redirected to a different server that will contain the information we created or edited so that we can test them.
  - This activity diagram below in Figure 15 is only the administrator, for a normal user it becomes slightly different and will also be presented.

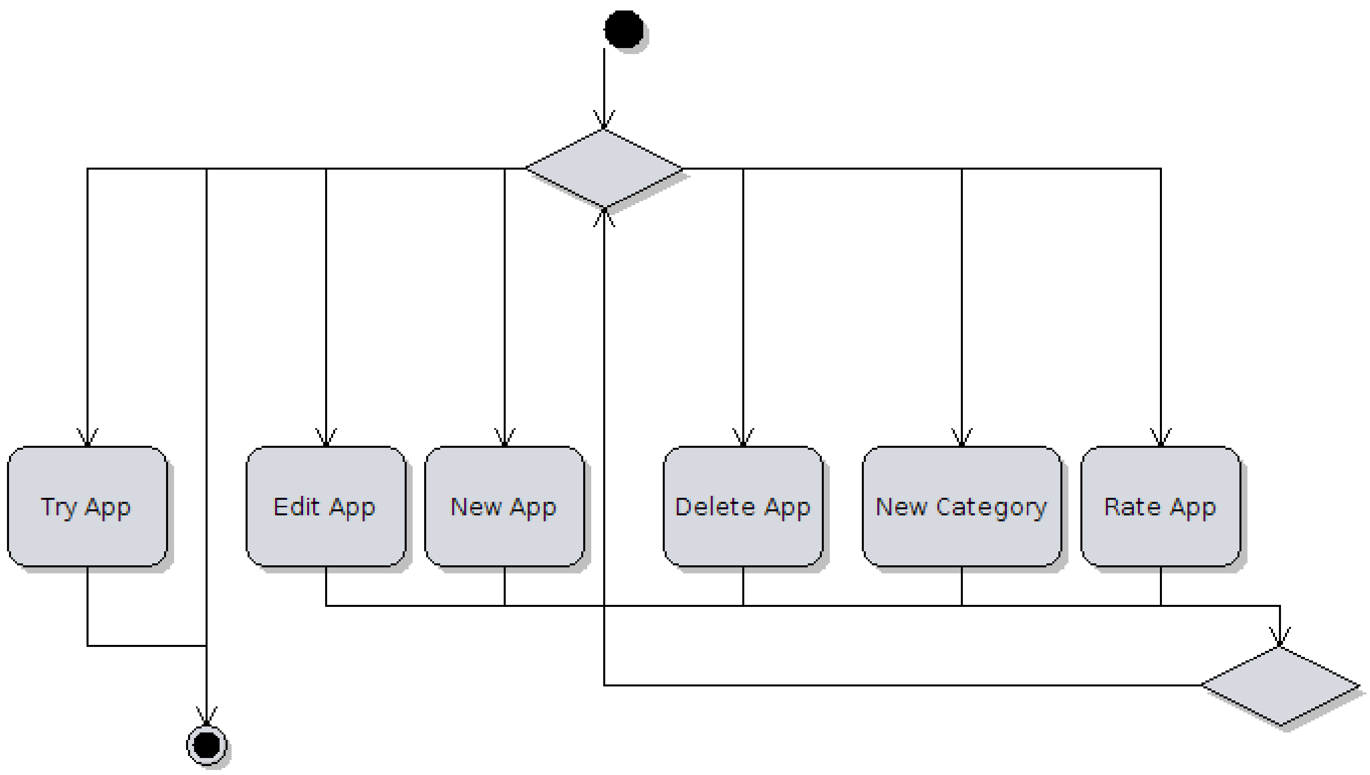


Figure 15: Smarturbia Activity Diagram

The following activity diagram in Figure 16 will be for a normal user without superuser permissions.

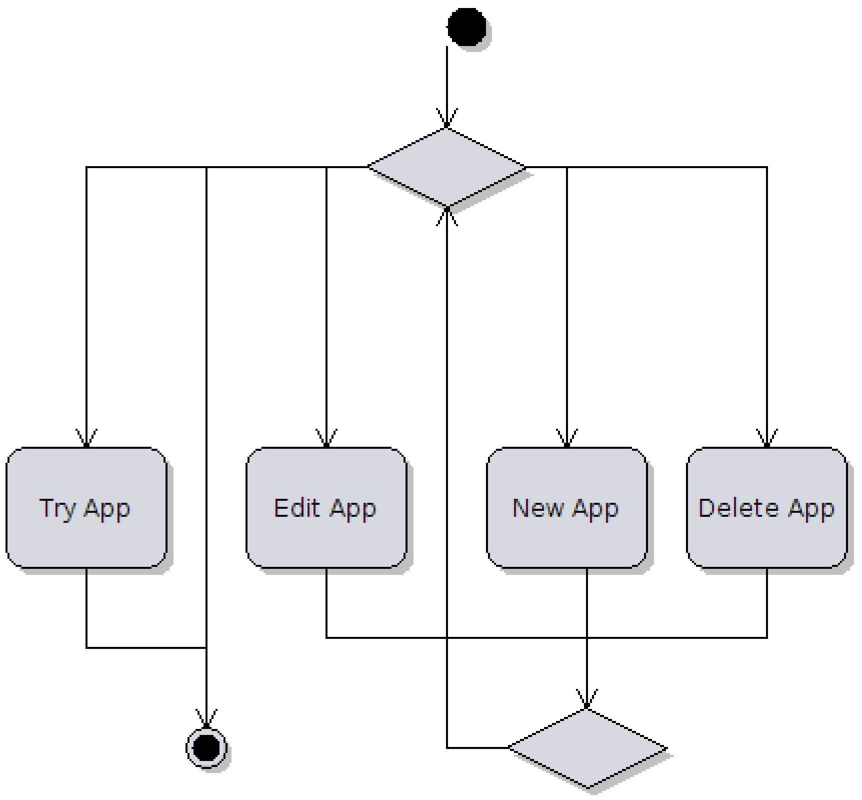


Figure 16: Smarturbia Activity Diagram

- **Class Diagram**
  - This class diagram in Figure 17 represents the ORM abstraction for Smarturbia module. It also shows the existing relationships between the classe entities before being mapped into tables.

