Master In Informatics Engineering
Internship
Final Report

CitizenPal

Student:
Omar Asaad
ozaki@student.dei.uc.pt

Company:
Tangível
info@tangivel.com

Supervisor of DEI :
Filipe Araujo
filipius@uc.pt

Supervisor of Tangivel:
José Campos
jcampos@tangivel.com

September 1, 2016

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

Abstract

Currently, countries are competing to provide better services to citizens, by automating their services and making them available on the Internet. This makes citizens lives easier, by reducing the time and effort they spend interacting with the public administration. With the smartphone revolution, these services have become even more available and closer to the citizen. Unfortunately, electronic government (e-government) services are varied and different, because they belong to different institutions. Furthermore, each institution may use specific technology in those services.

In Portugal, a lot of e-government services have been already automated and are available on the Internet. Until now, however, there is no mobile application that allows citizens to interact with these services. This project studies the Portuguese e-government services and understands Portuguese government portals, in order to develop a mobile platform that manages the main obligations of citizens to the state and makes them work under a single application, using a coherent and simple to use interface.

Key Words: e-government, e-services, citizens, events, state services, public portals, mobile services

Contents

List of Figures

List of Tables

Definitions, Acronyms, and Abbreviations

RESTful	Representational State Transfer
API	Application Program Interface
DWP	Dynamic Web Project
JEE	Platform, Enterprise Edition (Java EE) is the standard in community-driven enterprise software.
User	The user is the actual utilizer of the application. In the mobile application, he is the citizen. In the back-office, he is the Admin.
TLS	Transport Layer Security
HTTPs	Hypertext Transfer Protocol (HTTP) within a connection encrypted by Transport Layer Security or its predecessor
AMA	In portuguese "Agência para a Modernização Administrativa", The Agency for Administrative Modernisation.
IDE	Integrated Development Environment.
API	Application Program Interface.
SDK	Software Development Kit
JRE	Java Runtime Environment
HTML	Hyper Text Markup Language
SPAs	Single Page Applications
UIs	User Interfaces.
JSON	JavaScript Object Notation.
RDP	Relational Database Provider
SQL	Structured Query Language
SSL	Secure Sockets Layer
MVC	Model View Controller
FCM	Firebase Clould Messaging
URL	Uniform Resource Identifier
CORS	Cross-Origin Resource Sharing
AMI	Amazon Machine Image
RDP	Remote Desktop Protocol
RDS	Relational Database Service
WAR	Web application ARchive
DFD	Data Flow Diagram
HDPI	High Dots Per Inch (Screen Resolution)

1 Introduction

The relation between the state and its citizens is of great importance. The state provides several services such as health services, citizens registrations, taxes and vehicles managements by its institutions. Recently, with the technological revolution, these services became automated (e-services). This improvement allows citizens to access most of its public services online. However, the uprising of smart phones with their great capability and connectivity, drives these services to be more accessible and easy to use. This project provides a prototype of a mobile app that makes these public e-services close and accessible to every citizen.

The document reflects the work that has been done for a CitizenPal proposal. CitizenPal proposal was introduced by a company called Tangivel. It is located in IPN (Institute Pedro Nunes) part of the effort to obtain a Master Degree in Informatics Engineering - Software Engineering field.

Following section gives a description of the purpose of the project. In addition, it determines the scope and the motivation of the project. Then, it specifies the methods and results obtained. Finally, it provides some definitions and abbreviations used in this documents.

1.1 Project's Goal

The main goal of the CitizenPal project is to develop a mobile application that demonstrates the ability of a citizen to get state-related events from several public e-services in one place using a smart phone device. It allows citizens to be notified and gives more details related to his events.

1.2 Scope and Motivation

In order to develop this application, this report identifies the most important tasks that citizens usually do, which are related to institutions that are owned by the state.

This report explores the relationships between the citizens in Portugal and the state. By exploring the main agencies and e-services in Portugal, the project raises many concerns related to the project.

To understand more about the e-services of the public portal, Tangivel had several meeting with the AMA[1] that responsible for these e-services. Then they have decided to build a prototype that reflects the idea of CitizenPal project. For more information, see project planning 10.8 in Appendixes.

E-services are very diverse, they cover most aspects of the citizens, companies and organizations. This project is focusing on the individual citizens tasks. Enterprises and companies tasks or related-relations with e-government are not included.
The project is targeting the citizens whom are familiar with the use of the current technology devices, such as smartphones, laptops, etc.

The motivation of the project is to develop a more efficient way to get the state e-services closer to the citizen. The main motivations can be summarized as follows:

1. Help citizen to remember more efficiently the state-related event and other important state obligations for example, renewing the Citizen Card, payment of taxes).

2. Give citizens more contextual information to their location, taxes situation and necessary documents are needed to perform these tasks. Which Assist citizens to do these tasks proactively.

3. Avoid passing through unpleasant experiences that create feelings of anger and frustration, and increase the citizen's lack of confidence in the state system.

[1]AGÊNCIA PARA A MODERNIZAÇÃO ADMINISTRATIVA

1.3 Method and Results

The e-government services in Portugal are very diverse. There are several institutions that already have their services online. We started by browsing all the e-government services and related institutions, to extract specific citizen-oriented tasks and events offered by these services. This study includes an overview of similar projects and platforms that have been made for the same purpose. Furthermore, Tangivel the company that is responsible for the project, contacted the Portuguese government agency that responsible of most of these services.

Based on this effort, and considering the goals of this project, we developed a demonstration of a mobile application for the citizens, and a number of accompanying server-side software, to support this mobile application. The result of the project is building the following three major applications:

- **Server app**: A Dynamic Web Java application with a RESTful API that receives requests from both the mobile-app and the back-office app.
- **Mobile app**: An Android Mobile application that citizens use to receive their events data and notifications.
- **Back-Office app**: A Web Client application that controls and manages the data of the application.

These three aforementioned applications works together to simulate the CitizenPal project.

1.4 Document Structure

This document represents the final thesis of CitizenPal project. The document is organized as 10 chapters.

Chapter 1 gives an **Introduction** to CitizenPal project, scope, motivation, methods Then Definitions and Abbreviations.

Chapter 2 explains **Project Background** which contains a description of project's perspective, users characteristics and general constrains that are related to the project. In addition, it specifies the techniques and technologies that have been used in the project.

Chapter 3 specifies the **Requirements** of the project. It aims to give more understanding of the project context by introducing users stories and scenarios. Then enlisting the External Interface Requirement such as user, hardware, software and communication interfaces. Furthermore, it specifies the functional and non-functional requirements. In the end, it points out some important notes as prioritization of some previous requirements requirements.

Chapter 4 illustrates the **Implementation** of the CitizenPal System. It depicts the Architecture of whole system. This section describes the three main architectural components of server app, mobile app and back office app. It explains what have been done to make them work together. Finally, it points out some assumptions and dependencies.

Chapter 5 decorate the project **Deployment** and describe the deployment's specifications and devices.

Chapter 6 shows **Testing** process and test sets for the system. In this document, two types of testing has been done. A load testing and user acceptance input testing.

Chapter 7 have a **Discussion and Future Work** about the project in general. Discussion points out some important decisions during the life time of the project. Finally, it shows some important issues that can be done as . These Future works can improve and enhance the project.

Chapter 8 draws **Conclusion** of the main issues in the project and presents the results that have been obtained.

Chapter 9 enlists all **References and Bibliographies** that have been used in this documents.

Chapter 10 has all the extra **Appendices** documents and resources that include in the project.

2 Project Background

2.1 Overview Model

The impact of the technology and information access is rising in government services. In fact, the interactive nature of the Internet technology has a great potential to make government function better that it currently does. Therefore, most of modern governments tend to improve their services for citizens. Consequently, rising the projects that had been made in this area, most of them related to e-government and e-services approach.
E-services refers to the delivery of information and services online through the Internet or other digital means. In Portugal, citizens are using the Internet services very frequent. In fact, the e-government indicators refer that the individuals that are using Internet to interact with public authorities is also increasing as figure 1 shows.

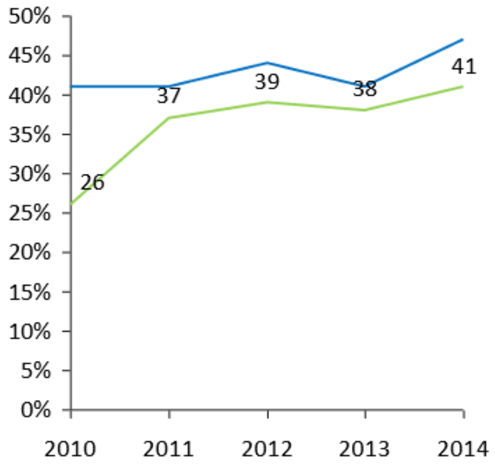


Figure 1: Percentage of citizens using public e-services in Portugal by year

The figure 1 compares the statistic indicator for the European Union average (blue line) and Portugal state indicator(green line)[1]. This increasing indicates that citizens are aware of these services. This gives CitizenPal app more chance to be used.

As a part of EU, the state of Portugal has great concerns to develop and improve their online services for the citizens. For that reason in 2007, the Agency of Administrative Modernization or (AMA) has bean created. AMA is the public institution that fulfills the duties of the Ministers Council Presidency, in the areas of modernization and administrative simplification and electronic administration. It works under the authority of the Secretary of State for Administrative Modernization.

To reduce bureaucracy and improve the usability of public services, the “Citizen Spot” or “Loja do cidadão” has been created. The citizen spot was the result of the cooperation between the central government and the local government. It should be opened and ready to collaborate with other entities [4]. This will allow all these entities that also provide public services, to be gathered in one place with several political and economical benefits.

In addition to “Citizen Spot”, AMA has developed several electronic web portals, to be served under the e-government infrastructure such as “Portal do Cidadão” or “Citizen Portal”, “Portal da Empresa” or “Enterprise Portal”, “Portal das Finanças” or “Taxes Portal”, and Open Data Portal, among others. A web portal is a web site or service that offers a broad array of resources and services.
The following points are examining these portals in more details so as to have better understanding for their e-services:

- The “Citizen Portal” [2] is the central channel for electronic access to public services. It currently offers many citizen-oriented services. These services are provided by several bodies and public entities. Citizen can be able to consult different kinds of information, regarding daily life events such as birth, death, employments, health , justice and tax systems.
  The keys that citizen should provide to access these portal usually the citizen national number and a password.

- The “Enterprise Portal” [3] provides electronic access to public services supplied to businesses and entrepreneurs. It serves as portal for enterprises and companies, and it brings together information of interest for business activities related to the management, expansion and closure of enterprises.
  In order to access these services that citizen should provides business’s fiscal number “NIF” and password.

- The “Taxes Portal” [4] provides citizens several services, which allows for example tax submission and fiscal situation consultation, among a wide range of services.
  Taxes Portal needs fisical citizen number “NIF” and password to get access these services.

- “Health Portal” or “Portal do Utente” [5] The Citizen Health Portal aims to be a bridge between the citizen and the National Health Services. It offers health related information and services, as booking a doctor appointment or renewing regular medication.
  Citizen needs to provide his Social Security number or “Numero de Segurança Social” SNS and a password to access health services.

The interaction between these portals and the citizens happens when a citizen visits the portals to consult some information or to submit some tax forms.
A citizen can register then log in on most important portals using his service numbers and password. Every portal should have different credentials. This way of interaction is very productive and it reduces the bureaucracy, but in some cases the State has to notify the citizens for specific events and deadlines. Citizens sometimes do not check the portals, or they forgot the dates, or do not read the emails from the portals. This notification is crucial because usually the payments or the submission of an information has to be in specific period of month or year, if not, the tax system imposes an extra fine on delay.

This project is a prototype that is allowing citizens to place all the needed credentials in one place which is the citizenPal app. Then the app can retrieve and deliver these important events right to his mobile device.

[2]Citizen Portal:`https://www.portaldocidadao.pt/`
[3]Enterprise Portal:`https://bde.portaldocidadao.pt`
[4]Taxes Portal:`http://www.portaldasfinancas.gov.pt`
[5]Health Portal: `https://servicos.min-saude.pt/utente/`

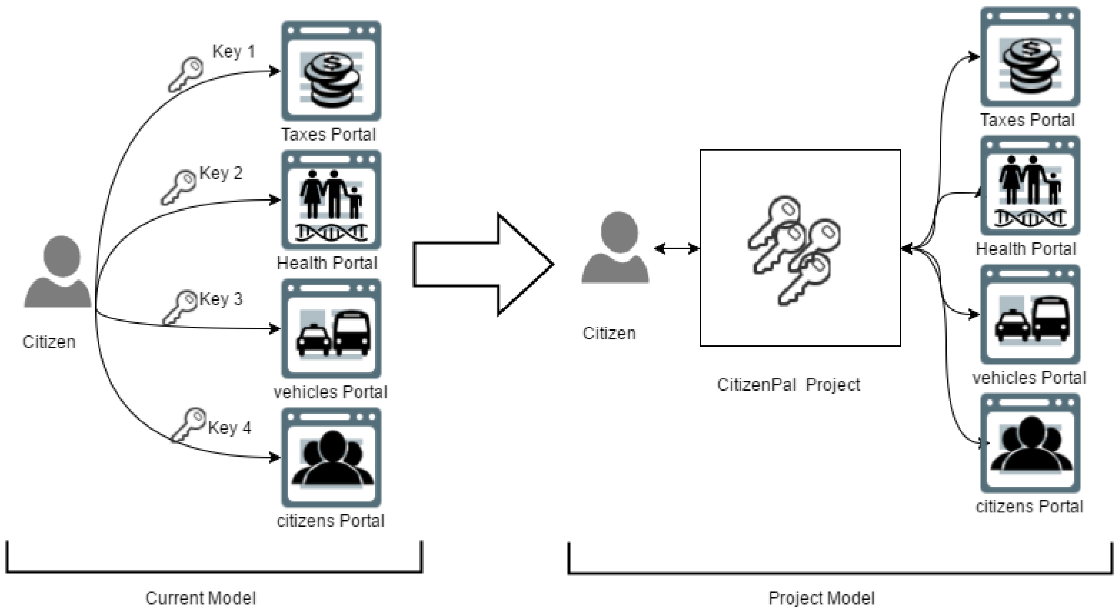


Figure 2: CitizenPal product model

As it can be seen in figure 2, CitizenPal changes the current interaction model with public services. It allows citizen to save and place all credentials inside the app. And the project access the portals and bring the most important events and tasks to the citizen.

2.2 Product Perspectives

In order to make the model of the CitizenPal application more meaningful, specific aspects should be pointed out. This section defines several points that would clarify and refine the project.

Portals have several tasks and data to retrieve. Therefore, CitizenPal should define a generic event type, which refers to a task that could take place. Each event belongs to specific category. for example "Model 3 submission" is an event belong to Taxes Category. This prototype covers following category :

- Taxes
- Health
- Vehicles
- Citizens

All the previous portals have been controlled by the government. The government is more conscious to give access to these portals. For that reason, and to proceed in the project. It is important to have a kind or "proxy" (middle tier) that can be completed in the future as they gives access to these portals.

These specific events data should be retrieved and stored in somewhere. Then they will be delivered to the citizen's smart phone. As a result, CitizenPal should have some kind of interfaces that retrieve these information and store then deliver them to the citizen.

Events data have to be stored, this require a management application or a back-office to control these stored data on the application. This will allows an manager to control the citizen events data. This prototype should have this kind of mechanism although it rise several considerations related to privacy.

Since the citizen will use and store his keys of these portals. The project has to have a secure connection and secure database to store these credentials. Security is important aspect for the project.

The citizen's event, usually, can be valid for specific period of time. It has starting and ending date. Nevertheless, it could more flexible and effective to allow citizen to organize these events with his own calendar. So citizen can attach this Event at specific date in that period, depending on his schedule.

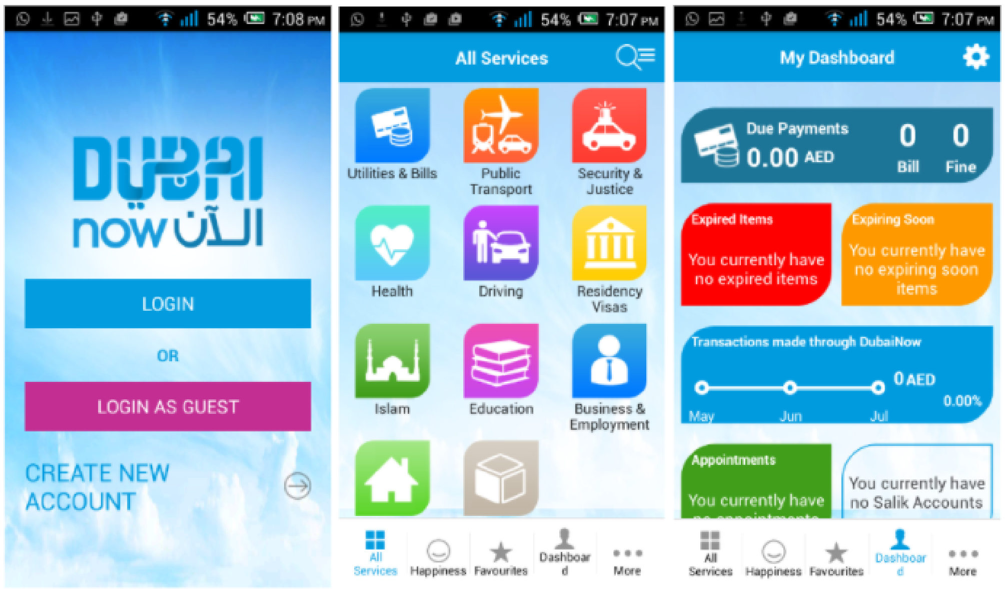
Usually, the events are stored in app storage even if these event has been done by the citizen. Therefore, it is more user friendly to provide the citizen a way to refer to an event as done or finished.

To put the CitizenPal product in perspective, lets examine the related current products that already exist. Most of the current apps that are already made in the store is targeting specific services such as:

- **E-services app:** Smart-phone apps that provide e-government services in Portugal are not exist yet for citizens. As it has been seeing previously, there is that "Citizen's Place" which is a new assisted service model, closer to the populations, which brings together services available on the Internet from various public bodies and companies in a single physical access point, which acts as an extension of the Citizen Shops.

  Internationally, There are some apps that provide e-services from several institutions on one application such as DubaiNow app which provides the citizens in Dubai with a lot of public services from several governments entities such as bills and payments, health care , driving and vehicles, housing, residency visas, security and justice, education and more.

  Figure 3 shows several screenshots of DubaiNow app.



You currently have no expired items

You currently have no expiring soon items

You currently have no appointments

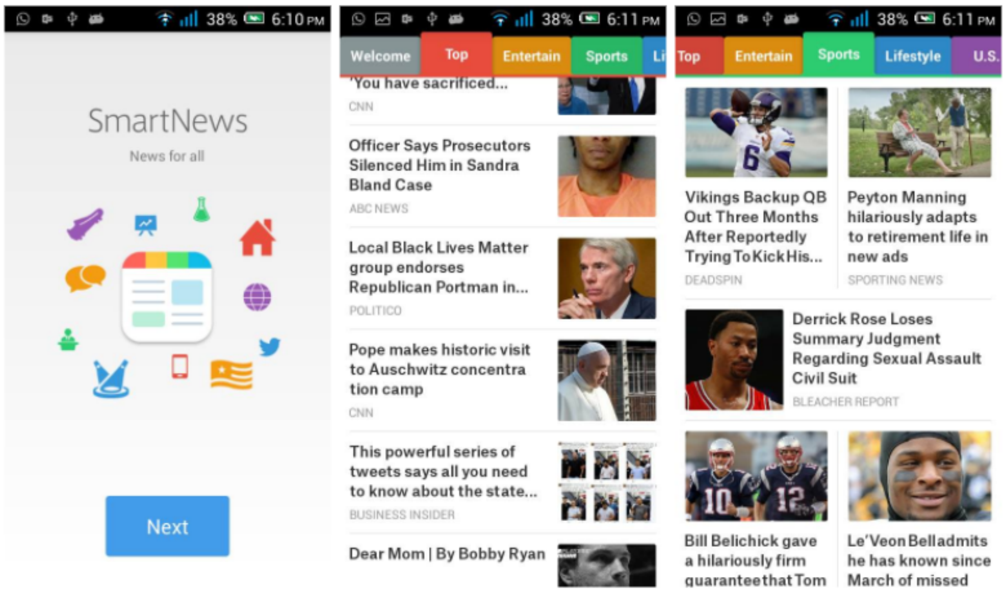
You currently have no Salik Accounts

Figure 3: DubaiNow is a unified public services application.

- **Collective News apps** : That brings news from different portals such and news agencies. It allows the User to obtains news services from several portals. For example, SmartNews app which analyzes the articles of the day to deliver the top trending stories from the trustful sources such as NBC News, TechCrunch,The Huffington Post and more.

  This app might have similar structure since it retrieves the information from several resources then store them in one place. Even though, this application does not require any

verification or storing keys in the application.
Farther more, this app uses user's current location to provide accurate weather forecasting, articles specific to your area, and enhance your social media options. It notify the user as well for the new interested articles. Figure 4 shows several screenshots of smartNews app.


SmartNews
News for all
Next

Welcome Top Entertain Sports Li

'You have sacrificed...
CNN

Officer Says Prosecutors Silenced Him in Sandra Bland Case
ABC NEWS

Local Black Lives Matter group endorses Republican Portman in...
POLITICO

Pope makes historic visit to Auschwitz concentra tion camp
CNN

This powerful series of tweets says all you need to know about the state...
BUSINESS INSIDER

Dear Mom | By Bobby Ryan

Top Entertain Sports Lifestyle U.S.

Vikings Backup QB Out Three Months After Reportedly Trying To Kick His...
DEADSPIN

Peyton Manning hilariously adapts to retirement life in new ads
SPORTING NEWS

Derrick Rose Loses Summary Judgment Regarding Sexual Assault Civil Suit
BLEACHER REPORT

Bill Belichick gave a hilariously firm guarantee that Tom

Le'Veon Bell admits he has known since March of missed

Figure 4: SmartNews is a collective news application.

CitizenPal Application might simulates and provides this ability. It focuses on retrieving collective and important information for user events.

2.3 Technologies and Techniques

In this section contains all Technologies that has been used to planning, developing and deploying the project. Including programming libraries, data forms, IDEs and APIs.

2.3.1 Server App Technologies

Server application has been developed on Eclipse MARS Java EE IDE for web Developers. It is a Java Dynamic Web Application by using Enterprise Edition (Java EE). Server app works as RESTful API. RESTful is a great software architectural style that allows provide client-Sever application with several benefits such as simplicity , performance, lightweight and scalability.
The libraries that has been used in the development are:

- **Jersey API:** Is a framework that works as implementation of JAX-RS in Java programming language. It supports several features that allows building Web services and their clients. Is support several media type and data formats such as xml and JSON.In addition it has a authentication structure and client API to be used in the client.

  Jersey 2.2.7 has been used in the development link to library reference [4].

- **Jackson API:** Jackson is a very popular and efficient Java-based library to serialize or map Java objects to JSON and vice versa. Also it is the ability to be integrated with Jersey API.

  Jackson v1.9.0 is used in the current development.

The RESTful Web services is deployed on Appache Tomcat. Appache Tomocat is an open source software implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies.[5]

Appache Tomcat 7.0 is used in the current development.

2.3.2 Smart-Phone App Technologies

The smart-phone App is and Android application that build Using Android Studio IDE. Android Studio is the official IDE for Android development which supported by Google. [6]

Android Studio 2.1 is used in the current development. The libraries that used in the development of the Android app are :

- **Jersey Client API:** a client-side of Jersey API. 2.3.1
- **Jackson** : for the data format as well.That suitable with android development.

Android App has several Specifications that determine the main properties of the android application deployment. They can be found in app.gradle in the main android app path. Some of important properties are :

- **compileSdkVersion** : it specifies the current SDK level that used to compile the app to particular version of Android. The SDK level that has been used in CitizenPal development is SDK 23.0 which use to compile app for Android 6.0 (Android Marshmallow)
- **minSdkVersion**: it is the minimum SDK level version that the app can be working on. In CitizenPal, it is SDK 17.0 which runs over android 4.2 (Jelly Beans)

That means that the CitizenPal will not be able to install or work on the Android systems that are before Android 4.2 (Jelly Beans). The app can is capable to work on Android systems that are after Android 6.0 (Marshmallow), But the new features of the new Android system might not work properly.

2.3.3 Back-Office Technologies

The back-office is individual web-client application that is working on different machine than than the sever. It build using JavaScript and HTML. The app developed using WebStorm IDE. The APIs and libraries that has been used In the development are :

- **AngularJS:** A JavaScript API used to develop a responsive web-client application. AngularJS extends HTML with new attributes. It allows us to build a Modern web application which called SPAs (Single Page Application) that has several advantages. It used MVC software pattern with give the application the ability to developed, managed , developed more easily. [7]
  This API Uses JQuery to be able to works.

  CitizenPal project uses Angular 1.3.20 used in the in the development of the back-Office.
- **Bootstrap:** Used for the UI of the application. A sleek, intuitive, and most popular HTML, CSS, and JS framework for developing responsive projects on the web. [8]

  CitizenPal project uses Bootstrap 3.3.6 for for the UIs of the Back-Office.

2.3.4 Database Technologies:

CitizenPal project uses Relational Database to store and retrieve the citizen events. Mysql 5.7 used as database provider. To manage the database tables, CitizenPal project uses MYSQL WorkBench 6.3 as tool.

Figure 5: Technologies APIs and IDEs brands

2.4 User Characteristics

CitizenPal app is targeting users who are in a level of education that can use smartphone features and can be aware with network services. Basically, CitizenPal app is targeting most of citizens that are in age of 18 to 40 years old. The users should have an obligations that related to governments services such as taxes, renew card, car registration.
On other hand, since the app should have a back office window to control and monitor some information in it. It has to have a system manager or administrator. The administrator should be familiar in using the Internet and should receive a training of using this the administration panel and he should be aware of the regulations terms. In general, there are two types of users shall use this app:

- **Citizens:** They are the smart-phone users with background of using the Internet and have already obligations related to public services.
- **Admin:** The administrator of the app, should responsible of controlling the data, familiar with using Internet. and awareness of privacy terms and regulation. In addition, he should know the back-office tasks and functions.

2.5 General Constraints

Since the app is being developed to provide the citizens with their events, It should have high status in terms of privacy and regulation.

2.5.1 Regulation and Privacy

CitizenPal application intend to collect some sensitive information from the citizens. In order achieves this, the application should respect the regulations. Before availing the project to public, it should consult a lawyer that have expertise in government sensitive data. This should be done by collaborating with the related agencies of the government. In this context, we are proceeding in developing the project functionality to present the idea assuming that this issue can be resolved later.

2.5.2 Security and Encryption

Cititzens should register an account in application, they should agree the terms of use and conditions stating that none of the user's information will be disclosed to entities outside of the company.

In CitizenPal database, there has to be a means to preserve the user's privacy, in the event of a security breach. So, the most sensitive information, like the user's password or locations must be encrypted, in an attempt to minimize the damages.

All communications are going to carried out by HTTPs protocol by using SSL encryption with signed certificate that allows to verifies authentications between the Server app from one side, and the smart-phone app and back-office app from other side. Details will be explained in the implementation section 4.

Finally, In order to provide the ability of managing the user data. Server app should distinguish between the citizen connection and the admin connection. This will create kind of authorization policy for the incoming communication. The document will go through it with more details in the chapter 4.

3 Requirements

This section introduces mainly the requirements of the project. It examine the functional and non-functional requirement. It presents a user stories and External Interfaces.

3.1 Users Scenarios and Stories

In order to understand the app users, It is an important, to be aware with the target user. This section will presents two Users stories and scenarios to help us to understand the project.

3.1.1 User Story 1.Inês Santos : University Student

Personal personal info:

- **Age:** 23 years old.
- **Home Address:** Coimbra.
- **Civil status:** Single.
- **Profession:** Master student in pharmacy.

Experiences:

- **Internet:** Regular using.
- **Smart-phone device:** Samsung Galaxy Mini.Android KitKat
- **Apps:** She uses a little, specially the social application.
- **Computer:** She uses it a little for studying purposes.

Inês's Story

Ines is the younger sister among three children. She is still living with her parents. She has to study during week days. In addition, her parents are old and they can not use the Internet. She is helping them for taxes payments. She will appreciate if there is a way to know when the event of payments of renewing some public documents happen without going to checkout that for her parents.
This app will provide her with a perfect way to keep track of her events or her parents events without making and big effort.

3.1.2 User Story 2.João Veríssimo : Employee in Decathlon

Personal personal info:

- **Age:** 26 years old.
- **Home Address:** Lisbon.
- **Civil status:** married.
- **Profession:** Employee at Decathlon company.

Experiences:

- **Internet:** Regular using.
- **Smart-phone device:** Alcatel pop 7, Jelly Bean.
- **Apps using:** A little. social used.
- **Computer's Using:** he uses it as a part of his job.

João's Story

João is a hard worker employed at Decathlon in Lisbon. He goes to the work place by his car. His full time job gives him a limited time to do his other obligations such as car's taxes and Modelo 3 submission of his taxes. He would like to know the schedule of that payments because he prefers to manage his taxes payments. So he can handle his time out of his job.

3.2 External Interface Requirements

3.2.1 User Interfaces

User interface (UI) has been designed to both end-user applications, smart phone and back office application. Server application does not have an user interface.

Smart-Phone User Interface

It is important to scratch the interface before start developing which will give us a overview of the interface and functions of the project. Next figures contains the prototype user interfaces that have been planned.

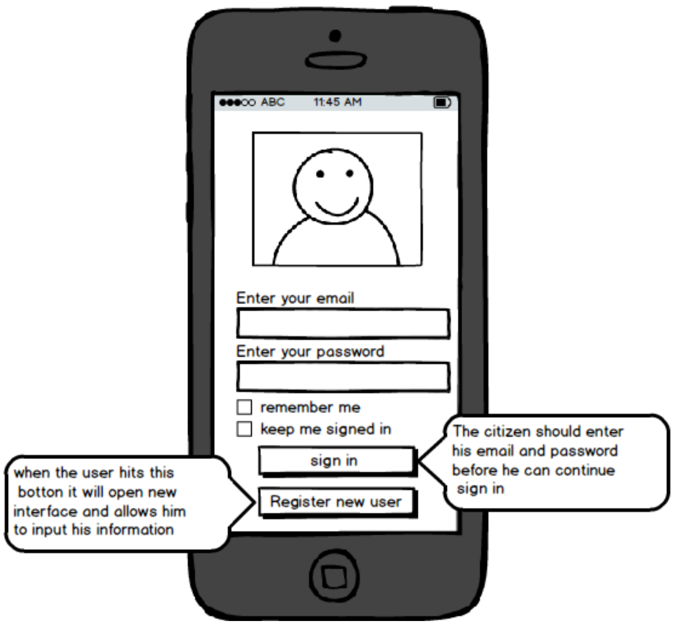

●●●○○ ABC 11:45 AM
Enter your email
Enter your password
remember me
keep me signed in
sign in
Register new user
The citizen should enter his email and password before he can continue sign in
when the user hits this botton it will open new interface and allows him to input his information

Figure 6: CitizenPal smart-phone app, login page

The rest of the user interfaces for the smart-phone application can be seen in appendixes 10.1

Back-Office User Interface

Since the back-office app is and web client application. The interface should be designed to be responsive and easy to use. In early planning, The scratch design was the following.

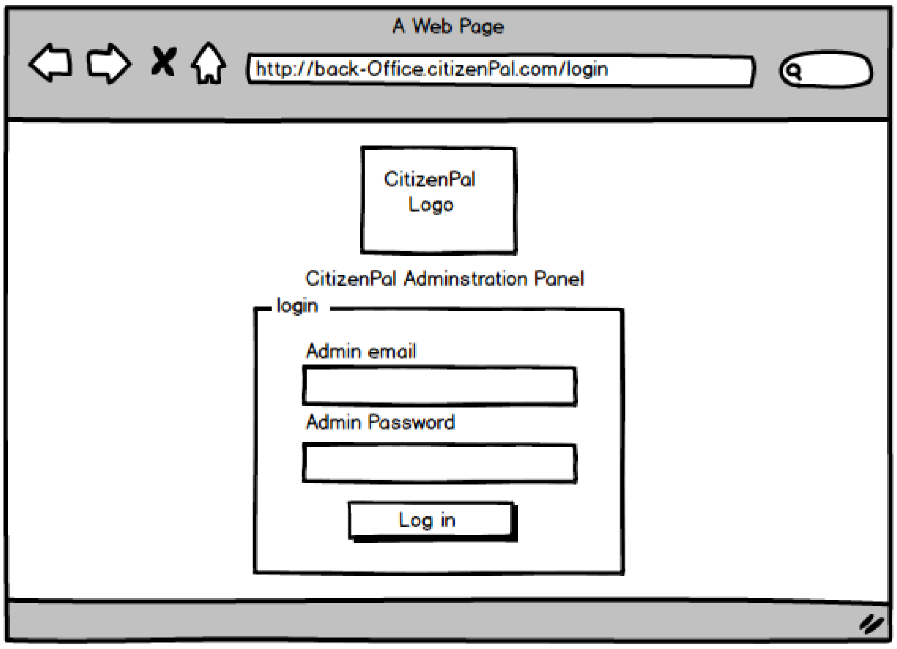

A Web Page
http://back-Office.citizenPal.com/login
CitizenPal Logo
CitizenPal Adminstration Panel
login
Admin email
Admin Password
Log in

Figure 7: UI of CitizenPal back-office app, login page

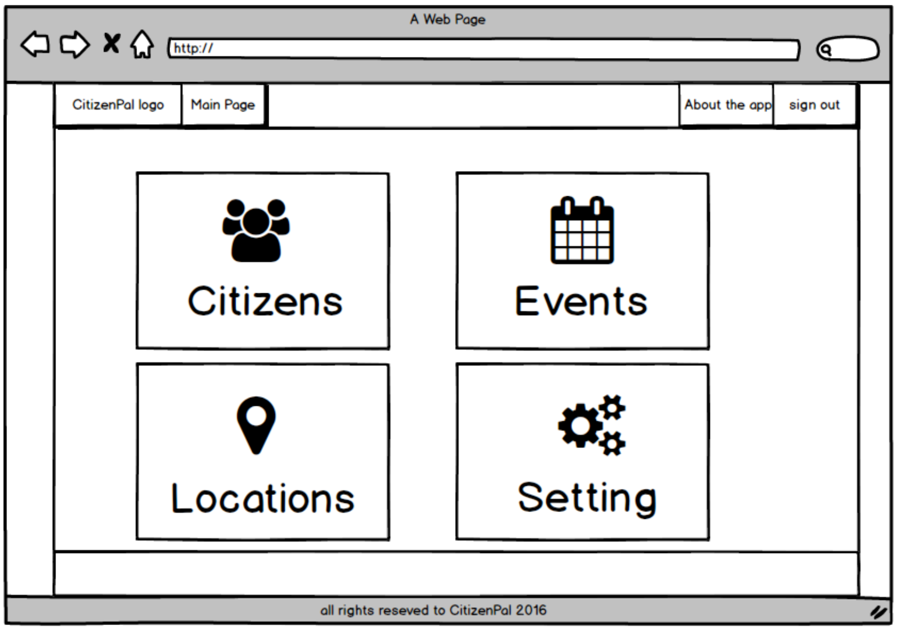


Figure 8: CitizenPal back-office app, main page

The full user interfaces scratches can be seen in the Appendixes section 10.2.

3.2.2 Hardware Interfaces

For the project to function, it has to have several hardware interfaces. CitizenPal requires smart phone with Android system installed and functioning. CitizenPal smart phone app will be tested on Android device with 480*800 HDPI with 4.0 inch screen. It might have some differences with the layout on smart-phones with other resolution display. The following figure 10 shows hardware interfaces for CitizenPal system.

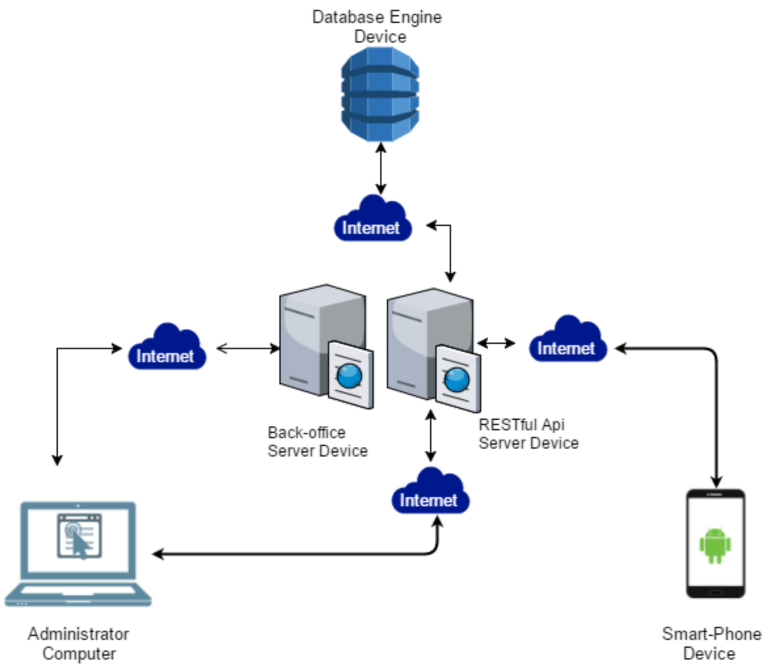


Figure 9: Hardware interfaces of CitizenPal project

In addition, there is a need for a two machines that can deploy the server app and another one for deploying the back-office app and it has to be always connected to Internet.