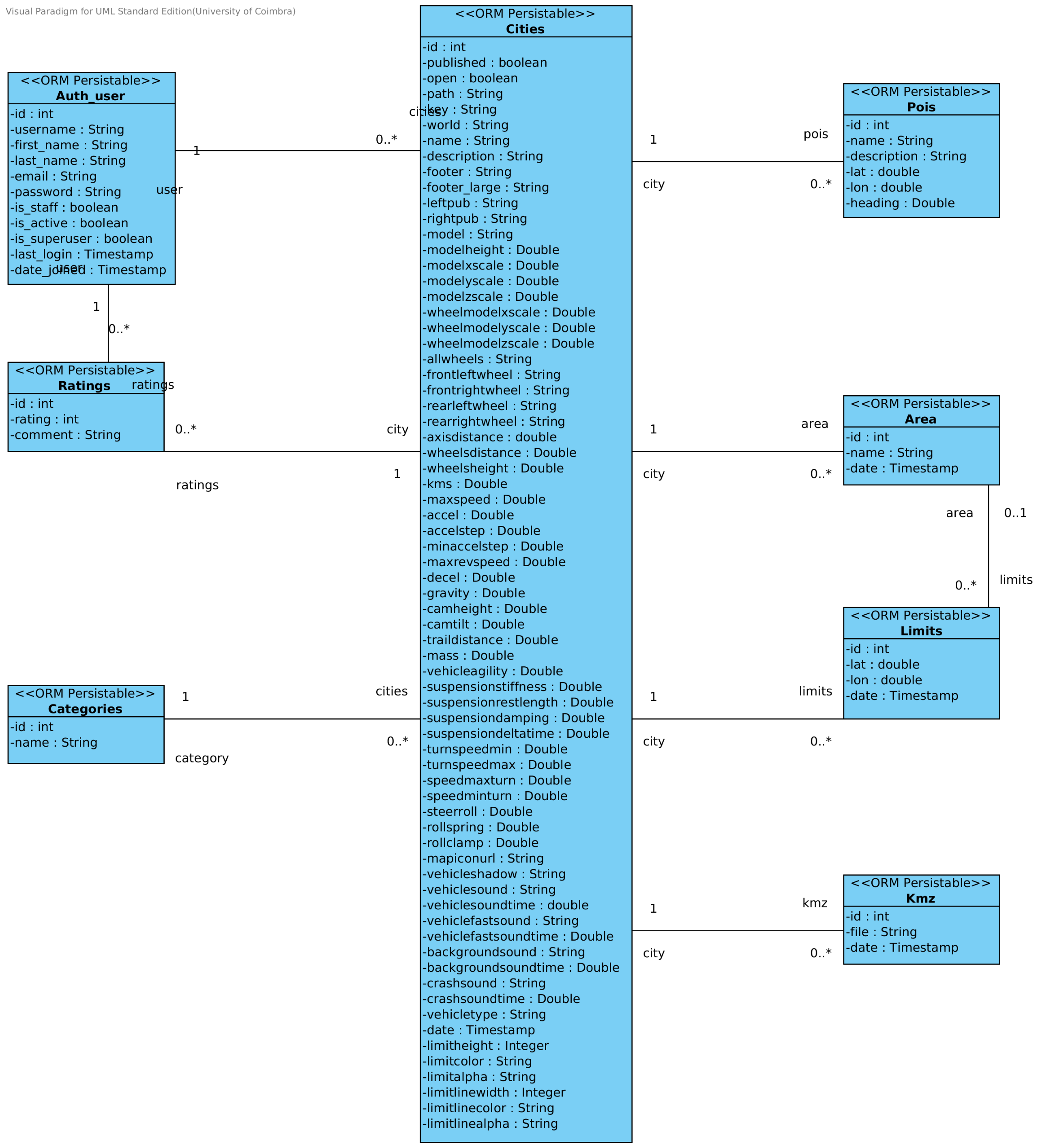


Figure 17: Smarturbia Class Diagram (Partial View)

4.4.4 Tasks

For the execution of certain tasks, like for example data replication to remote servers, or database updates and so on, we decided to use Celery [9] which is a module that can be easily integrated with DJANGO in order to manage tasks, similar to a CRON job application.

We are also using REDIS that is our Celery broker (temporary database acting similarly to a cache database).

- **Celery**
  - The main advantage of using CRON jobs (Celery) is that the appointed tasks are executed by an independent DEAMON that can actually run on a independent server and as part of a cluster of DEAMONS executing individual or collective tasks. This brings great advantage when we decide to scale our applications, and we must notice that Celery requires very little configuration (at a basic level) in order to get things up and running and integrating with our application. Celery workers execute their given tasks at a specific time and order (decided by the default values or overridden by the developer).

- **REDIS**
  - REDIS is a NO-SQL key/value high performance database similar to a hash-table, in other words we can retrieve the value by providing the key associated to the value.
  - Celery requires a broker in order to store its workers, and assigned tasks. The broker choice can be REDIS, RABBITMQ, COUCHDB, MySQL Db or any other database supported by Celery. Each one of them has their own particularities and the choice should be based upon those particularities and their integration with Celery.

5 Risk Analysis

In this section we present a table associated with the risk analysis, where we present the problems we may encounter and possible solutions.

Problems	Solutions
Not knowing the used framework	Study and explore used technologies and tools
Not knowing the programming languages	Study and explore used technologies and tools
Not knowing the communication technologies such as REST, SOAP, JSON, XML, etc	Study and explore used technologies and tools
Limited control over changes on remote databases	No solution
Main development on remote applications made by others and poorly documented	Study and explore , build prototypes
Strong dependency on Google's API in order for application to fully work	Use another API
Possible learning and usage of automated testing tools	Study and explore used technologies and tools , document evolution in order to see what can be done in a specific amount of time

Table 5: Risk Analysis Problems and Solutions

When developing youbeQadmin some of this risks appeared. This risks where:

- **Not knowing the programming languages**
  - PHP and JAVASCRIPT were fairly new to me, so I had to learn the basics
- **Main development on remote applications made by others and poorly documented**
  - Smarturbia had undocumented and uncommented code, so I had to study and explore the code on my own.
- **Possible learning and usage of automated testing tools**
  - I had never done proper automated testing, so this was a new experience and I had to read a lot of documentation to understand how the testing frameworks work.

6 Achievements

6.1 youbeQadmin (BETA version)

Before the end of the Internship the youbeQadmin application and modules developed were deployed into production. In this section there are some print-screens of the web pages of the final product.

To avoid confusion to the user because of the name, the logo was renamed from youbeQadmin to Smarturbia Admin, the rest remains the same.