In order for the back-office to function, Administrator's computer device is required. This device works as client eligible to connect to use the Internet.

Furthermore, The data should be stored in a machine that hosts Relational Database Provider (RDP) server. As the previous devices. It should be connected to Internet.

3.2.3 Software Interfaces

In order for the citizen to use the application, he should have an smart phone with Android OS installed on it. Android OS is occupying the majority of the market share, with around 82.8% [9] . This will give more potentials for the app to be installed on more devices. CitizenPal app can work on Android OS Jelly Beans or 4.2 and above. The previous Android platforms should not have that much effects because the latest statistics from Google showed that the percentage for smart-phones that are still using Android OS before than Jelly Beans version is around 3.7% from all devices that use Android OS.[10]

For the CitizenPal server application, the machine that host the server app should have a Apache server installed and running. The server application will be deployed on the Apache tomcat server as WAR file. The Apache server should be running and ready to receive a requests from both smart-phone app and the back-office app. More details in the Deployment chapter 5.

Since the back-office is an web-client application, it needs a browser to run the app, and server to host the app as well. The application has been developed and hosted on Apache tomcat 7 and tested Google Chrome Version 51.0.x. But it works on other browsers such as Mozilla Firefox or Internet Explorer. It might need to deactivate the security options with the back-office because the application should accept the sef-signed certificate.

3.2.4 Communications Interfaces

Most the functionalities in the project needs and Internet connection. CitizenPal smart phone app should be able to request specific resources from the server. The Back-office as well will have to send requests to the server too. The server needs to connect to database and the portals interfaces to receive store and receives data. All requests that carried out will be a stateless requests over the HTTP protocol. That means, it will not be any information stored about the client in the request session. Every request should prove itself by carrying user's credential before it can allowed to access the resource is showing. Farther more, using HTTPs added more security layer to connection between the client and the server. figure 10 shows the stateless requests and communications interfaces.

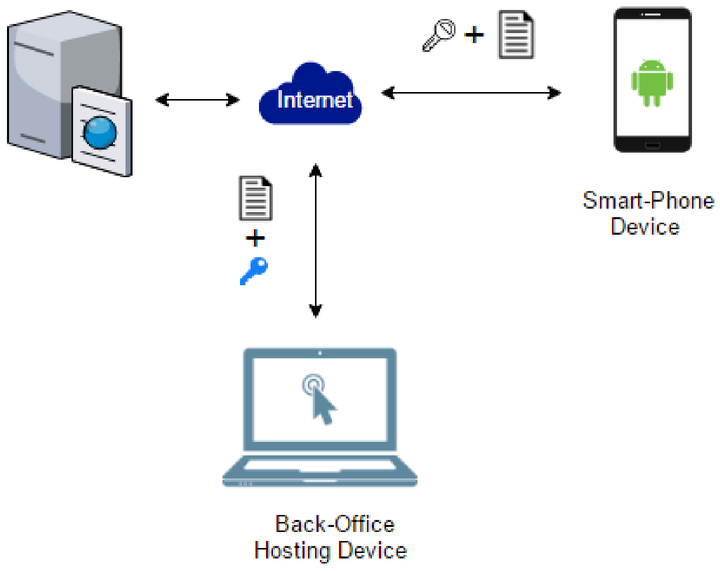


Figure 10: Communications interfaces of CitizenPal project

3.3 Functional Requirements

This section describes the features and functional tasks of the project. These features and tasks has been described as use cases, functional requirement and data flow diagram.

3.3.1 Use Cases

Use cases aimed to help to describe the system for external point view. The Use Cases of the project is described based on these properties.

Use Case ID	A unique numeric identifier for the Use Case
Use Case Name	A short name for the Use Case using an active verb phrase
Use Case Description	Briefly description of the use case and some previous conditions
Use Case Actor	Actor who's goal is being satisfied by this use case and has the primary interest in the outcome of this use case. In CitizenPal, There are three main Actors: • **Citizen**: an actor that plays the major role in smart-phone application. • **Admin**: an actor that plays the major role in back-office web application. • **Client**: an actor that plays the major role in the server server application. It could be a the client from smart-phone , or from the back-office.
Priority Level	The use case level specifies the priority of the use case. There are three levels for the use case priority : • **Essential**: Use case is not acceptable unless these requirements are satisfied. • **Conditional**: would enhance the product, but the product is not unacceptable if absent. • **Optional**: Use case that may or may not be worthwhile.

Table 1: Use Cases Properties

The following is the list of the use cases of the CitizenPal project. The details of each of them can be found in the Appendixes in section 10.3 . The use cases has been categorized based on the application type.

Smart-Phone Use Case

Usecase Id	Name	Periority
SP-UC-01	Register User	Essential
SP-UC-02	Log in user	Essential
SP-UC-03	Show User events	Essential
SP-UC-04	Filter Events	Essential
SP-UC-05	Show Events Details	Essential
SP-UC-06	Schedule Event	Essential
SP-UC-07	Mark Event	Essential
SP-UC-08	Save Email And password	Essential
SP-UC-09	Keep me logged in	Essential
SP-UC-10	Notify for new event	Essential

Table 2: Smart-phone Use Cases

Use Cases diagram

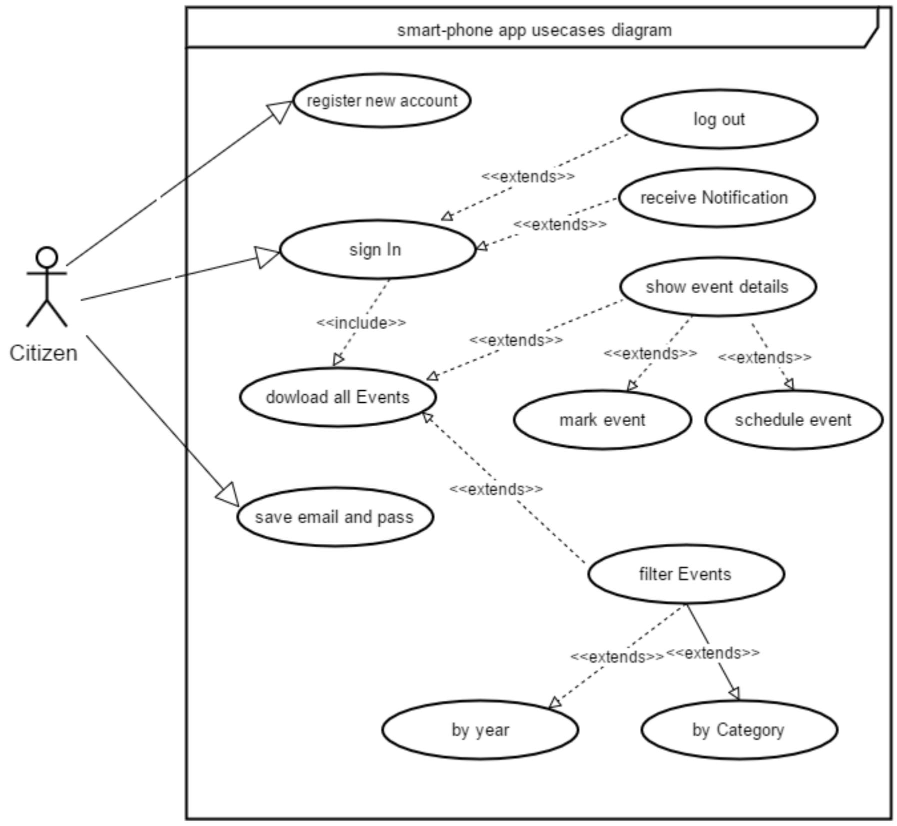


Figure 11: Use cases diagram of the smart-phone app.

Back-Office Use Case

Usecase Id	Name	Priority level
BO-UC-01	Log in Admin	Essential
BO-UC-02	Show all citizens info	Essential
BO-UC-03	Add New Citizen	Essential
BO-UC-04	Edit citizen Information	Essential
BO-UC-05	Delete citizen	Essential
BO-UC-06	Show Citizen Event	Essential
BO-UC-07	detach citizen Event	Essential
BO-UC-08	Attach Event	Essential
BO-UC-09	Filter Event Record	Essential
BO-UC-10	Filter Citizen Record	Essential
BO-UC-11	Show Events information	Essential
BO-UC-12	Edit Events Information	Essential
BO-UC-13	Add New Event	Essential
BO-UC-14	Delete Event Record	Essential
BO-UC-15	Browse Event Users	Essential
BO-UC-16	Deliver Event	Essential
BO-UC-17	Deliver User Event	Essential
BO-UC-18	Show Locations Info	Essential
BO-UC-19	Edit Location Title	Essential
BO-UC-20	Add Location Title	Essential
BO-UC-21	Delete Location Information	Essential
BO-UC-22	Portal Setting	Essential
BO-UC-23	Add Portal links info	Recommended
BO-UC-24	Admin should keep logged in during the session	Essential

Table 3: Back-Office Use Cases

back-office use cases diagrams

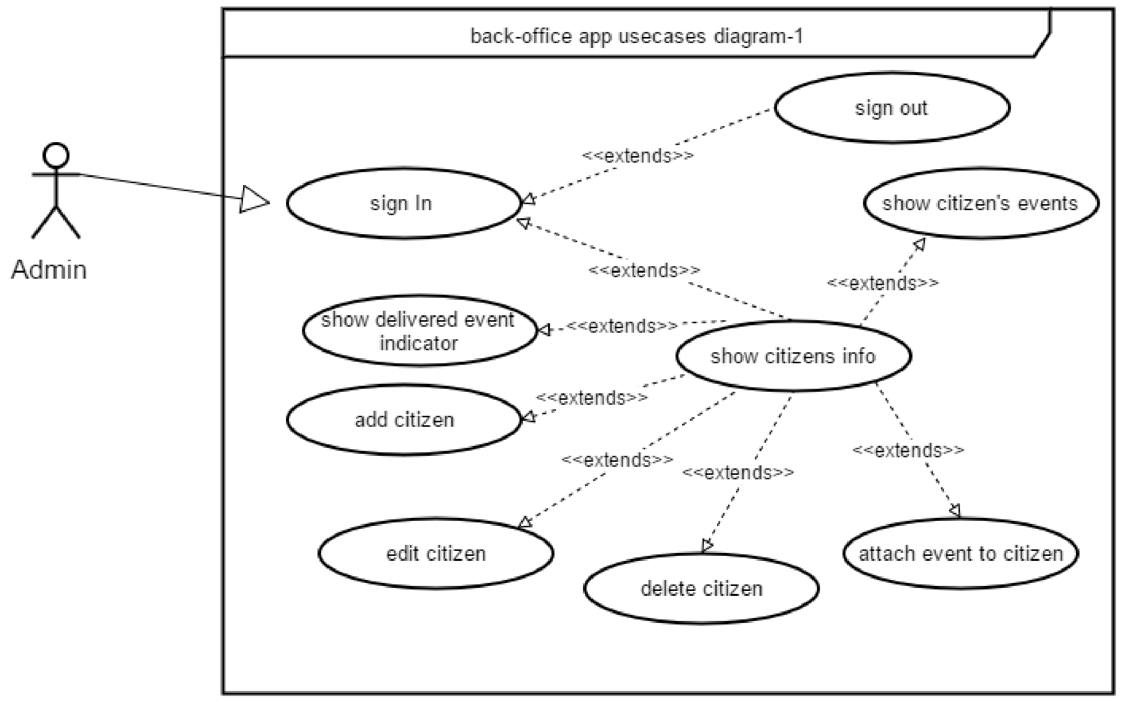


Figure 12: Use Cases diagram of back-office, first diagram.

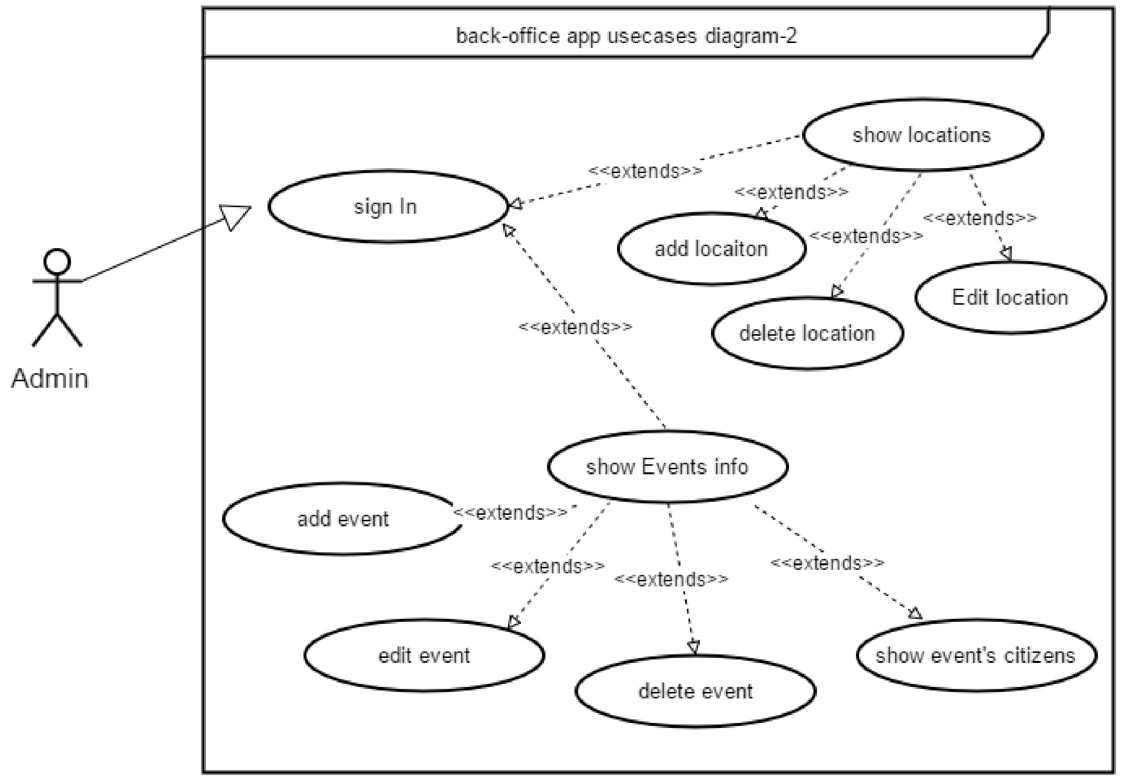


Figure 13: Use Cases diagram of back-office, second diagram.

Server application use cases

The server receives a request from a client (smart-phone app or back-office app). The server has the following use cases:

Usecase Id	Name	Priority
SA-UC-01	Register new citizen	Essential
SA-UC-02	Add new citizen	Conditional
SA-UC-03	Log in admin	Essential
SA-UC-04	Delete citizen account	Conditional
SA-UC-05	Edit citizen	Essential
SA-UC-06	Edit citizen's flag	Essential
SA-UC-07	Retrieve citizen info	Essential
SA-UC-08	Retrieve citizen events	Essential
SA-UC-09	Add event to Citizen	Essential
SA-UC-10	Detach an event from citizen	Essential
SA-UC-11	Retrieve all locations info	Essential
SA-UC-12	Add new location	Optional
SA-UC-13	Edit location info	Optional
SA-UC-14	Retrieve all events info	Essential
SA-UC-15	Edit event info	Essential
SA-UC-16	Retrieve citizens of specific event	Essential
SA-UC-17	Authenticate Admin	Essential
SA-UC-18	Authenticate citizen	Essential
SA-UC-19	Add new event record.	Essential
SA-UC-20	log requests to file.	Optional

Table 4: The server Use Cases

Server app use case diagram

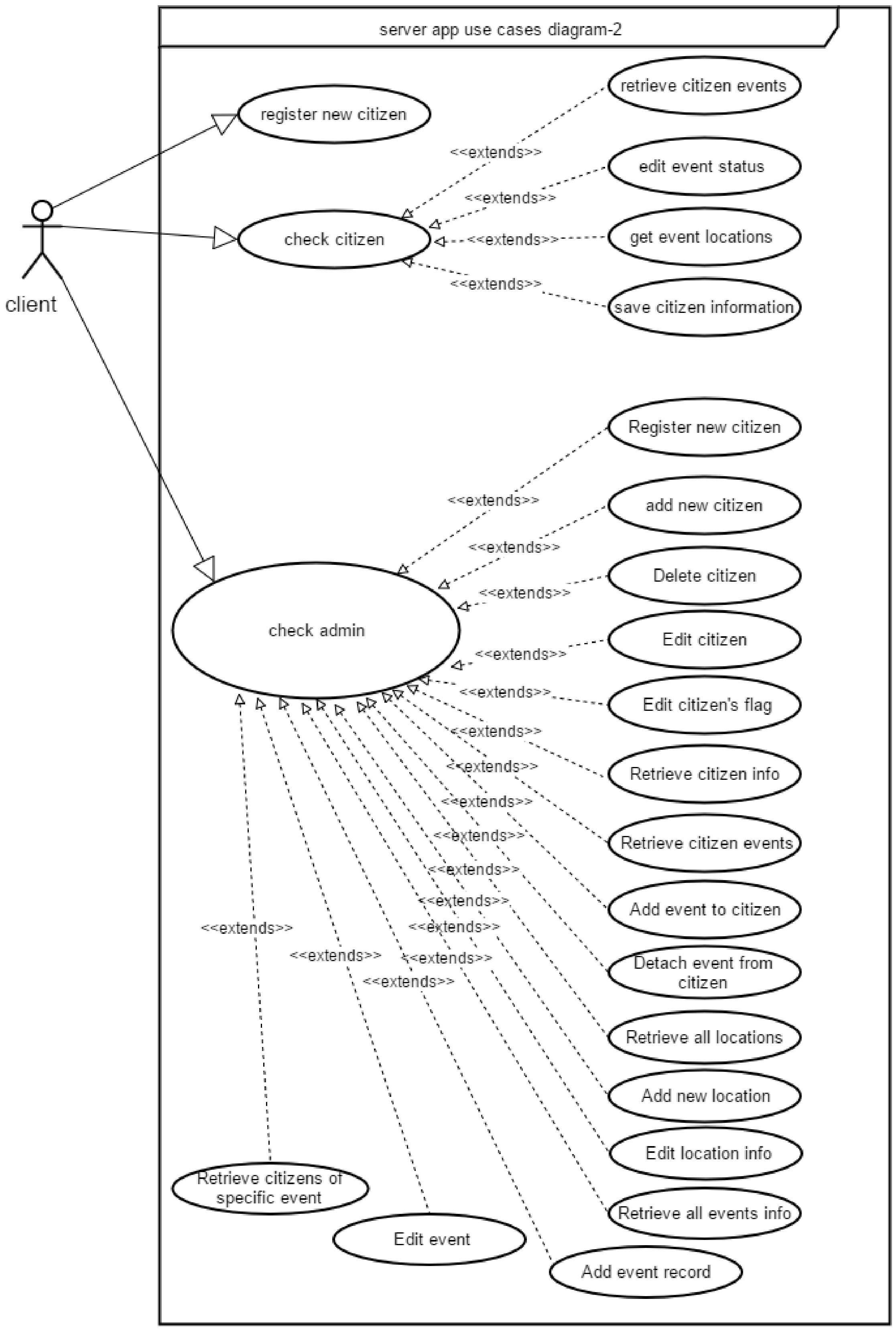


Figure 14: Use cases diagram of Server app.

These use cases are explained in details in Appendixes 10.3.2

3.3.2 Data Flow Diagrams

This diagram provides a visual explanation of the information flow within the CitizenPal system. It is showing the external entities that are involved in the system. Initially, The DFD context diagram is the general gives the general view of data flow in the system.

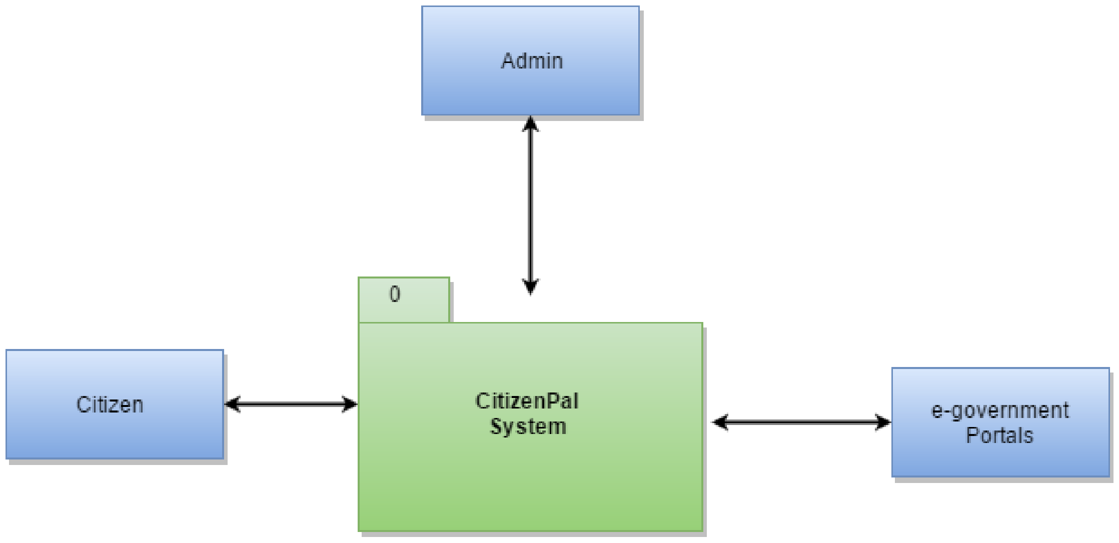


Figure 15: Data flow diagram, context view.

This diagram refers to first level diagram as it can be seen from figure15. The system receives and send information from all external entities that are involved. If we break down the level one we will have the following diagram the shows level 2 of the data flow.

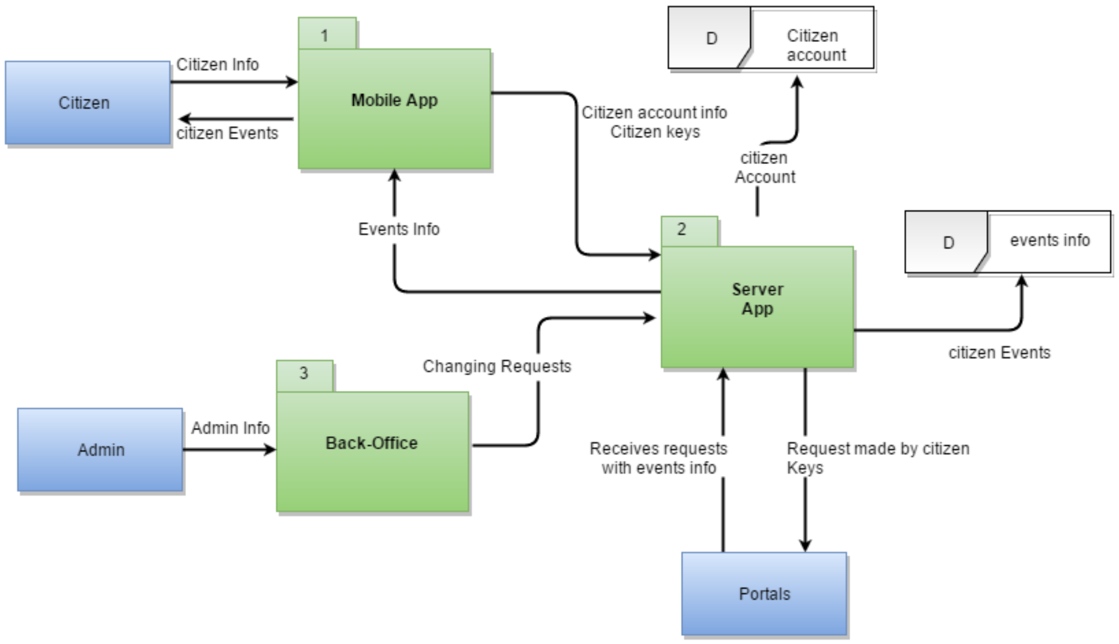


Figure 16: Data flow diagram, first level view.

As the figure 16 shows that the system consist of three main subsystems. First the data transfer from the mobile device to the server application that store the data to entities that are referred to **D** such as citizen account info. Then the server send requests to the public portals to the retrieves information related to citizen events. The server receives the responses analayze the requests. Extract the events info then store it to data event entity attached to user account. This event sends to mobile app after that. Then, mobile application can notify and show citizen events data.

In addition there is and back-office subsystem. this subsystem can send requests to change data by the admin. The admin should provide his credentials to back office. Finally, The back-office app can send request to server to perform the changes.

The diagram 16 above gives and idea how the data is flowing throw the subsystems. That was important to understand the CitizenPal System.

3.4 Non-functional Requirements

3.4.1 Security Requirements

Since the project transfers data throw the Internet, security is crucial issue. There are several major security features should be considered as security requirements:

- **Authentication:** authentication usually refers the process of determining whether someone or something is, in fact, who or what it is declared to be. To ensure the authentication is really met, the application should authentication services that provide both machine and user authentications.
  - **Machine Authentication:** In order to assure that CitizenPal server is really the server of CitizenPal project. The project should use a digital certificate technology that signed by certificate Authority (CA) as figure 17 shows. However, using self-signed certificate also acceptable just in the prototype as section Prioritization points out. 3.5.

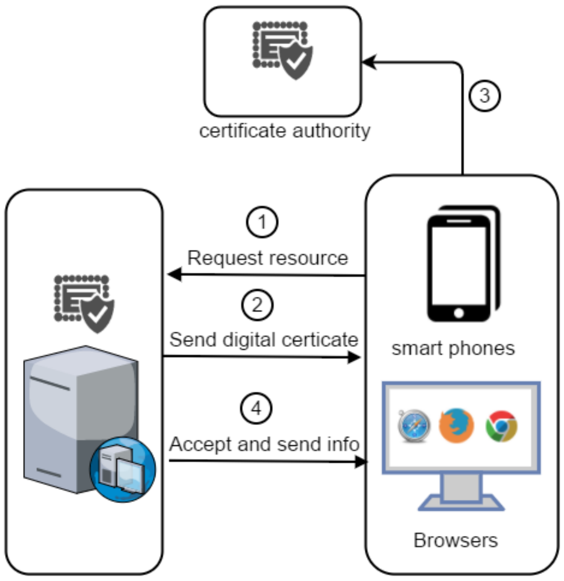


Figure 17: Authentication using digital certificate.

  - **User Authentication**: The server application in CitizenPal project is going to be used by two types of users Citizens, and Admin. It should have an user authentication to determine who is requesting and apply its corresponding authorization policy. This should down using email and password. Since the connection between the client(Admin or citizens) is stateless in RESTful API, every request comes to the server should carry its own authentication keys. Clients should be authenticate using email and password. So, every request should carries out its email and password. as the diagram below shows.

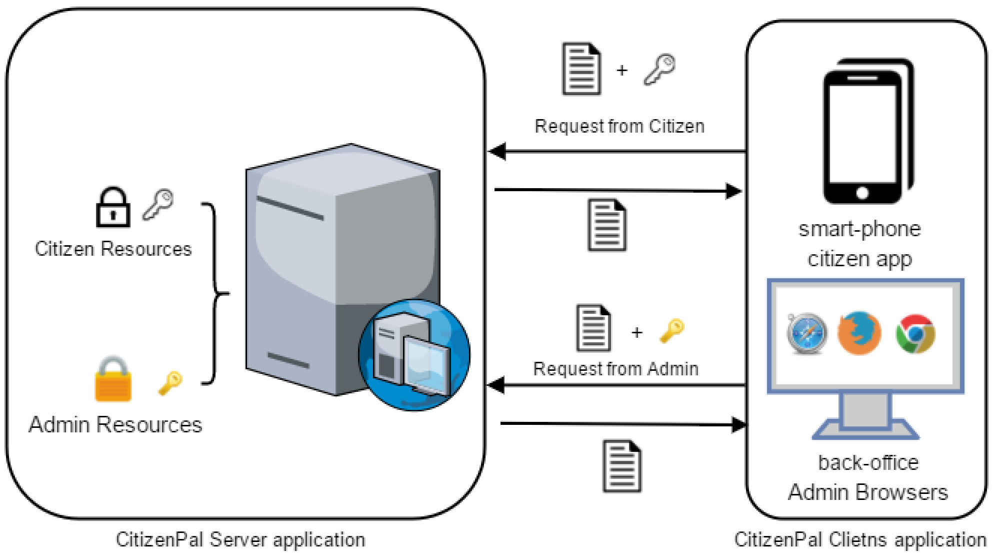


Figure 18: User authentication using email and password.

The requests that does not have correct password and user email will not give access to these resources. For more about the Http request of server application see Implementation section 4.3.1

- **Encryption:** encryption is a key factor in this project as well, Two issues should be required in the project:
  - **Secure Data Transmitting:** since the requests built on HTTP protocol. The project should use HTTPs which assure the SSL encryption over the http protocol. The previous digital certificate should bind a cryptographic key. Typically, SSL is a trade off tool used to secure the data transmitting.
  - **Encrypt Sensitive Data:** data such as user password should be stored hashed using one of secure hashing algorithms. and should not be retrieved at all.
- **Secure User's Input:** In order to assure that the data is stored in a correct form, all data from user should pass throw data validation process. This process should prevent SQL-injection attacks. For the prototype, validation process could be on both client and server side.

3.4.2 Designing Requirements :

This section, presents the standards that should be met in designing and developing the programs in this project. Primary benefit of adhering to software standards is efficiency such as increasing performance, reducing the maintenance time and Reducing business risks. CitizenPal project should use the following standards throw the design and developments of the project.

- **RESTful standard:** Designing restful API is an art. There are several methods and techniques used by lead companies in designing there APIs such as facebook and google and twitter. Every company uses his own method. CitizenPal should used the a common standard getting benefits from there APIs. [11]
- **MVC standard:** Is a software design standard for developing web applications. A Model View Controller pattern is made up of the following three parts Model, View, and controller.

This model should be used in designing web-client application. AngularJS has a good structure to use this pattern in the development.

- **Notification Service:** The mobile device should show up notification when ever there is a new event is adding to the citizen (requirement SP-UC-11) 3.3.1. Examining the designing requirements of service is important before starting to know which one is proper solution. There is two approaches for pushing notification service:

  - **Firebase Cloud Messaging :** Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost. Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to users. This service is sending and an instant messages to user's smart-phone app. [12]
  - **Alarm Manager Service**: Another method that might used to implement the notification service to use alarm manager that can run the background of android user. This service has to stay running in the background even if the application is closed, more details 4.4.2 .

  CitizenPal Smart-phone should use one of these approaches to implements the notification services. Comparing the complexity and the needs of the project, it will be much suitable to use Alarm Manager Service because of :

  - It is not necessary for the citizen to receives this notification for the event instantly. So the application can check for new event once a day. FCM might be more useful in services that needs more instance messaging.
  - Unlike Alarm Manager Service, FCM needs an account for the project on the service and need more configuration for the project. This will increase the complexity of the project.

3.5 Prioritization

This section is pointing out some topics that might be considered more important than the other in term of the first release of this project.

3.5.1 Use Cases Priorities:

Every use case has a priority level. "Essential" level should be applied in the prototype release. In addition, "Optional level" could be satisfied or could not.

3.5.2 Security Priorities:

All security requirements should be satisfied but there is some issues should be pointed out in this prototype release.

- Connecting to public authority portals: The prototype has no connections to public portals of e-portals of the government services. This part of the project should be on hold till accessing the portal is granted. Read more about this in Planning section 10.8.
- Self-Signed certificate used in the prototype release can be self-signed certificate instead of a certificate that signed by Certificate Authority.
- Validation: data validation process should be applied in both client side and the service side. This should be satisfied in prototype release.

4 Implementation

This section describes architectural components of the Citizen project. It gives an overview of implementation process and explains the mains architectural components of the project.

4.1 Components Overview

Considering the aspects that have been mentioned in the Requirements chapter 3. CitizenPal project is consisting of the following main following parts:

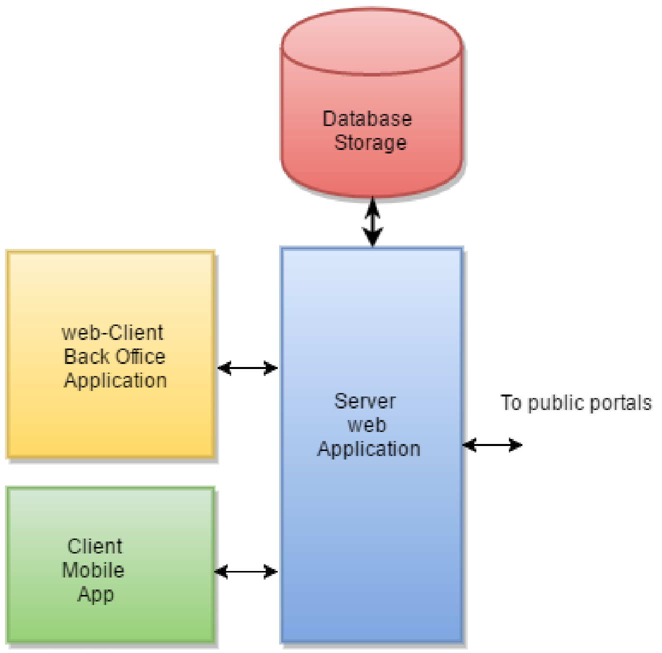


Figure 19: Main required applications of CitizenPal project

- **Server Application:** A server side application that fetches the data from the public portals, store the data in database, receives and replies requests from a smart-phone application and back-office application.
- **Smart-phone app:** a smart phone client application in which the citizen uses to access his events info.
- **Back-Office app:** a web client application that can manage the users events and portals info. It works as administration panel of the CitizenPal project.
- **Database:** a database that stores and saves the user's events and data. The database is interacting only with the Server App.

After building these applications, it is an important to integrate them together. The integration is a part of the implementation between server and smart-phone app, server and back-office app.

4.2 System Architecture

In order to explain the architecture, several levels of abstraction has been adapted. The abstraction levels allows to understand the project by breaking down its elements from high level to more low level. In this architecture, several terms will be used in order to have unambiguous understanding of the system. These terms are:

- **User:** an actual users of the application.

- **Container:** represents a group of components or a group of other containers that have a common feature of task. It could be application that consists of containers and components or, it could be a container that contains several components.
- **Component:** represents the lowest level of abstraction. Basically it is a file or class.
- **Relation:** An arrow titled by verb.

This approach of representing the architecture is created by Simon Brown for more details see [15].

Context Diagram

The context diagram represents the outside view of CitizenPal software system. CitizenPal [Software System] in dark blue block represents the system. Other blocks are the external systems in which the software system has to interact with.

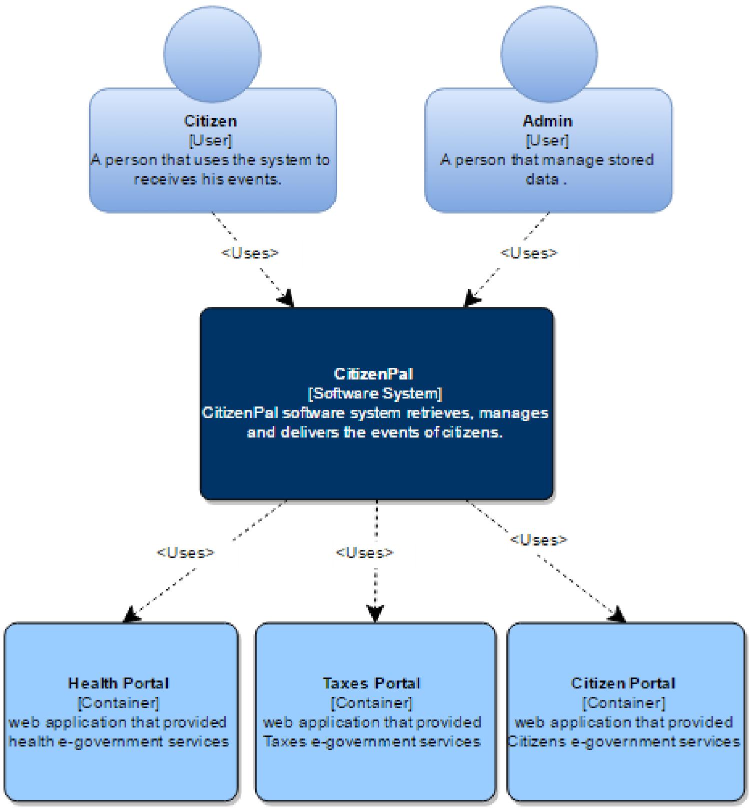

Citizen
[User]
A person that uses the system to receives his events.
Admin
[User]
A person that manage stored data .
<Uses>
<Uses>
CitizenPal
[Software System]
CitizenPal software system retrieves, manages and delivers the events of citizens.
<Uses>
<Uses>
<Uses>
Health Portal
[Container]
web application that provided health e-government services
Taxes Portal
[Container]
web application that provided Taxes e-government services
Citizen Portal
[Container]
web application that provided Citizens e-government services

Figure 20: Context diagram of CitizenPal system

The figure 20 represents mainly the users and the external entities that the system is using. As it can be seen, CitizenPal system used by the both a citizen and admin as Users. In addition, CitizenPal uses public authorities portals to request citizen information.