Figure 18 is the main Login Page for the youbeQadmin from the Smarturbia perspective where the user or administrator authenticate themselves with valid credentials.

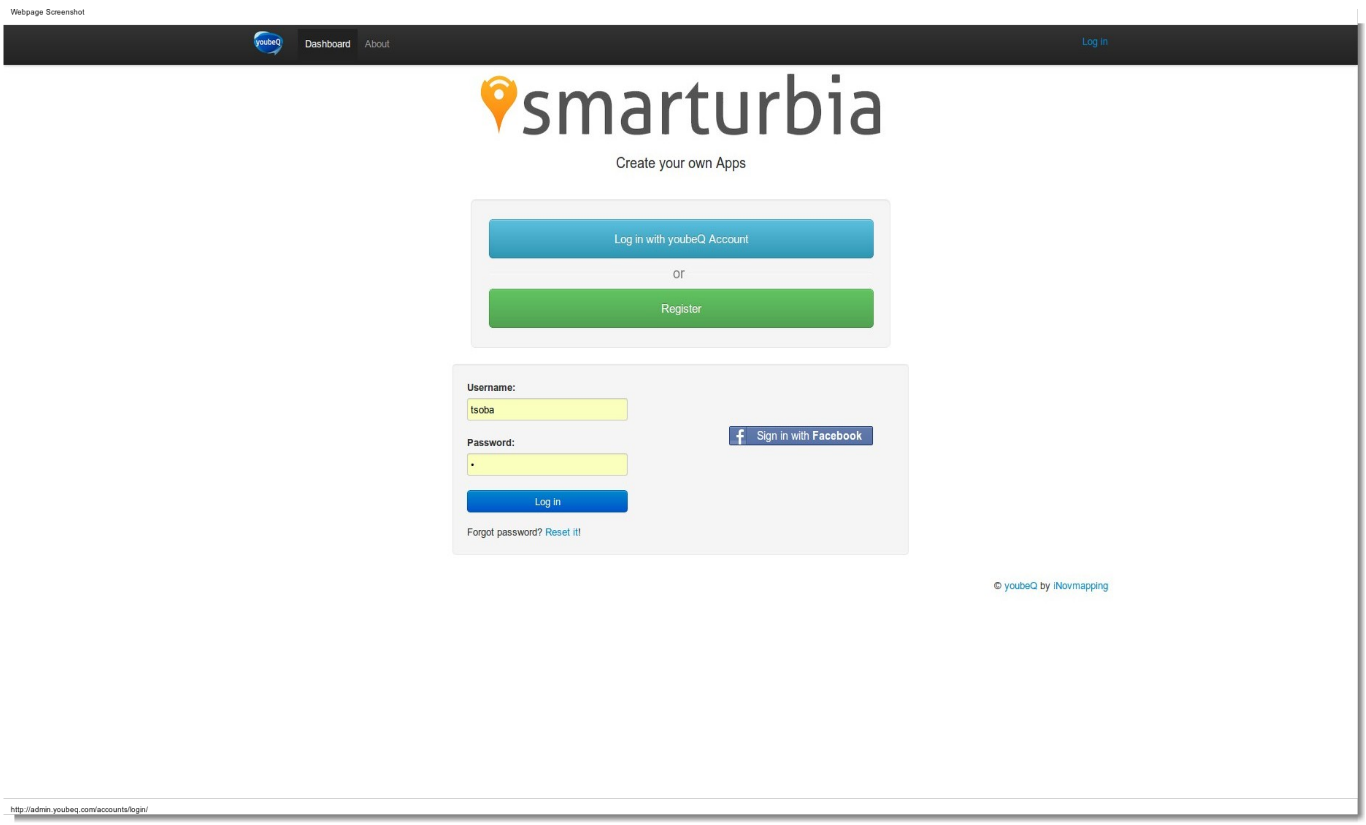


Figure 18: Login Page

Once authenticated the main page in Figure 19 comes into place, with all the available modules / services that can be managed.