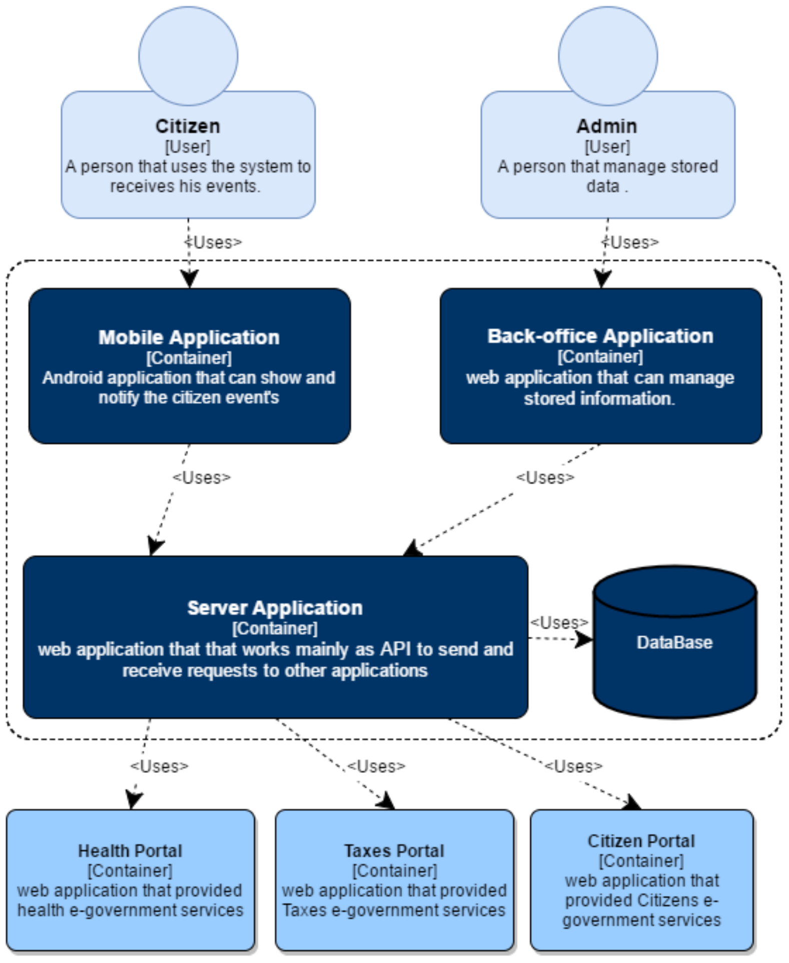

Citizen
[User]
A person that uses the system to receives his events.
Admin
[User]
A person that manage stored data .
<Uses>
<Uses>
Mobile Application
[Container]
Android application that can show and notify the citizen event's
Back-office Application
[Container]
web application that can manage stored information.
<Uses>
<Uses>
Server Application
[Container]
web application that that works mainly as API to send and receive requests to other applications
<Uses>
DataBase
<Uses>
<Uses>
<Uses>
Health Portal
[Container]
web application that provided health e-government services
Taxes Portal
[Container]
web application that provided Taxes e-government services
Citizen Portal
[Container]
web application that provided Citizens e-government services

Figure 21: Context diagram of CitizenPal system with more details

CitizenPal [software system] block consists of the containers that are represented in figure 21. In addition, it specifies the interaction of these containers with users and with portals. In this case, admin uses back-office [Container] and citizen uses mobile-application [Container]. Finally, as it can be seen from the figure 21 server application [Container] uses the public portals. In the following sections, these three containers will be explored with more details.

4.3 Server Application Architecture

Server application is the core of this project. It has been built as Java Dynamic Web application. Server application has to run on Apache server. In general, Server application is running and waiting for a requests from the client (back-office app or mobile app).

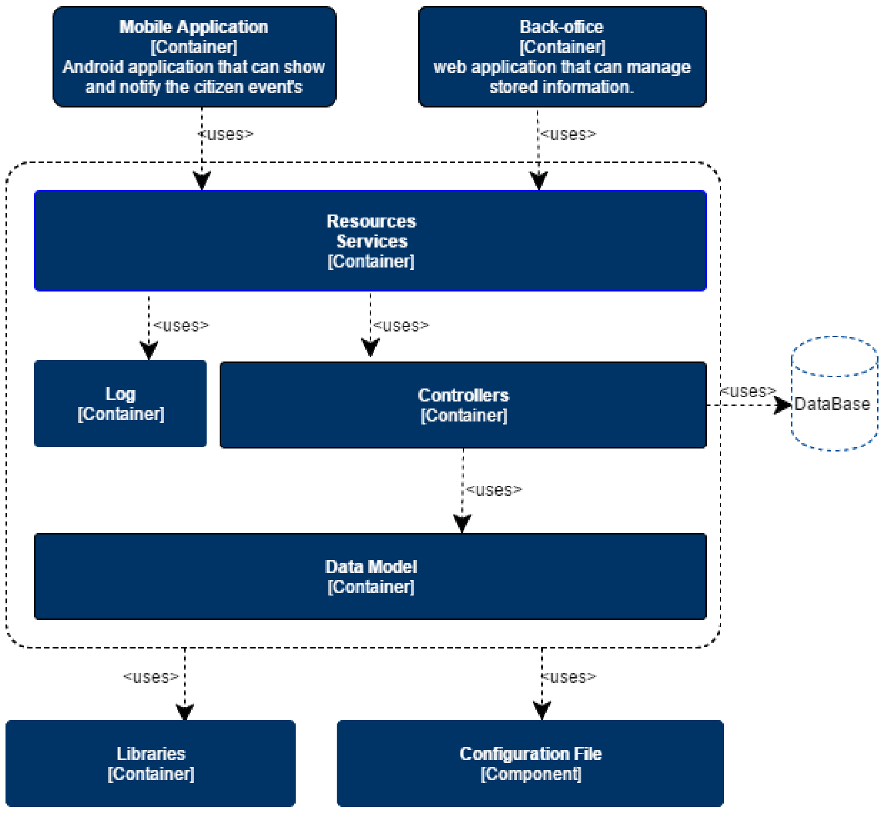


Figure 22: Architecture of server application

Figure 22 shows the architecture of Server application (inside a doted box). Server application consists of several blocks as following:

1. Resources Services [Container]

2. Controllers [Container]

3. log [Container]

4. Data Model [Container]

5. Libraries [Container]

6. Configuration File [Component]

As it can be seen in figure 22 , the controllers in server application uses Database to store and retrieve the information. In addition, it uses the libraries and configuration file. In the following, we will examine these containers in more details.

- **Resource Services** contains all classes that responsible for RESTful API. Basically, they have been built based on RESTful theory. The following figure 23 shows the components of Resource Services.

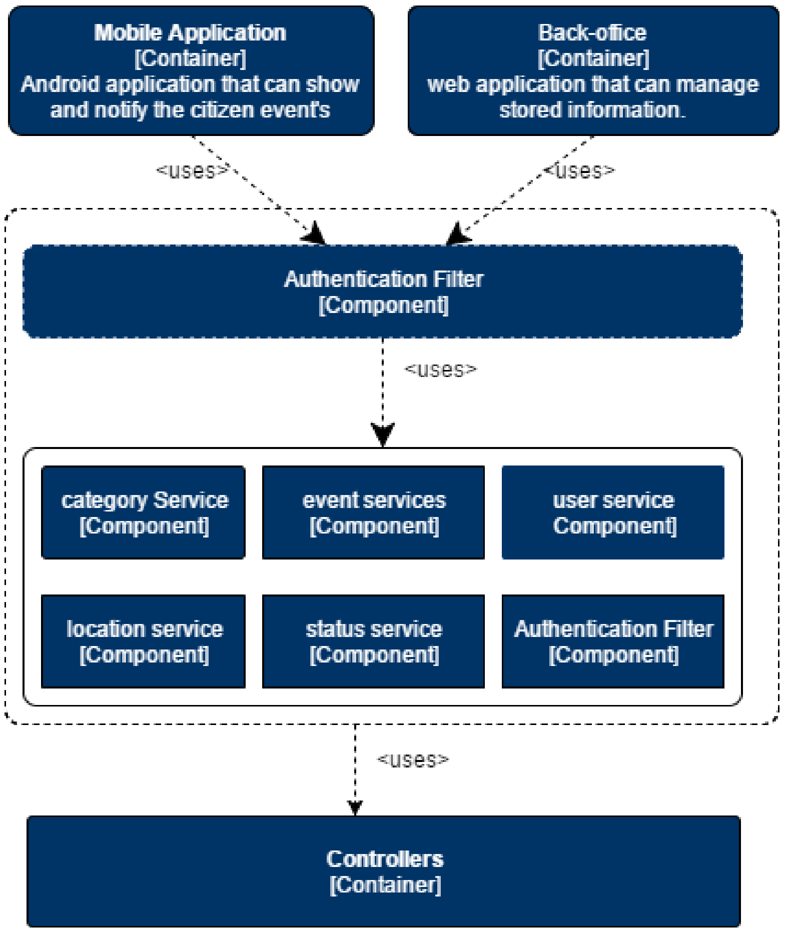


Figure 23: Resource services architecture

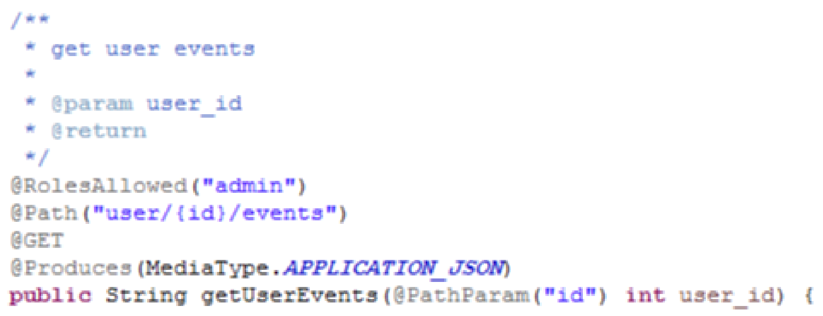
Figure 23 shows the components of resource services container (inside doted box). It has the classes that that represents RESTful API services such as user service [Component], events service [Component], location service [Component] and status service [Component]. It uses controllers [Container]. In addition to that, it has an authentication filter [Component]. Authentication filter can filter the requests and give the authorization that is needed based on client type and the resource service.

Every service has several functions. These functions provide the resources. The resources represented as URLs. URL in Jersey Library (Java library that has been used) refers as a path. A path could be assigned to specific resource state. Basically, Every Resource has the following properties:

1. **Path:** A URL that is used to access the resource by the client. Usually, it refers in Jersey API as @Path("/resource-name"). Naming these URLs has to be based on the standard that has seen earlier in Chapter 3 , Designing Requirements 3.4.2 .
2. **Method:** An HTTP method that used to refer to the resource's action. There are five methods GET, POST, DELETE, PUT and HEAD. These methods determine the behavior of the resource. Also they represented by annotations as following @GET, @POST, @DELETE, @PUT and @HEAD.
3. **Input and Output:** Resource sometimes consumes or produces some data. These data should be carried out with body of the HTTP request or by HTTP response. Usually it represents by @Consumes("Data-Type") and @Produces("Data-Type").

4. **Role:** a name represents the authorization policy. Since CitizenPal has just admin and citizen as a user, Resources should have a role policy such as "admin" for admin user of the back-office. In addition, citizen users resource is having the default role from the authentication Filter. Role represented as annotation @RolesAllowed("role-name").

Example of resource below :



```
/**
 * get user events
 *
 * @param user_id
 * @return
 */
@RolesAllowed("admin")
@Path("user/{id}/events")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String getUserEvents(@PathParam("id") int user_id) {
```

Figure 24: RESTful resources properties annotations.

All resource properties can be seen clearly in the figure 24. In addition to that, some times it is important to send parameter as part of URL query which can be done using @PathParam as figure 24 showed. It should refers to the parameter's name with brackets
{id}
inside the URL's path.

To browse full list of all Resources of CitizenPal API, see Appendixes at API Resources section.10.6 .

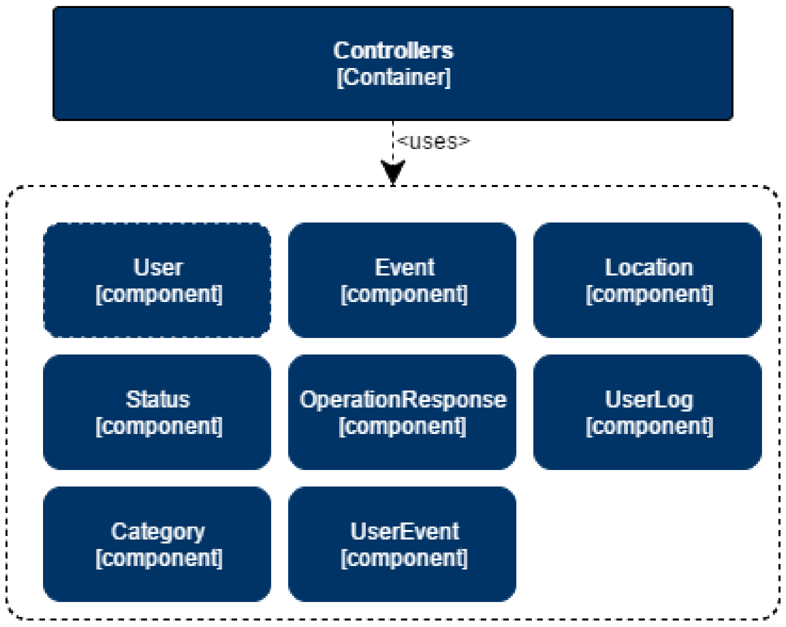


Figure 25: Data model components of server application

- **Data Model** contains Java Beans of the data model. Java Beans are Java classes. These classes represent the objects that should be send and received on the requests. They clearly can be seen in figure 25 inside the doted block.

- **Controllers:** contains several classes which can control and manipulate the data between the resources services [Container] and data model [Container]. It responsible for connecting to database, data validation and data conversion. Its components showed inside the doted block of figure 26. as following:

  1. **Sql Functions** [Component]
     A class contains all functions that are needed to perform Sql operations. Clearly, it needs to use the Data Connection, Validation and Conversion components as figure 26 shows.
  2. **Data Connection** [Component]
     A class that responsible database connection configurations such as opening and closing the connection with the database.
  3. **Data Conversion** [Component]
     A class the contains several function the responsible for converting between the data types and and format.
  4. **Data Validation** [component]
     A class that responsible for validating the data and verify its format and type.

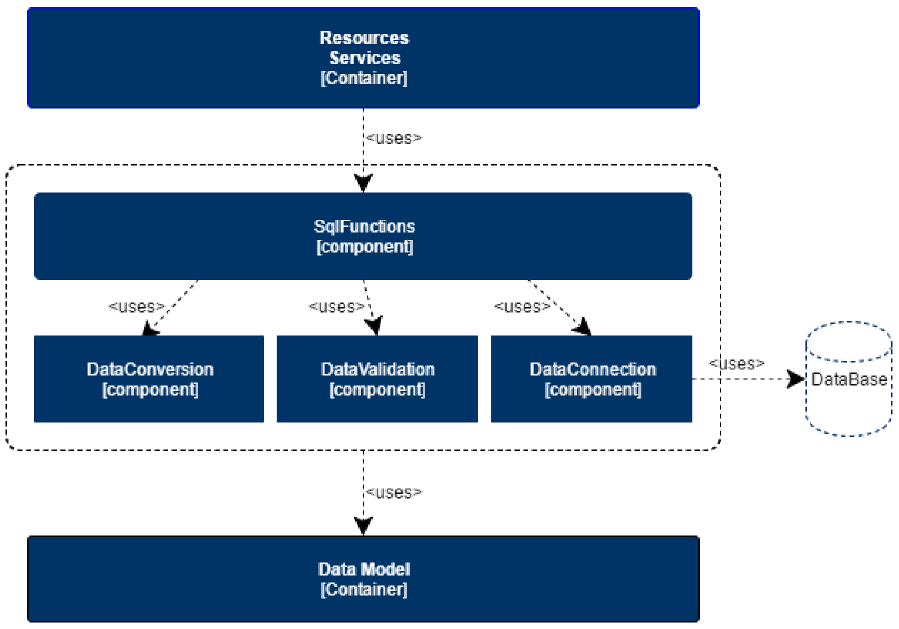

Resources Services [Container]
<uses>
SqlFunctions [component]
<uses>
<uses>
<uses>
DataConversion [component]
DataValidation [component]
DataConnection [component]
<uses>
DataBase
<uses>
Data Model [Container]

Figure 26: Server controller components

Controllers used by Resource Services to perfrom the Sql Operations. It uses Data model as we can see from figure26.

The rest of the **Server Application** blocks are:

- **Logs** a component that responsible of logging the requests, errors , and activities to external file.

- **Libraries:** a container showed in the bottom of figure 22. It Contains all Java libraries. Java Libraries are JARs files that used in for the development of the server.

- **Configuration file** basically, it is the configuration file in Java Dynamic web project is XML format file. usually called web.xml. web.xml Contains properties and configurations of the project features. It can be seen in bottom of figure 22 as well.

4.3.1 HTTP Request:

All HTTP requests that are asking a resource in CitizenPal API passes through authentication filter. This filter determines if this request is allowed to pass to the service or not. This filter allows Server application to authenticate every request. Also it specifies the authorization policy. Authorization policy specifies if this request allowed to access this resource or otherwise it denies the access.

In order for a request to be accepted the request should carry an authentication header with email name and password. The email and password should be Base64 encoded and added to header with authorization tag. Authorization tag in header should have the following format:
Headers['Authorization']="Basic {email} : {password}"
The authentication filter is retrieving this header and check if the user email and password is correct. The Admin has a static email and password, they should match with the ones in the header. Then the authentication filter check the resource role and decide if permit access or not. The same thing happen for the citizen request, except that the filter is retrieving the email and password from the database by using one of controllers classes.

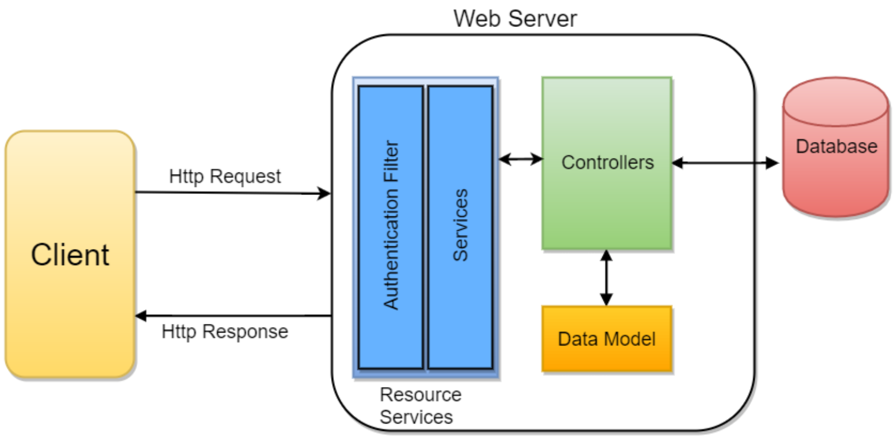


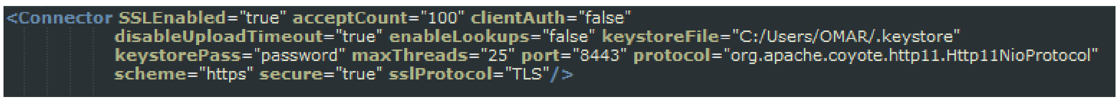
Figure 27: Http request resource process.