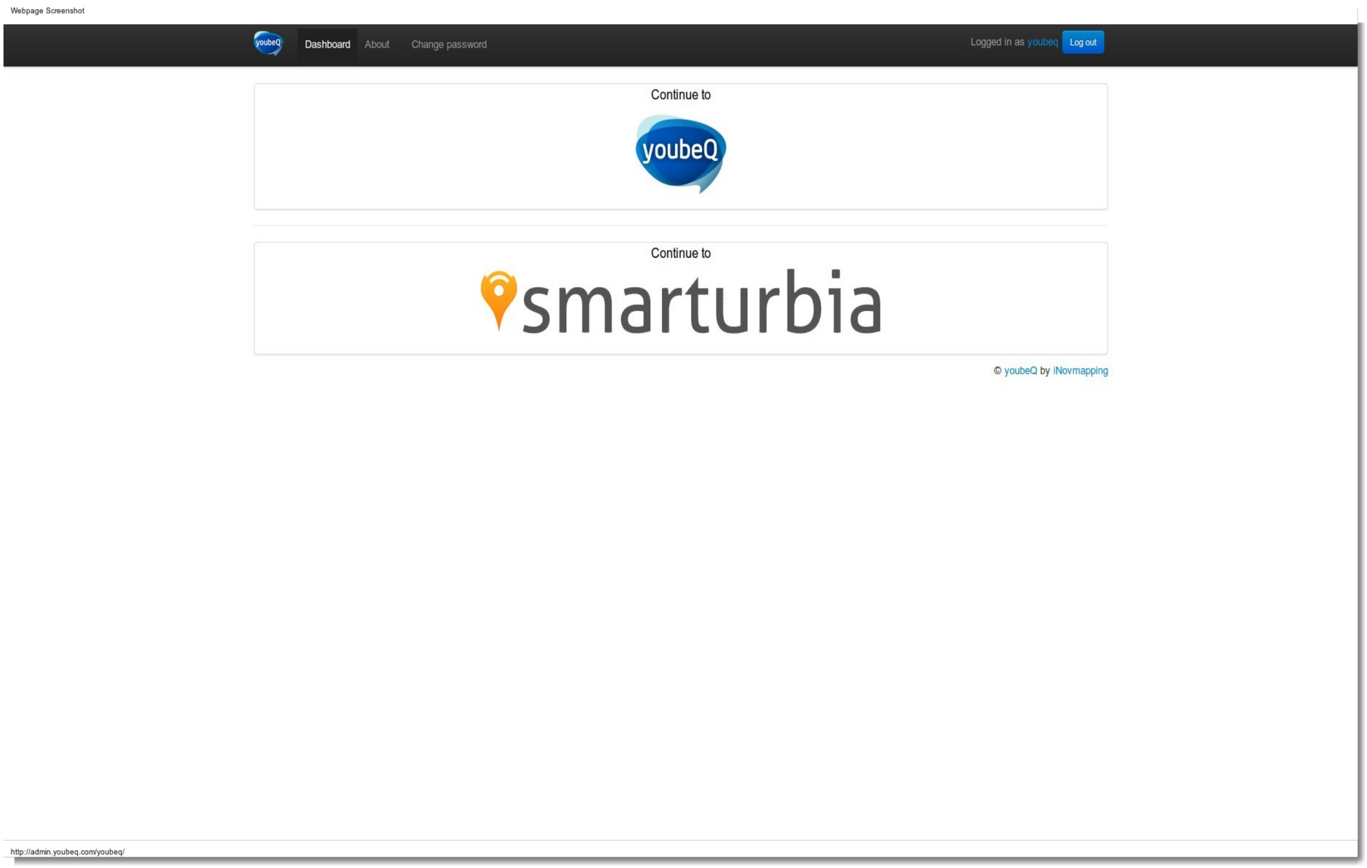


Figure 19: Main Page

Figure 20 is the statistics module, where we can see all statistics related to iNovmapping's products (for now only youbeQ and Smarturbia are available).

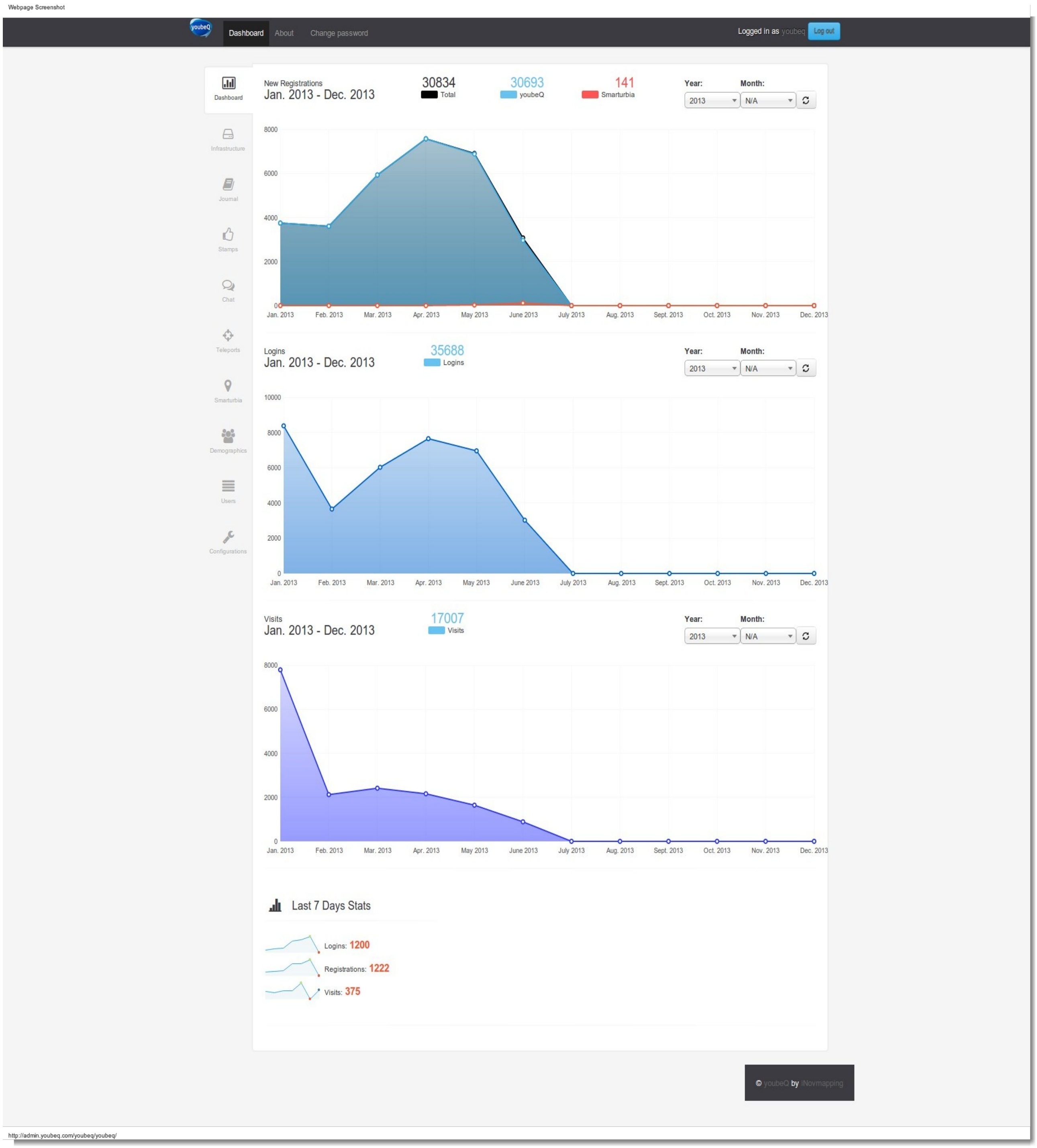
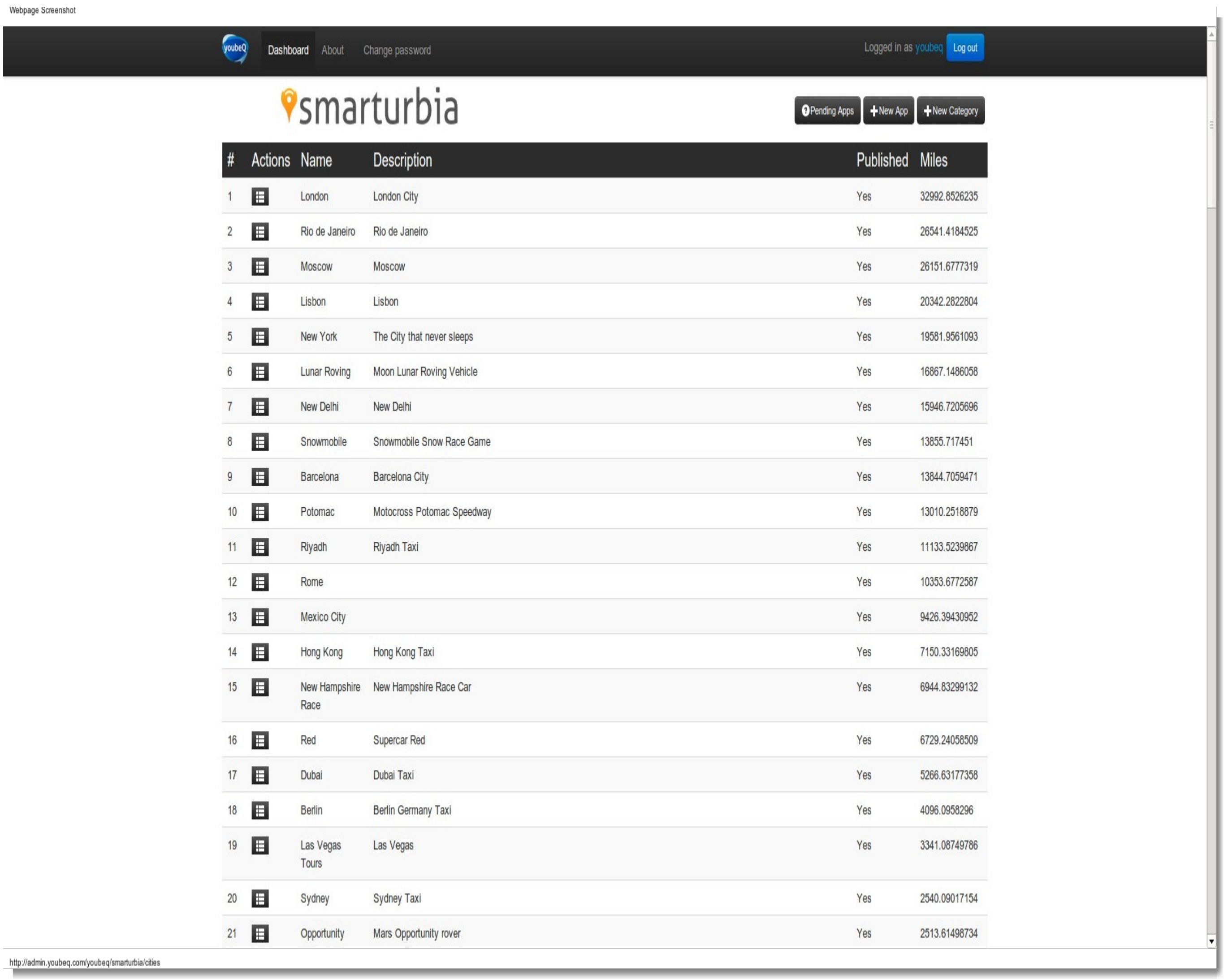