If the request is not authorized. Authentication filter denies the request and send HTTP respond with code 401 which means "Unauthorized".

The successful CitizenPal response always is using OperationRespose Object. it can be seen in Data model figure 26. OperationResponse is an object designed to carry the respond of the requests. OperationResponse consists of three attributes:

- **Result:** an integer number that determine if the operation is performed successfully. 1 means it success, 0 means not success or error.
- **Output:** an output of the operation. Usually it is one of the Data Model Object.
- **Causes:** a string data explain the case of error of not successful operation.

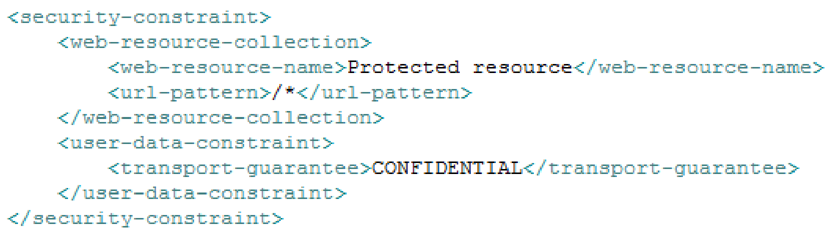
To achieve the security authentication, It is important to configure Apache server to accept the HTTPs. This can be done by adding some changes to server.xml in {Apache-Location} after generating the a self-signed certificate for it.



```
<Connector SSLEnabled="true" acceptCount="100" clientAuth="false"
           disableUploadTimeout="true" enableLookups="false" keystoreFile="C:/Users/OMAR/.keystore"
           keystorePass="password" maxThreads="25" port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           scheme="https" secure="true" sslProtocol="TLS"/>
```

Figure 28: Ssl over Http and specifying the key store file location

Another Important step is to configure the server application to use a secure connection. which can be done by adding a configurations lines inside web.xml configuration file.



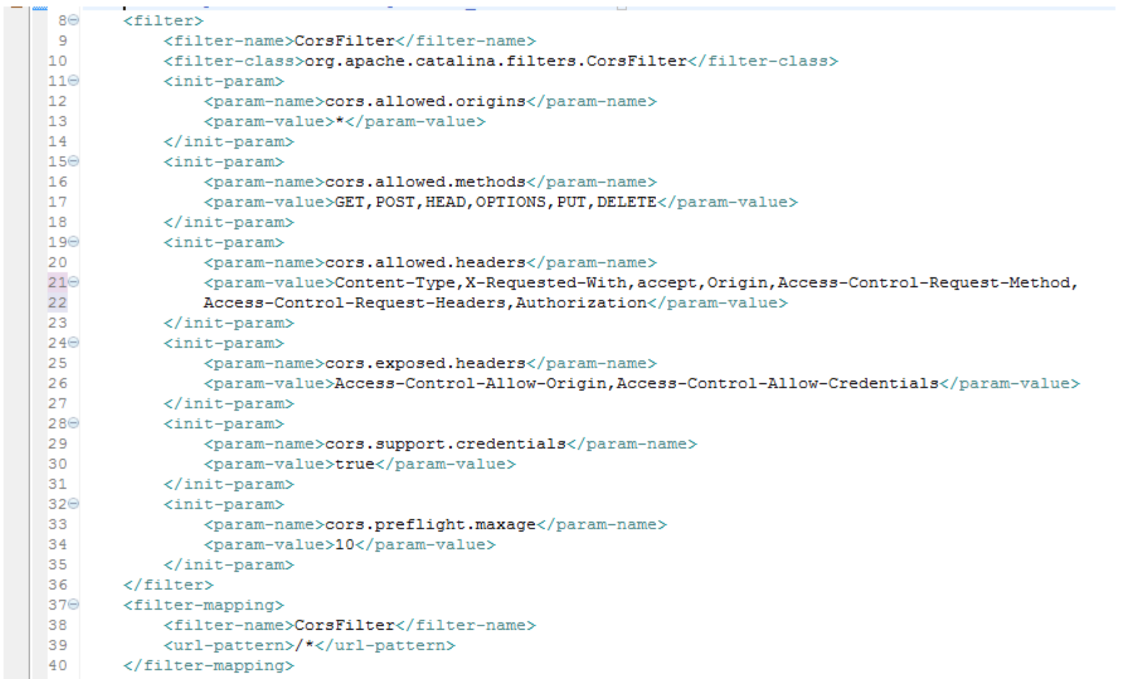
```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Protected resource</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Figure 29: Configuring secure connection and secure resources

The url-pattern defines which url needs to be protected, and it can given a name as well. transport-guarantee tells the application that protected transport layer connection HTTPS should be used to transmit the data. It has three options CONFIDENTIAL, INTEGRAL, or NONE. Server uses CONFIDENTIAL to prevent other entities from observing the contents of the transmission. for more info related to other options [13].

4.3.2 Enabling Cors

In order for the web back-office application to send and receives requests from Server application, CORS should be enabled. CORS stands for **C**ross-**O**rigin **R**esource **S**haring which is a specification that open access across domain-boundaries. Because of the simple-origin policy prevents the JavaScript application to make requests across domain boundaries. Simple origin policy has been developed in early time of web browsers. It allows a script from one page to access content of other pages only if they have the same origin. the same origin means same URL, port number and host. In this case, since the back-office can been hosting in different server. It will not allowed to access another resource out of its boundaries. Fortunately, Apache 7.0 support CORS standard and can be activated. To activate CORS on Tomcat Apache, Server app should be configured to accept cross filter by expressing that in web.xml as it can be seen in below lines.



```
<filter>
    <filter-name>CorsFilter</filter-name>
    <filter-class>org.apache.catalina.filters.CorsFilter</filter-class>
    <init-param>
        <param-name>cors.allowed.origins</param-name>
        <param-value>*</param-value>
    </init-param>
    <init-param>
        <param-name>cors.allowed.methods</param-name>
        <param-value>GET,POST,HEAD,OPTIONS,PUT,DELETE</param-value>
    </init-param>
    <init-param>
        <param-name>cors.allowed.headers</param-name>
        <param-value>Content-Type,X-Requested-With,accept,Origin,Access-Control-Request-Method,
        Access-Control-Request-Headers,Authorization</param-value>
    </init-param>
    <init-param>
        <param-name>cors.exposed.headers</param-name>
        <param-value>Access-Control-Allow-Origin,Access-Control-Allow-Credentials</param-value>
    </init-param>
    <init-param>
        <param-name>cors.support.credentials</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>cors.preflight.maxage</param-name>
        <param-value>10</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>CorsFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Figure 30: Enabling CORS headers and methods

4.4 Android App Architecture

Android application works as client to Server application. CitizenPal Android app consists of the several components divided to three important layers. The following diagram shows these layers (inside doted block) 31. Android app is used by citizen user.

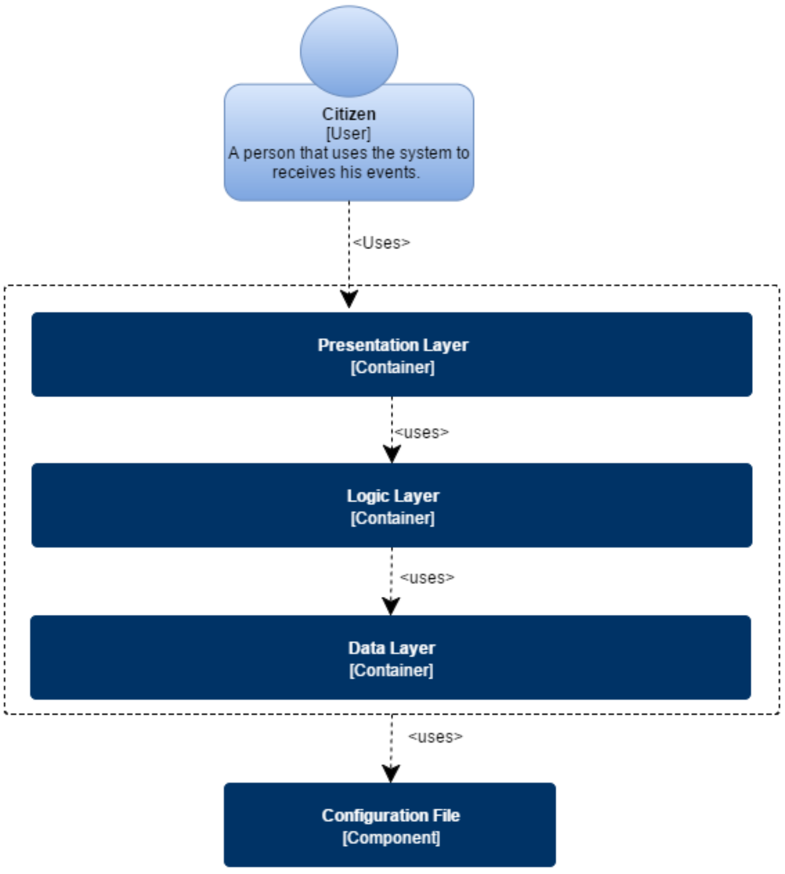


Figure 31: Architectural layers of CitizenPal smart-phone app

Android Application can be structured as these layers:

1. Presentation Layer [Container]
2. Logic Layer [Container]
3. Data Layer [Container]

Android Application is used directly by with citizen User and the application uses a configuration file as it can be seen in figure 31 Android app layers are going to be explained in more details in the following sections.

4.4.1 Presentation Layer

This layer consists of files that represents the design and the layout of the user interface. It contains all activities layout files. Activities layout files are XML files that represents the design of the interfaces components such as labels, Text Inputs, and Buttons. Figure 32 shows the presentation layer inside the doted block. Also, it contains another resources that are needed for the layout such as :

- **Strings**: a file contains all the text names represented in the layout xml files.

- **Drawable**: a folder contains images and photos.
- **Values**:a file contains parameters that used in layout xml files.
- **Menu** : a file contains a menus elements used in layout as well.

Theses elements usually has a name works as ID and it can be used any whenever they needed in the layout XML Activity file. Activity file is window or interface in the Android app. Users of Android app usually interacting with the activity as it can been seen in the figure 32.

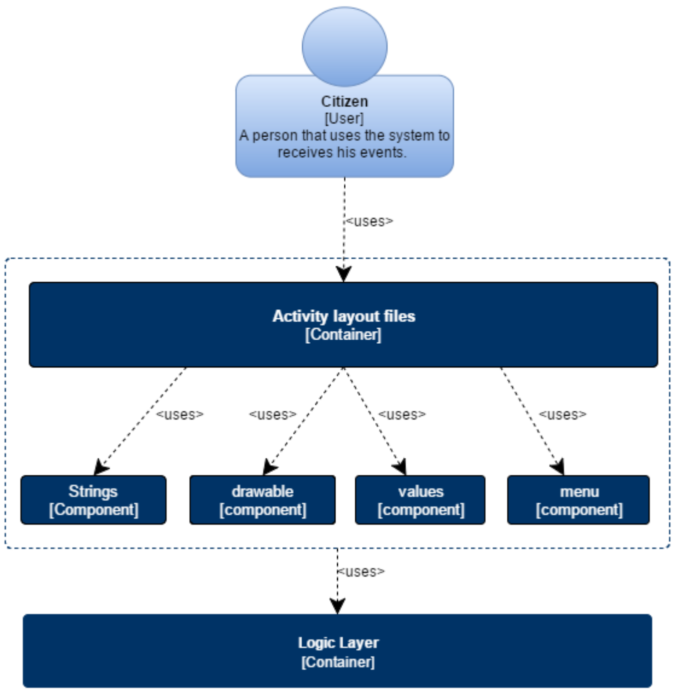


Figure 32: Presentation layer of CitizenPal smart-phone app

The Activity layout files [Container] consists of several XML files that represents the design of the android layout figure 33.

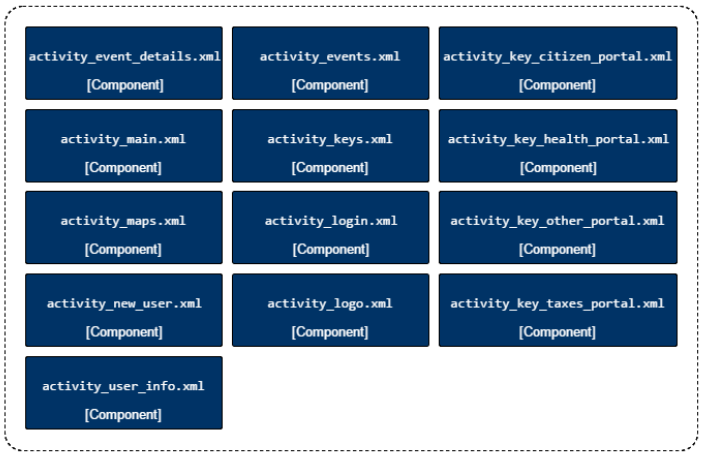


Figure 33: All activities of CitizenPal layout architecture

4.4.2 Logic Layer

Logic Layer controls app behavior and actions. In the logic layer there are the following containers that are showed in figure 34 inside doted block.

1. Activity classes [Container]
2. Async Tasks [Container]
3. Controllers [Container]
4. Notification Service [Container]
5. Restful Client [Container]

These five aforementioned blocks will be explained with more details in the following points.

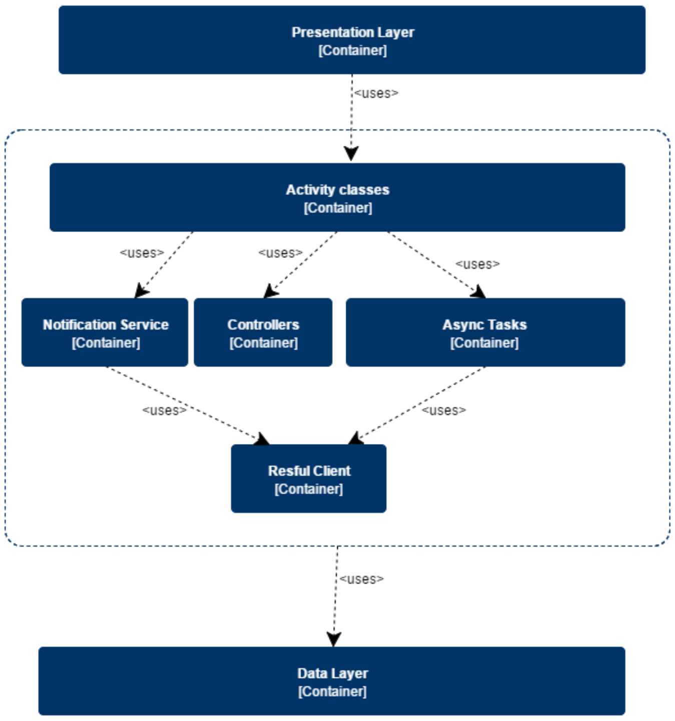


Figure 34: Logic layer architecture of citizenPal smart-phone app

These containers can be categorized to two types Foreground and background. The Foreground components usually runs in the main thread of the CitizenPal application. It usually drives the actions and performs the tasks of the activities. It has several classes such as controllers class, Data Model class and RESTfull clients class. In other hands, Background components contains the components that responsible of running the long term tasks or actions that need to connect with Server application to fetch the data. Lets go through them with few more details:

- **Foreground:**
  - **Activity Classes:** Every Activity layout(xml file) has Java class that controls its life time. The activities are the major usage of other components in the logic layer.
  - **RESTful Client:** consists of the classes that responsible of building the HTTP RESTful requests, authenticate and receives the response. Figure 35 showed its components inside doted block.

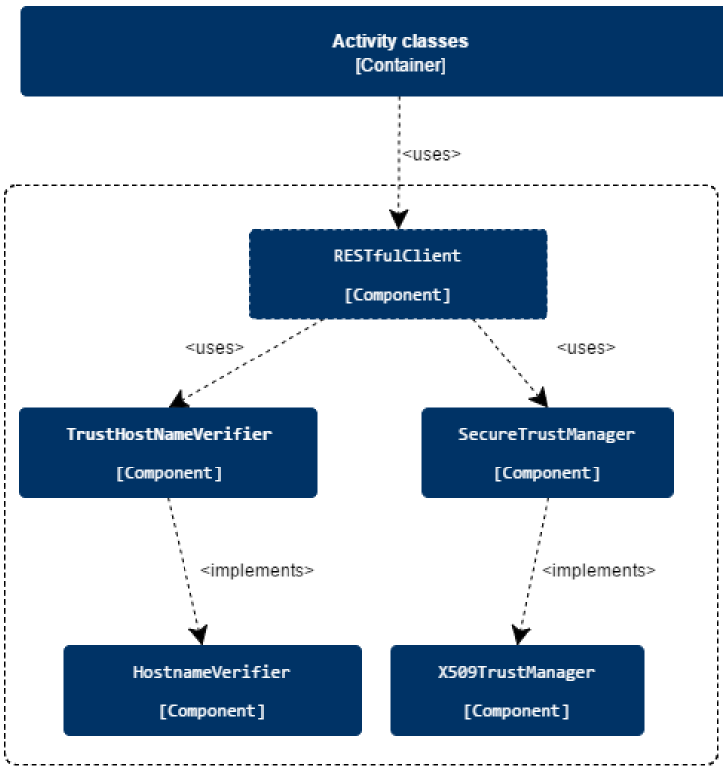


Figure 35: Restful client architectural components.

In the normal scenario when the certificate is signed by known Certificate Authority. RESTful client will verify the identity of the server by CAs and connection will be accepted. But since the Certificate that the server in CitizenPal that has been used is self signed certificate (which means the server is behaving as its own CA), RESTful client does not trust this certificate. Then, it will give an Exception during the SSL hand shake process.
Fortunately, this can be overcome by creating a secure TrustManager, A TrustManager is what the system uses to validate certificates from the server. TrustManager allows to trust the server certificate directly. This way has some downsides [14], but it can be done securely.

In addition, during ssl handshaking, if the URL's hostname and the server's identification hostname mismatch, the verification mechanism can call back to implementers of HostnameVerifier interface to determine if this connection should be allowed. In this case it will choose the default which is to terminate the connection.

– **Controllers:** contains foreground classes in which that are related to some actions such as storing, conversion and encoding data and storing. it also has some classes that manage adapting the lists layouts. figure 36

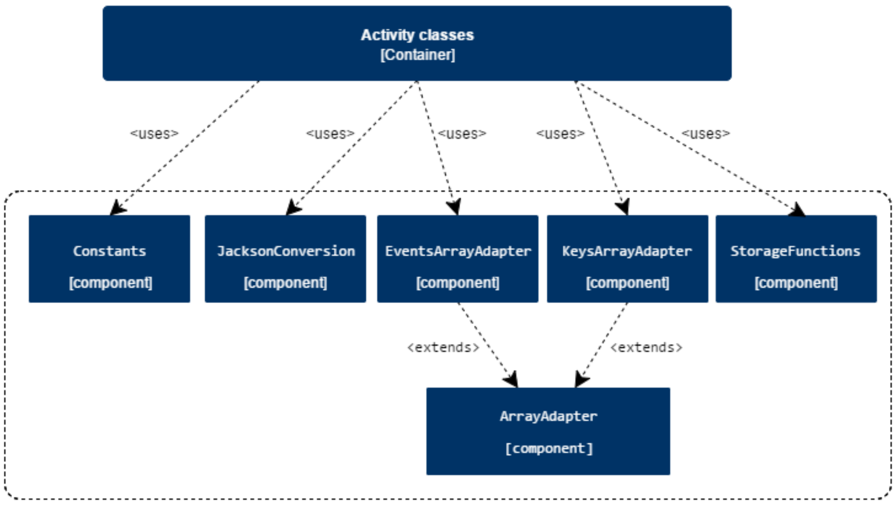


Figure 36: CitizenPal smart-phone controllers architectural components

- **Background:**
  - **Async Tasks:** contains asynchronous tasks that are performing in the background. These tasks are usually responsible for performing the RESTFul requests that connect to CitizenPal server and retrieve the information. Android recommended that the tasks that needs long term execution time should be performed in individual thread. So it will not create the Application Not Responding (ANR) situation. Figure 37 showed theses tasks inside doted block.

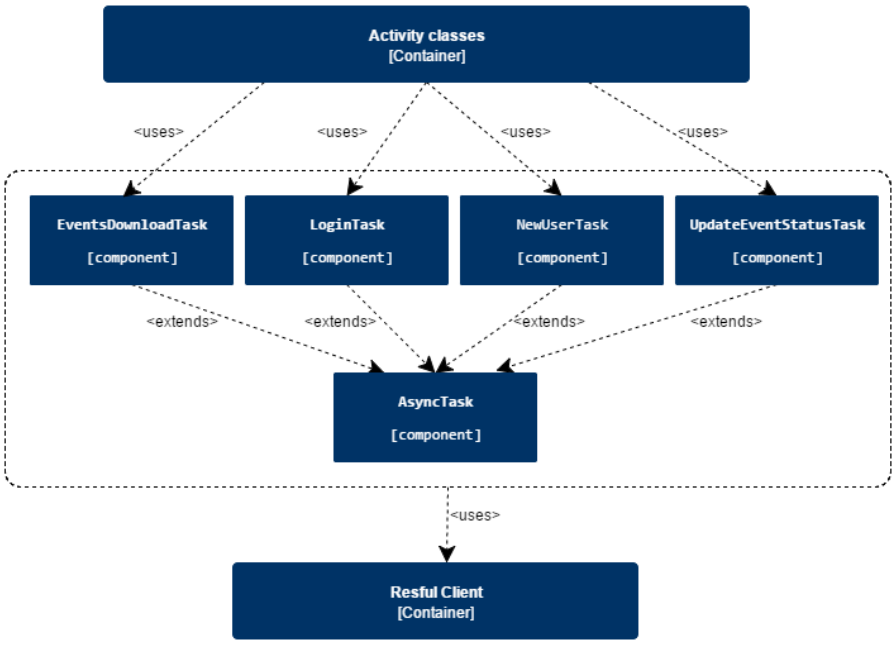


Figure 37: Asynchronous tasks architectural components

    Activity classes use the async tasks in to ask or send requests to the server. from other hand, async tasks uses the RESTful client classes to make the requests.
  - **Notification Service:** contains the classes of notification service that is running in the background even if the application is closed figure 38 showed the components of notification service.

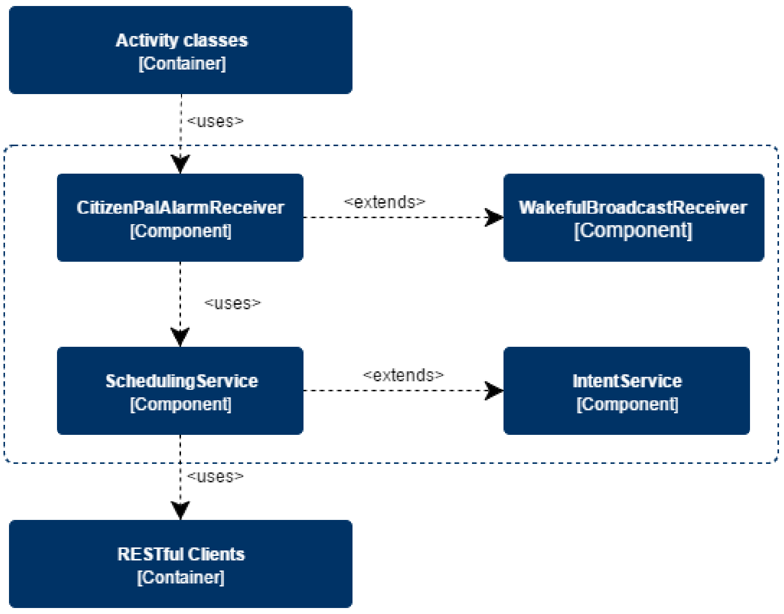
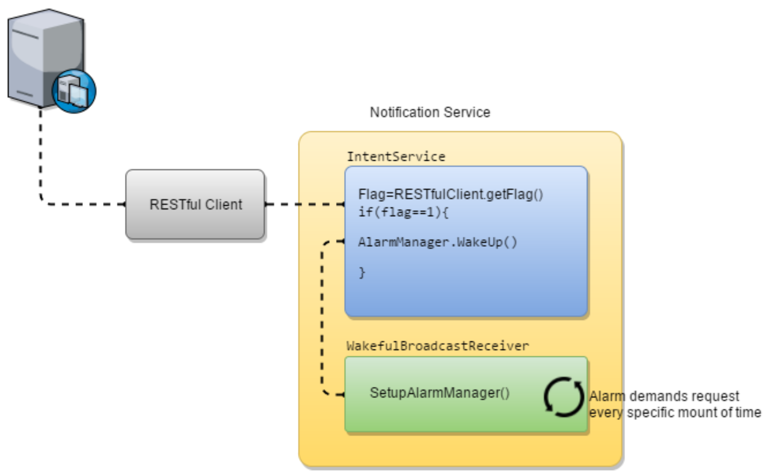


Figure 38: Notification service components diagram.

Figure 38 shows that, it has two major components for the notification service:

* CitizenPal Alarm Receiver: which is java classes that runs when the scheduling time happen and wake up the android device.
* Scheduling Service: is a components that responsible to schedule the tasks and perform the task. It extends the indent service which is an asynchronous tasks.

The notification service uses AlarmManager that are provided by Android SDK. AlarmManager allows the app to run a background task. This task can be still running even if the application is closed. It can wake up the device as well. The following figure 39 depicting the process of the notification service:



```
Flag=RESTfulClient.getFlag()
if(flag==1){
AlarmManager.WakeUp()
}
```

Figure 39: Notification service process

As it can be seen, IntentService is a service that handles asynchronous requests on demand. The Alarm Manager allows to schedule a specific task every period of time. Alarm Manager ask to perform IntentService when the time accrues. IntentService requests the resource flag from CitizenPal Server. Then it checks the flag if the flag equal 1, this means that the user has new event. And the device should show a notification and wake up the device. Waking up the device is supported by the Alarm Manager as well.

The resource flag, is and integer number linked to every Citizen account. This resource flag is changed to 1 when new event added to this citizen account. When the citizen open the notification and see the events, this flag is changed to 0 to refers that this user does not have new events.

4.4.3 Data Layer

layer that contains the components where the data is stored . The data abstraction layer consists of two major components. SharedPreference storage where the data citizen events stored. Another is the Certificate trust storage where the x509 certificate stored.

4.4.4 Configuration File

Android Application has a very important configuration file called AndroidManifest.xml. The manifest file provides essential information about the app to the Android system such as permissions, names , parameters of the app. Since the CitizenPal will use Some Resources of the device, It needs Permissions too. CitizenPal shall use the following Resources:

- **Maps**: because it needs to specify the location event (where the event is taking place)
- **System Account and Calender:** because it needs to schedule the event System user calender account.
- **Internet:** because it it has to download the Events information from the CitizenPal server.
- **External Storage:** to store the information that retrieved from CitizenPal server.

4.4.5 Screens and Their Relationships

Screens relationships diagram represents the major navigation relation between the windows (Activities) in the CitizenPal android application.

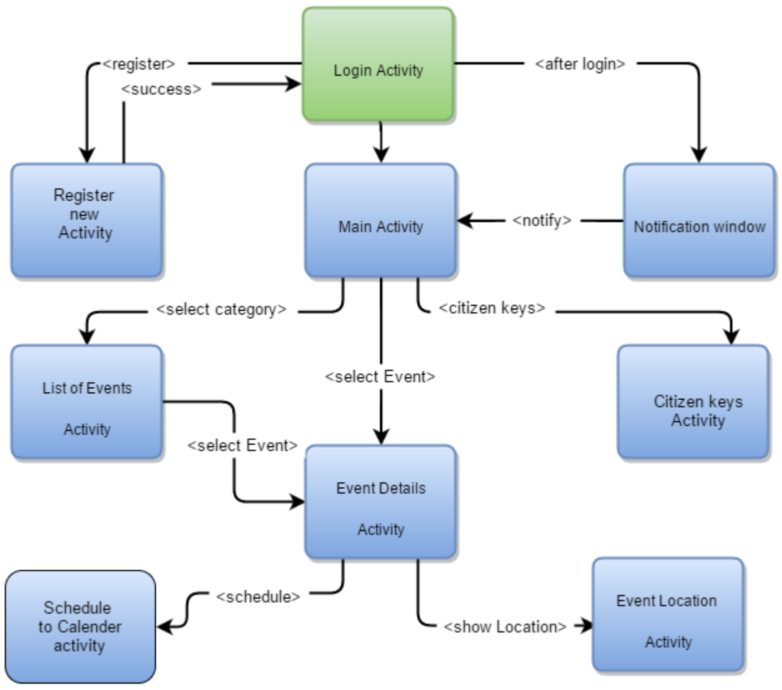


Figure 40: Screens relationships diagram

4.5 Back-Office App Architecture

4.5.1 SPA Software Pattern

The back-office is another part of the project. It controls and manages the data. Mainly, it built as Javascript web application using AngularJS. AngularJS app is an Single Page Application (SPA). SPA allows us to redraw any part of the UI without requiring a server roundtrip to retrieve HTML. This achieved by separating data layer from the presentation layer by having data model layer that handle the data and view layer that read from the model.
SPA is loading the application with one single load from the server. This loading downloads all the application and its Libraries and never have another trip again to server. AngularJs Allows the app to have its own routing policy or structure.

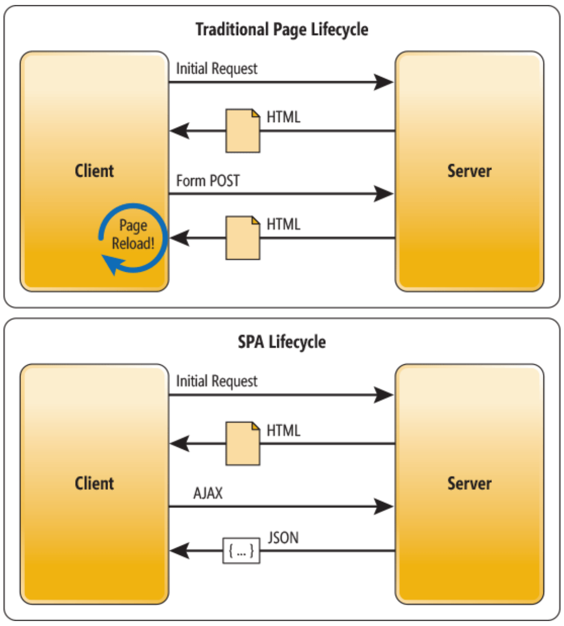


Figure 41: SPA lifecycle vs. traditional page lifecycle

CitizenPal Back-office is an SPA application. The following figure 42 shows the components of the Back-office application. The application is downloaded once with all his libraries from the citizenPal back-office server. It runs on any browser.
In addition, AngularJs allows the app to have its own routing policy which gives the application a more flexibility. Lets examine the the components from down to up. There is Views [Container] that uses App Configuration [Component] and controller [Component]. Controller uses both Data Model [component] and Authentication Service [Component].

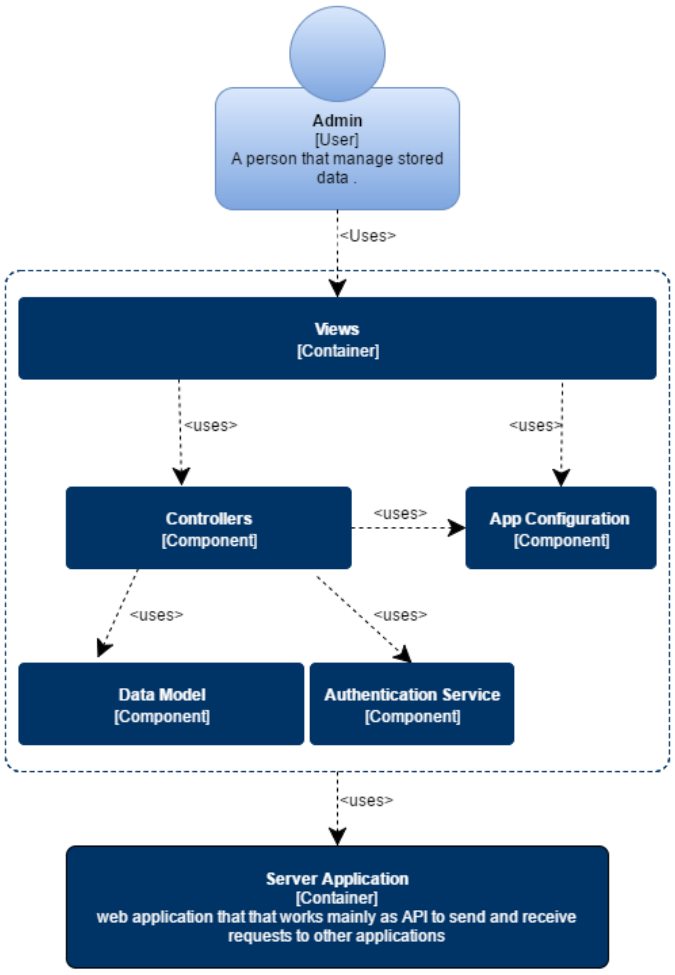


Figure 42: CitizenPal web-client Angular architecture components

4.5.2 Views

They are (User Interfaces) UIs that the user will interact with. Views are mainly a HTML forms. These HTML forms should be linked to specifics paths or routes using routing policy. The figure below gives an example how App Configuration component links that views with controllers and gives it a route path.

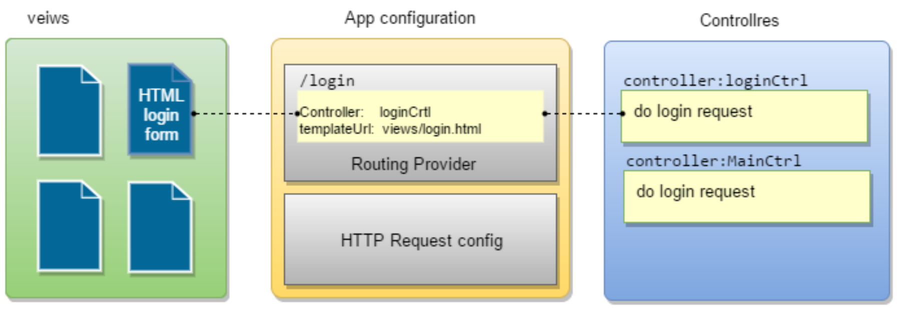


Figure 43: Controlling views by routing policy

As it can seen, the view “Login.html” has a linked controller “loginCtrl”. In addition, both of them (controller and View) have a routing path (routing policy) in this case “\login” .

4.5.3 App Configuration

App Configuration is a file that contains the configuration that linked with project during its executing. Mainly, it used to define several configurations for the app such :

1. **Routing Policy:** As it has from the example in figure 44 routing policy is very important that links the controllers with views in AngularJS.

2. **Run configs:** It is important to setup some important components as soon as the application started to run such as cookies used and and some HTTP headers used during HTTP requests. Configs here are used to check the cookies that stored when the user success the login for every time the user perform any request in the app. The figure below explains the runs configs. When a controller tries to perform any action, the angularJs app runs the run configs part first.

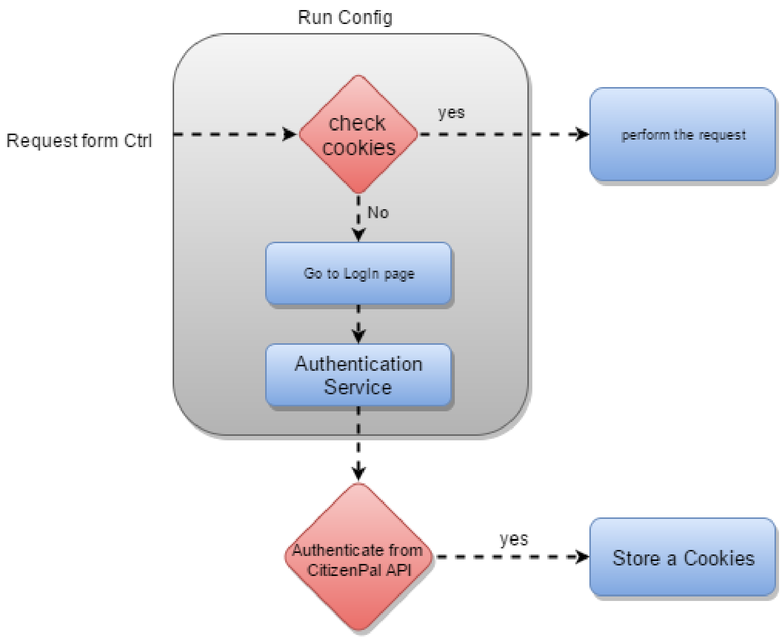


Figure 44: Back-office running configs logic

4.5.4 Controllers

It is an components that contains the actual actions and behavior of the application. It perform HTTP requests and drives the navigation between the views. It has several controllers.

4.5.5 Authentication Service

This is very important components. it is used to set up the authentication headers on the HTTP request before can be sent to the RESTful API Server.