Figure 20: youbeQadmin statistics for the administrator of youbeQ and Smarturbia

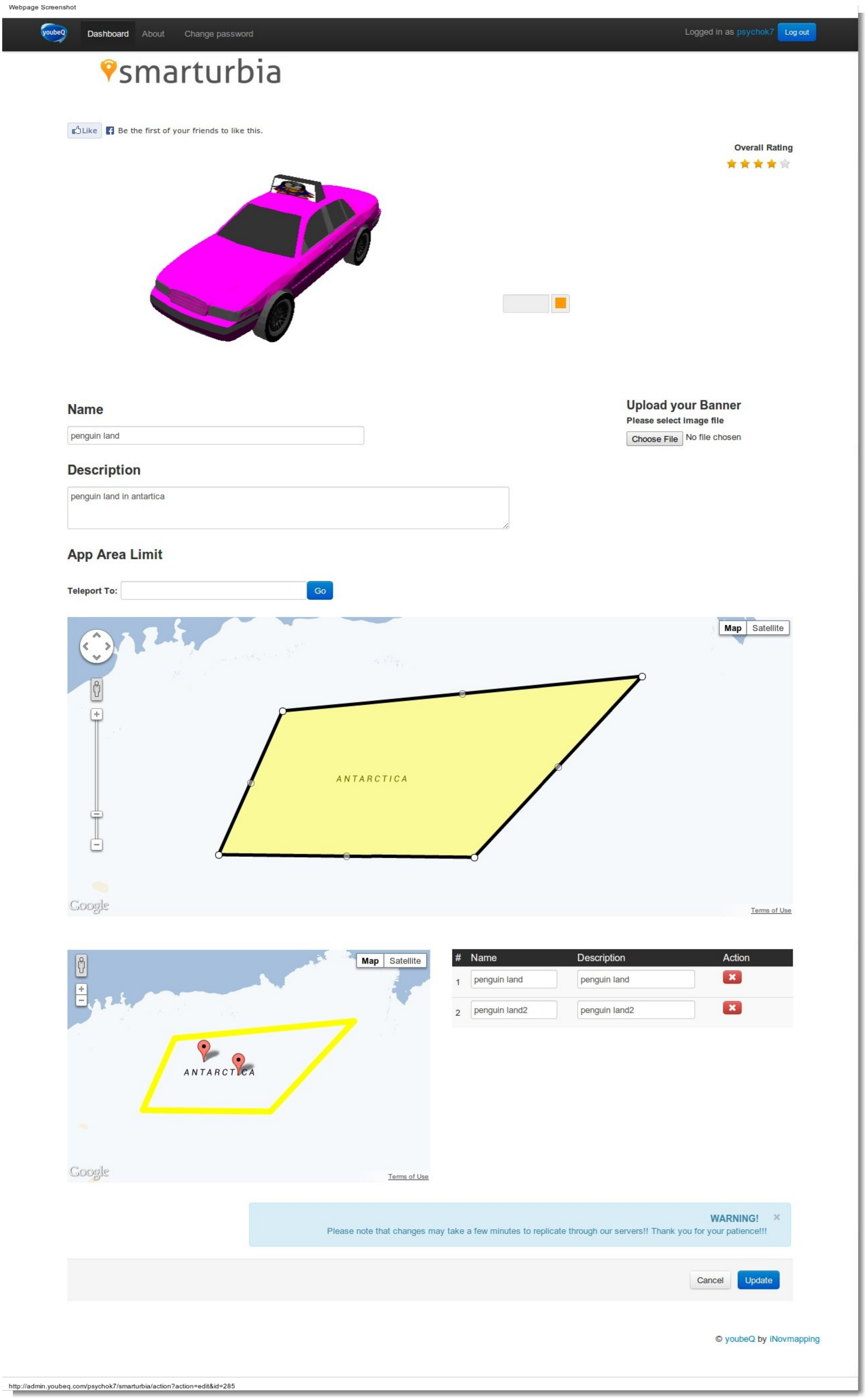
In Figure 21 we have a list of all available APPS/cities (created by the user or admin) in Smarturbia module, where we have the power to delete, edit, try, rate, and create new ones.



#	Actions	Name	Description	Published	Miles
1		London	London City	Yes	32992.8526235
2		Rio de Janeiro	Rio de Janeiro	Yes	26541.4184525
3		Moscow	Moscow	Yes	26151.6777319
4		Lisbon	Lisbon	Yes	20342.2822804
5		New York	The City that never sleeps	Yes	19581.9561093
6		Lunar Roving	Moon Lunar Roving Vehicle	Yes	16867.1486058
7		New Delhi	New Delhi	Yes	15946.7205696
8		Snowmobile	Snowmobile Snow Race Game	Yes	13855.717451
9		Barcelona	Barcelona City	Yes	13844.7059471
10		Potomac	Motocross Potomac Speedway	Yes	13010.2518879
11		Riyadh	Riyadh Taxi	Yes	11133.5239867
12		Rome		Yes	10353.6772587
13		Mexico City		Yes	9426.39430952
14		Hong Kong	Hong Kong Taxi	Yes	7150.33169805
15		New Hampshire Race	New Hampshire Race Car	Yes	6944.83299132
16		Red	Supercar Red	Yes	6729.24058509
17		Dubai	Dubai Taxi	Yes	5266.63177358
18		Berlin	Berlin Germany Taxi	Yes	4096.0958296
19		Las Vegas Tours	Las Vegas	Yes	3341.08749786
20		Sydney	Sydney Taxi	Yes	2540.09017154
21		Opportunity	Mars Opportunity rover	Yes	2513.61498734

Figure 21: Smarturbia APPS

When editing or creating an APP/cities Figure 22 is the screen that we go into



Name

penguin land

Description

penguin land in antartica

Upload your Banner

Please select image file

Choose File No file chosen

App Area Limit

Teleport To: Go

#	Name	Description	Action
1	penguin land	penguin land	✖
2	penguin land2	penguin land2	✖

WARNING!
Please note that changes may take a few minutes to replicate through our servers!! Thank you for your patience!!!

© youbeQ by iNovmapping

http://admin.youbeq.com/psychok7/smarturbia/action?action=edit&id=285

Figure 22: Smarturbia APP Creation/Editing

7 Tests

This section presents the tests results for the youbeQadmin project.

DJANGO comes with a unit testing framework for an easy to use automated testing. We can use a collection of tests (test suite) to solve or avoid a number of problems.

The tests tried to cover most of the functionalities. There were some aspects that weren't fully tested, mainly the exception handling.

Others were tested while developing and they are no automated testing, like to example to test the if when creating an APP in Smarturbia module, if it replicates properly to the Smarturbia server. There were no automated tests for this scenario because of the time it would involve to develop such complex scripts.

There are some aspects of the application that still need to be fixed but a BETA version is already in production for users to use it.

There where also some performance testing (load balancing) and usability testing.

The complete test case specification and results can be found in *"Annex D - Software Testing"*.

The server specifications and all the rest of the performance information can also be found in this same annex.

The results for the usability testing can be found at *"Annex E - Usability Tests"*.

Below we have a traceability Matrix with the software requirement(s), and the corresponding test(s).

7.1 Traceability Matrix for Unit Testing

All test details can be found at *"Annex D - Software Testing".* The following table is the traceability matrix.

Requirement	Test Item(s)
YOUBEQADMIN-SRS-00010-REGISTRATION	YOUBEQADMIN-TCS-00010
YOUBEQADMIN-SRS-00020-REGISTRATION ERROR HANDLING	YOUBEQADMIN-TCS-00020 YOUBEQADMIN-TCS-00021
YOUBEQADMIN-SRS-00030-AUTHENTICATION	YOUBEQADMIN-TCS-00030 YOUBEQADMIN-TCS-00031
YOUBEQADMIN-SRS-00040-AUTHENTICATION ERROR HANDLING	YOUBEQADMIN-TCS-00040 YOUBEQADMIN-TCS-00041
YOUBEQADMIN-SRS-00050-USER MANAGEMENT	YOUBEQADMIN-TCS-00051 YOUBEQADMIN-TCS-00052 YOUBEQADMIN-TCS-00053
YOUBEQADMIN-SRS-00060-APPLICATION MANAGEMENT YOUBEQADMIN-SRS-00061-APPLICATION MANAGEMENT	YOUBEQADMIN-TCS-00060
YOUBEQADMIN-SRS-00150-APP MANAGEMENT YOUBEQADMIN-SRS-00151-CREATE APP YOUBEQADMIN-SRS-00152-EDIT APP YOUBEQADMIN-SRS-00153-REMOVE APP YOUBEQADMIN-SRS-00154-TRY APP	YOUBEQADMIN-TCS-00151 YOUBEQADMIN-TCS-00152 YOUBEQADMIN-TCS-00153 YOUBEQADMIN-TCS-00154
YOUBEQADMIN-SRS-00231-USER ADD RATING	YOUBEQADMIN-TCS-00231
YOUBEQADMIN-SRS-00232-USER EDIT RATING	YOUBEQADMIN-TCS-00232
YOUBEQADMIN-SRS-00110-STATISTICS MANAGEMENT ALERTS	YOUBEQADMIN-TCS-00110
YOUBEQADMIN-SRS-00111-STATISTICS MANAGEMENT ALERTS THRESHOLD CONFIGURATION	YOUBEQADMIN-TCS-00111
YOUBEQADMIN-SRS-00112-STATISTICS MANAGEMENT ALERTS ADD EMAIL TO NOTIFY	YOUBEQADMIN-TCS-00112
YOUBEQADMIN-SRS-00113-STATISTICS MANAGEMENT ALERTS DELETE EMAIL TO NOTIFY	YOUBEQADMIN-TCS-00113
YOUBEQADMIN-SRS-00130-REAL TIME	YOUBEQADMIN-TCS-00130

MONITORING	
YOUBEQADMIN-SRS-00140-STATISTICS YOUBEQADMIN-SRS-00141-REQUEST STATISTICS YOUBEQADMIN-SRS-00142-PARSE STATISTICS YOUBEQADMIN-SRS-00143-SAVE STATISTICS	YOUBEQADMIN-TCS-00140
YOUBEQADMIN-SRS-00400-USERS STATISTICS	YOUBEQADMIN-TCS-00400
YOUBEQADMIN-SRS-00410-JOURNAL STATISTICS	YOUBEQADMIN-TCS-00410
YOUBEQADMIN-SRS-00420-STAMPS STATISTICS	YOUBEQADMIN-TCS-00420
YOUBEQADMIN-SRS-00430-CHAT STATISTICS	YOUBEQADMIN-TCS-00430
YOUBEQADMIN-SRS-00440-TELEPORTS STATISTICS	YOUBEQADMIN-TCS-00440
YOUBEQADMIN-SRS-00450-DEMOGRAPHICS STATISTICS	YOUBEQADMIN-TCS-00450
YOUBEQADMIN-SRS-00460-USERS LIST STATISTICS	YOUBEQADMIN-TCS-00460

Table 6: Traceability Matrix

7.2 Performance Testing

In order to test the performance of the developed Web Application, some load balancing tests were made [10].

The complete performance test server specification and results can be found in *"Annex D - Software Testing"*.

The tests are actually comprised of two requests, one GET request to fetch the login.html page (this is done because all POST views are CSRF protected) followed by a POST request which is actually the login.

But the relevant results [11] are only for the POST request, where there is an interaction with the database for the login of the user.

Each test was made for 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000, 9.000 and 10.000 requests using Apache JMETER which is a load testing tool written in JAVA.

7.2.1 Load Testing

The following Figure 23 is the Error % for the POST request to do the login after the GET requests of the login.html so the browser can generate a CSRF token.

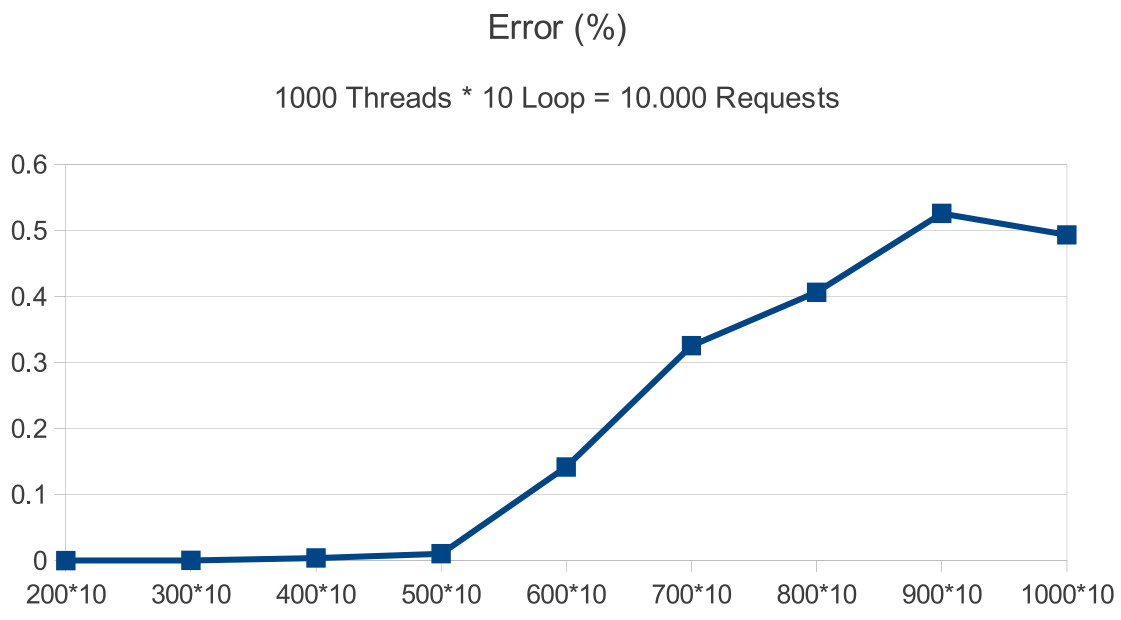


Figure 23: Testing Error % Post data Load Test

As we can see the error % increases when we increase the number of requests. This behavior is expected and also considered normal.

Still, the maximum error % for the maximum number of requests (10.000 requests) is relatively small (still less then 1%) which is acceptable.

The following Figure 24 is the throughput (requests per second) results for the POST request to do the login after the GET requests of the login.html so the browser can generate a CSRF token.

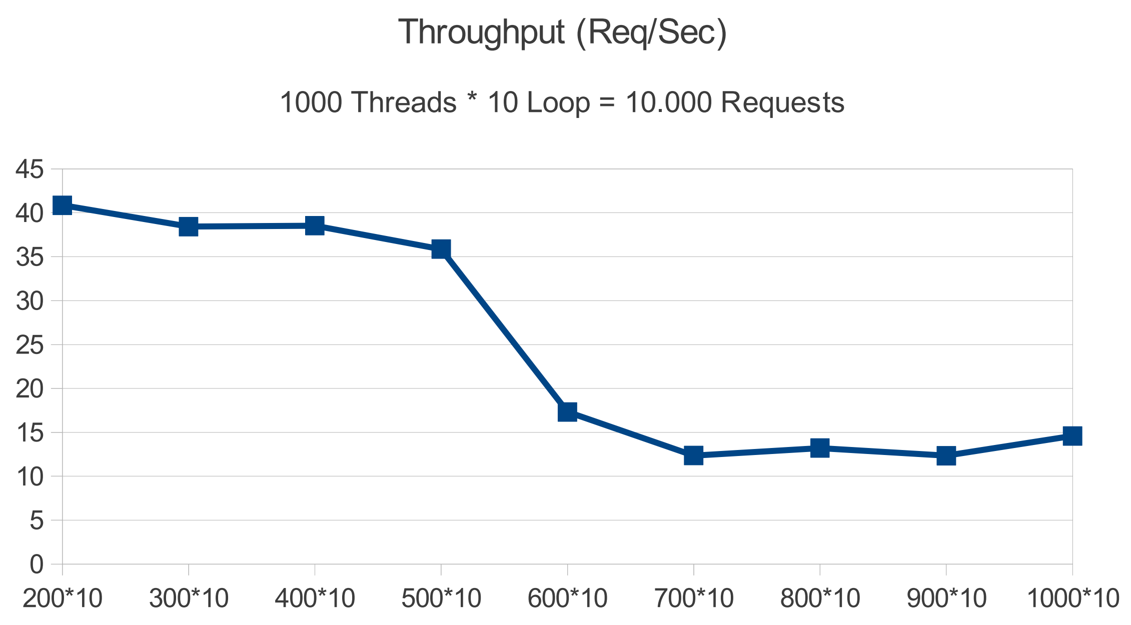


Figure 24: Testing Throughput of Post data Load Test

With the testing we were able to satisfy about 40 requests per second for 2.000 requests. This number tends to decrease as the number of requests increase. It does make sense, as this load test implies an interaction with the database and the more requests, the more resources need to be allocated, so the more requests we ask to the server, the longer it takes to process.

7.3 Usability Testing

For the purposes of evaluating the developed software by letting the users test, some usability tests were made.

All the information related to the Usability tests are in *“Annex E - Usability Tests”* for further consultation.

There were a total of 26 responses to the usability tests, that should take about 10 min total for each user. The usability test included the user registration/login, the creation of a SMARTURBIA APP, the editing of this APP, and the deletion of the APP. For each one of these tasks, the users had to rate their experience from 1 (very easy) to 5(very hard) as well as the estimated time he/she took to complete the given task.

In general, the user response was very positive. The users were able to execute all this tasks with minimum effort and rapidly.

8 Conclusions

In this internship we developed youbeQadmin, a management platform for youbeQ and Smarturbia.

Since time was an important factor there was a study in the state of the art to know what feasible alternatives were available, that could help speed up the development process in order to fulfill all or at the very least most of the software requirements while at the same time maintaining a well written, organized and scalable application.

So a decision had to be made regarding the use of a Web Framework, as opposed to developing everything without one. The advantages and disadvantages of Web Frameworks had to be taken into account when deciding this matter and if in fact a Web Framework was a proper solution for fast and reliable development, DJANGO (PYTHON) would be the companies chosen Web Framework.

Another decision that had to be made was the use of ORM when developing the application. Once again the advantages and disadvantages are a very good starting point to deciding whether to use this technology or just fall back to good old SQL.

After a thorough study of the state of the art, the decision to use Web Framework (DJANGO) was made, as well as using ORM for the prototyping phase (BETA release).

The project planing as well as the software requirements and architecture have proven to be extremely important allowing us to clearly see most of the stages to come and anticipate future problems avoiding schedule deviations.

The risk analysis has also proven to be of high importance as we can solve, or start solving problems that can delay the project right from the beginning.

All this software engineering tools are in fact reliable and time saving, as they help us build an healthy and strong base for our project and allow us to easily add improvements over the specifications in the future.

The testing phase was of equal importance. Automated Unit tests, performance tests and usability tests were made, although if there weren't so many complex requirements more automated unit tests could have been made.

Unfortunately, and specially for someone who is not very comfortable with automated testing, they can prove to be a little complex and time consuming at the beginning. In fact, before the internship I didn't really have much experience with automated testing.

While developing the manage platform, I also had to be in contact with other programming languages aside from PYTHON as well as open source technologies, in order to be able to integrate with the already existing and deployed applications.

Web development is certainly a System Information area of its own and I found myself confronted with many problems that I could not anticipate due to the lack of experience in the area. This was a task that I did overcome in time and that brought me very elucidative knowledge on the subjects.

8.1 Future Work

The work done for this project is not complete although the architecture has a solid base to support new requirements that may arise in the future.

There are some requirements that still need to be implemented, that are already specified but due to the lack of time and more prioritized requirements could not be completed.

There is also room for software optimizations. Some of them are already thought of since before even starting the development,when it was still in the specifying the requirements and architecture phase. The most important one is to speed up the database by converting the existing ORM solution to raw SQL. The database is usually the bottleneck of the DJANGO developed applications and the best optimization found until today was converting ORM to raw SQL (apart from SQL better written queries). The only problem for doing this right at the beginning is that it can take more time and it might make the development more complex.

There is in fact an article, from a MOZZILA FOUNDATION developer that was using DJANGO Web Framework to develop the MOZILLA VERBATIM website [12] and after developing it, he converted all queries to raw SQL and could speed up the database performance up to one thousand times faster [13].

This does in fact make us wonder why don't we just write raw SQL code, and the answer to this is that although optimization is always good, the time and resources that it takes might not compensate the amount of traffic that our web application has. This is the current case for the company, where the web traffic for some of its web applications do not match the investment.

References

1: Django Authors, Django Documentation, 2013
2: Brian D. Murphy, SLOCUM, JOSHUA, 1994, http://www.biographi.ca/en/bio/slocum_joshua_13E.html accessed June, 2013
3: Docforge, Web application framework, , http://docforge.com/wiki/Web_application_framework accessed June 2013
4: Ruben D'Oliveira, Pros And Cons Of Using Frameworks, 2011, http://www.1stwebdesigner.com/design/pros-cons-frameworks/ accessed June 2013
5: Robert Eckstein, Java SE Application Design With MVC, 2007
6: Jeffrey M. Barnes, Object-Relational Mapping as a PersistenceMechanism for Object-Oriented Applications, 2007
7: BHARGAV_VISANI, Waterfall Model,
8: Agile Alliance, Agile Development, , http://www.agilealliance.org/the-alliance/what-is-agile/ acessed June,2013
9: Celery Team, Celery: Distributed Task Queue, , http://celeryproject.org/accessed June, 2013
10: Brandon Konkle, Load Testing with JMeter, 2013, http://lincolnloop.com/blog/2011/sep/21/load-testing-jmeter-part-1-getting-started/ accessed June, 2013
11: Mike Kelly, Tips For Interpreting Results Jmeter, , http://searchsoftwarequality.techtarget.com/tip/Tips-for-interpreting-JMeter-results accessed June, 2013
12: Mozilla, Mozilla Verbatim, , https://localize.mozilla.org/ accesses June, 2013
13: Peter Bengtsson, An optimization story with Django – one thousand times faster!, , http://blog.mozilla.org/webdev/2011/12/15/django-optimization-story-thousand-times-faster/ accessed June, 2013

Annex A – State Of The Art

Annex B – Software Requirements

Annex C – Software Architecture

Annex D – Software Testing

Annex E – Usability Tests