4.5.6 Data Model

In JavaScript, the data is stored as JSON. AngularJS allows to bind the data to view elements. And any changes that happen in Date element is reflected on the View elements. This done using the controller of the HTML page controller.

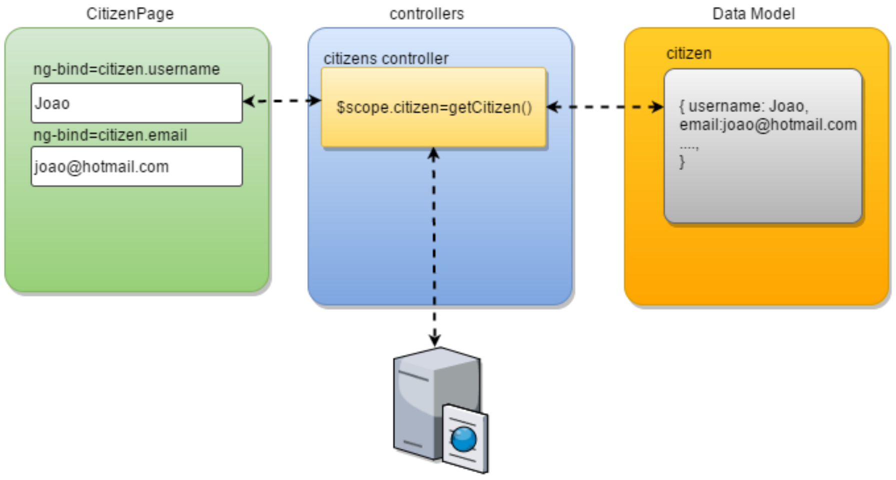


Figure 45: Data binding of html views element to data model

As the figure 48 shows, The controller retrieve the data by asking resources from API then stored as JSON object that binded to HTML element.

4.6 Database Diagram

Database is built using MySql database provider. Database is an individual part from RESTful API server and the Back-Office server. The database diagram is showed below:

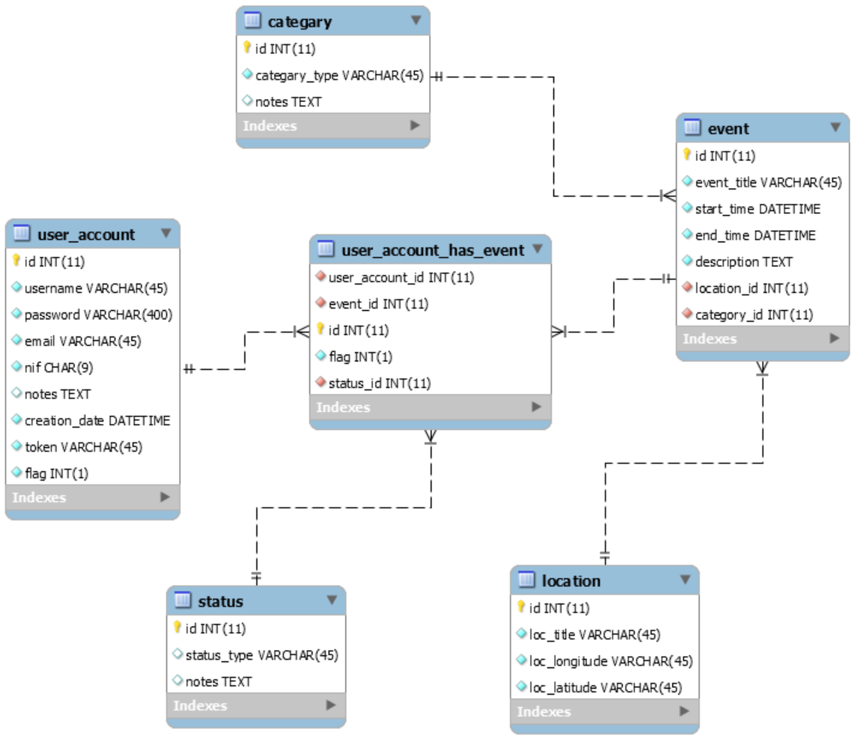


Figure 46: Database model entity relational diagram

4.7 Assumptions and Dependencies

This application is a prototype that presenting idea and functionality of CitizenPal proposal. In order to continue working and developing this product There are some few assumptions should be taking in account in this document.

- Accessing the public services of the government in Portugal is not granted yet. After having several meeting with AMA agency we decided to proceed in building the application with model the provide the ability to add and update events that can simulate the process.

- The current architecture is still working even after the accessibility to public services is granted. This can be accomplished by adding "proxy program " that allows the server app to connect with portals to get desired data.

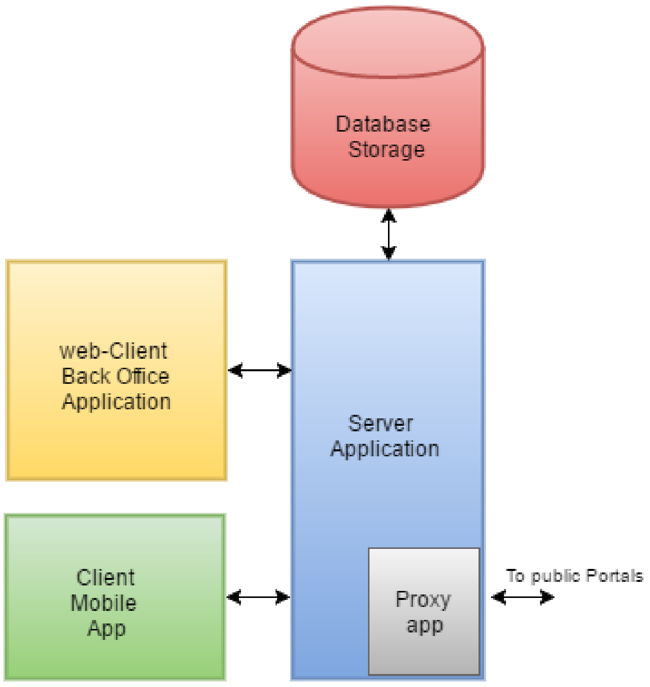


Figure 47: CitizenPal architecture with "Proxy Program" addition

5 Deployment

Deploying these project is essential tasks. This sections goes through the deployments process and the deployments specifications:

Deployment Specifications

To put these projects on work. Amazon EC2 is used for the deployment for both the API Restful server and Back-office server host. Amazon EC2 provides a free tier for all its clients. This free tier has been used to create two Virtual Machines with these specification :

- Microsoft Windows Server 2012 R2 Base
- 1 CPU , 1 GB RAM, 30GB Hard drive.

For Android Device, the applications is deployed on following phone:

- Phone: alcatel Pop C7 .
- OS: Android OS, v4.2 (Jelly Bean).
- Screen Resolution: 480 x 800 hdpi

It has been deployed on other projects as well. It could show some different related to resolution. The figure shows AMIs and there interactions

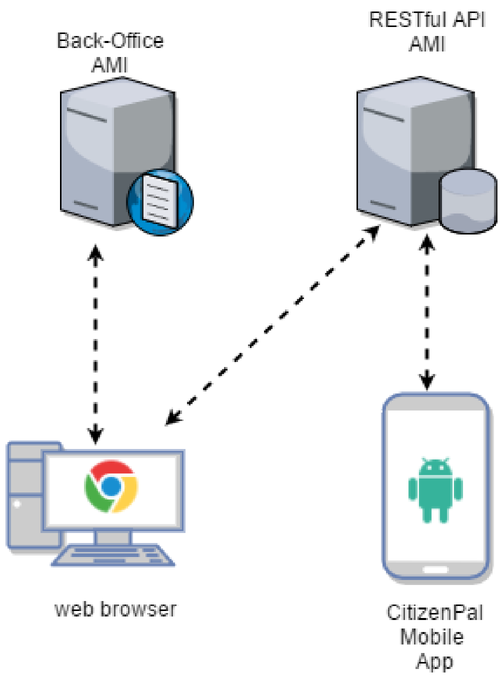


Figure 48: CitizenPal deployment machine diagram

The deployment steps has been explained in Appendix section 10.7.

6 Testing

This section explains the testing phase of the CitizenPal system. There are several types of testing can be done for the project such as :

- **User Acceptance Testing**: a set of test cases that shall be created for every functional requirement in order to be accepted.
- **Server Performance Testing**: a test that has been conducted in order to examines the actual performance respond of the server app when a large group of clients is performing requests and under a given configuration of infrastructure.

6.1 User Acceptance Testing

Acceptance testing is used to ensure that requirements are satisfied. This test will cover the functional requirements. In order to validate them, a set of input tests cases has been created. Every test case has the following properties:

Test Case ID	A unique numeric identifier for the Test Case
Use Case ID	A use case number that is referred in use case properties table 3.3.1 .
Description	Briefly description of the use case.
Pre-Condition	The previous conditions that should be satisfied for executing the input tests data.
Acceptance	It determines if data test set is accepted or not.
Data Input:	set of input data cover the regular User input. • **#no**: An input step number. • **Test Data**: A set of data that required for the current test step. • **Expected Result**: expected results that the application should produce after executing the test input. • **Actual Result**: real result that the application produced after executing the application. • **Acceptance**: It determines if the data set test is accepted or not

Table 5: Acceptance test cases properties

In some cases the test cases does not requires and data set and that is stated in the cases. The full test cases can be found in 10.5 section at Appendixes.

6.2 Server app Performance Testing

The performance tests is crucial because the server is going to be used by huge amount of citizens. It should have a good performance for for clients. JMeter is a tool that can be used to test Apache servers app and measure its performance.

6.2.1 Test Plan

Test plan simulates the citizens http requests. In this test plan, a normal citizen usage example of the citizenPal api. The following diagram 49 explains the test plan:

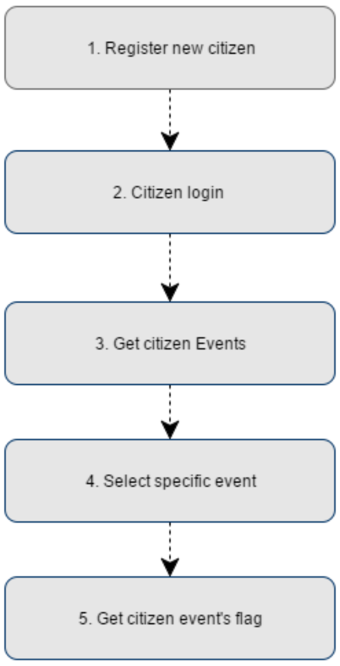


Figure 49: Performance test plan

To test the performance of the Server application, a test plan has been performed specific number of times. Every time is considered as user thread. After that measure the response.

Before start conducting the result, It is important to mention that the test plan has been performed on previous deployment specification 5 machines. The client is Jmeter application that placed in Coimbra, Portugal. The Internet connection that has been used is the enduroam university connection network.

6.2.2 Testing Results

This test will subject the Server Application to huge number of users and and measure the response. In this test, Server api app will be subjected to normal users requests then we will measure its response and discuss the results.
Normal load simulate the average expected numbers of users. In this case the test runs 1000 users.

#no	#Samples	**Min**	**Max**	**Average**	**Error %**	**Throughput**	**KB/sec**	**Avg. Bytes**
1	1000	0.1632	0.33705	0.2614	0.00%	34	10.54	317.5
2	1000	0.0362	0.23578	0.1257	0.00%	29.6	12.24	423
3	1000	0.0198	0.22436	0.1187	0.00%	35.6	104.38	2999
4	1000	0.0073	0.24815	0.0855	0.00%	38.3	25.49	682
5	1000	0.0024	0.1472	0.0448	0.00%	55.5	12.53	231
Total	5000	0.0024	0.33705	0.1272	0.00%	74.9	68.07	930.5

Table 6: Normal load testing summary results

The table 6 shows that 5000 requests has been send from 1000 users. Every one perform the previous 5 tasks. It shows the Minimum and the Maximum response time taken by the http request for every task. It also shows the Average response time and the percentage of failure requests. Throughput in the table represents number of requests per second. KB/sec stands for the throughput that measured in Kilobytes per second. Finally, Avg. Bytes represents the average size of the sample response in bytes. Min, Max and Average unites are minutes.

As it has seen from results, The avarage response time for task 1 (Register new user) most likely because its an insert and select query. It inserts the new user information. then it return the full record to the client. The tasks 5 (Get citizen events flag) has the lowest average and most Throughput. Because it is used to obtain 0 or 1 flag which is used to indicate if the citizen has new events or not.

In general, the server api can handle 1000 users without having and failure request as result indicate the Erros 0.0% for all 5000 requests.

7 Discussion and Future Work

Initially, this project is an prototype works as platform for citizens. It has been built with clear objective. The objective is to help the citizen manage easily his events that are related to public e-governments. The project takes two semesters. In the mean time, the project passed through obstacles in first semester. The main issues that it has faced during the planning was obtaining access to public authorities servers. After several meeting with AMA, postponement and delay have been sensed. The company responsible for the internship decided to change the direction the project and keep the objective the same. This alteration was made in order to present the concept and achieve its objectives. The main changes was to add the server application and back office application.

CitizenPal now can work as platform that controls Citizen events and give him an easy way to get informed for his new obligations. Meanwhile, this application can be integrated with e-government portal as soon as they give us access to their service.

Consequently, CitizenPal has been built by integrating three types of application. Smartphone app, RESTfull API server-side application and client-web application. The application has been built with recent technologies that are found. Using RESTful API for the server-side Application which considered lightweight and simple to handle a complex applications.

On other hand,the project included a building of Android-app which considered the most operating system used in the market. It has been developed by using Android Studio platform . Finally, building back-office as web-client application using AngularJs. AngularJS considered new Library has been build and supported by Google. Also it uses SPA software pattern which is started to be used recently for web application.
In the end, last challenge was the integrating these three types of application together with secure technologies such as HTTPs encryptions and crtificate authentication.

Ultimately, CitizenPal project and this internship was and opportunity to enrich the knowledge with several new technologies concepts that have been obtained during Master in software engineering.

This prototype is a system that could be the nucleus of a future larger project that includes most of the tasks that can be performed where the citizen in public services, the project could be developed in the case were given permission to access to public services for e-government. As previously discussed, there are several governments have done this to facilitate access to its services and increase citizens' confidence in its institutions . In this prototype, it's good to continue working in the future to improve and enrich the project through:

- **Mobile Performance Testing**: a test that can be conducted on smart-phone app to determine the effect of the app on device environment. such as network, data usage and buttery consumption.
- **Release Plan** It is important before the application can be available to all citizens, It is important to plan specific release schedule. Working on a release plan requires time and effort and the number of consultants and lawyers for data protection and information security. In addition to that, CitizenPal should be subjected to work of intensively testing. So as to reduce the errors and the possibility of failure.

8 Conclusion

CitizenPal was an a part of efforts to obtain the Master Degree In software Engineering. This effort has been done as internship at company named Tangivel. The project resulted to build the three typed main applications:

- RESTful API server-side Application.
- Android smart-phone Application.
- Web back office Application.

After building these applications, it was a great experience that has been gained to integrate between them so they can work together smoothly with security requirements. In addition, several software standard and patterns have been applied and used during the developments and design. Overall, these steps and applications has been planned, designed Then developed keeping in focus on the objective of making the life of citizen more easy and simple.

9 References and Bibliography

References

[1] (January, 2016) Statistical indicators reflect those of Eurostat http://appsso.eurostat.ec.europa.eu/nui/show.do

[2] (December , 2015) DIGITAL AGENDA FOR EUROPE
[DIGITAL AGENDA FOR EUROPE]https://ec.europa.eu/digital-agenda/en/news/eu-egovernment-report-2014-shows-usability-online-public-services-improving-not-fast

[3] (December , 2015) About ama , formal website for the Administration Modernization Agency.
http://www.ama.pt/

[4] (August , 2016) Jersey, RESTful Web Services in Java.
https://jersey.java.net/

[5] (August , 2016) Appache Tomcat
http://tomcat.apache.org/

[6] (August , 2016) Android Studio, The official Android IDE
https://developer.android.com/studio/index.html

[7] (August , 2016) Single page apps in depth
http://singlepageappbook.com/goal.html

[8] (August , 2016) Bootstrap
http://getbootstrap.com/

[9] (June , 2016) Smartphone OS Market Share, 2015 Q2
http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[10] (June , 2016) Number of devices running a given version of the Android platform
https://developer.android.com/about/dashboards/index.html

[11] (August , 2016) Web API Design— Crafting Interfaces that Developers Love
https://pages.apigee.com/web-api-design-website-h-ebook-registration.html?int_source=resources-main&int_medium=website&int_campaign=ebook&int_content=api-design

[12] (August , 2016) Firebase Cloud Messaging
https://firebase.google.com/

[13] (August , 2016) Specifying Security Constraints
https://docs.oracle.com/cd/E19226-01/820-7627/bncbk/index.html

[14] (August , 2016) Security with HTTPS and SSL
https://developer.android.com/training/articles/security-ssl.html#SelfSigned

[15] (August , 2016) Simon Brown's Software architecture as code
https://structurizr.com/help/about

10 Appendixes

This section will have the Appendixes of the project. Theses appendixes is and technical and business documents that explains the complete details of the CitizenPal Project. It includes the Uses cases, Flow charts , Requirements, and Test cases.

10.1 User Interfaces-Smart Phone App

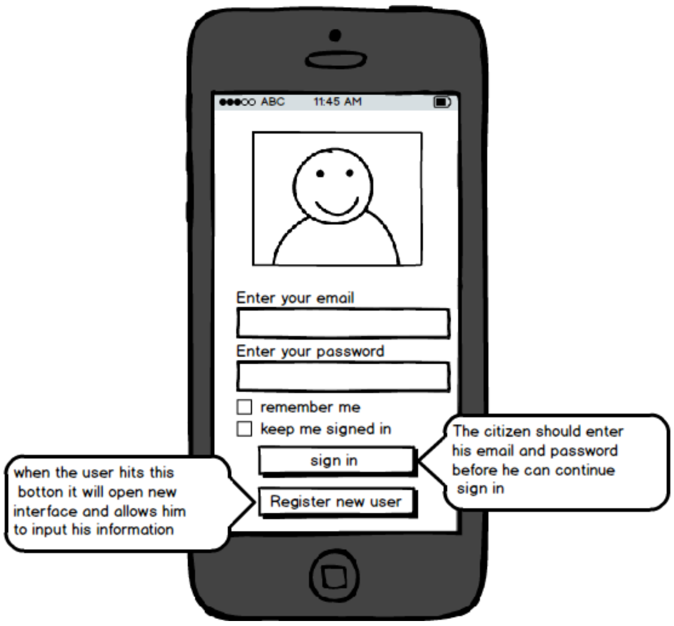

●●●○○ ABC 11:45 AM
Enter your email
Enter your password
remember me
keep me signed in
sign in
Register new user
when the user hits this botton it will open new interface and allows him to input his information
The citizen should enter his email and password before he can continue sign in

Figure 50: UI for CitizenPal smart-phone app, log in page

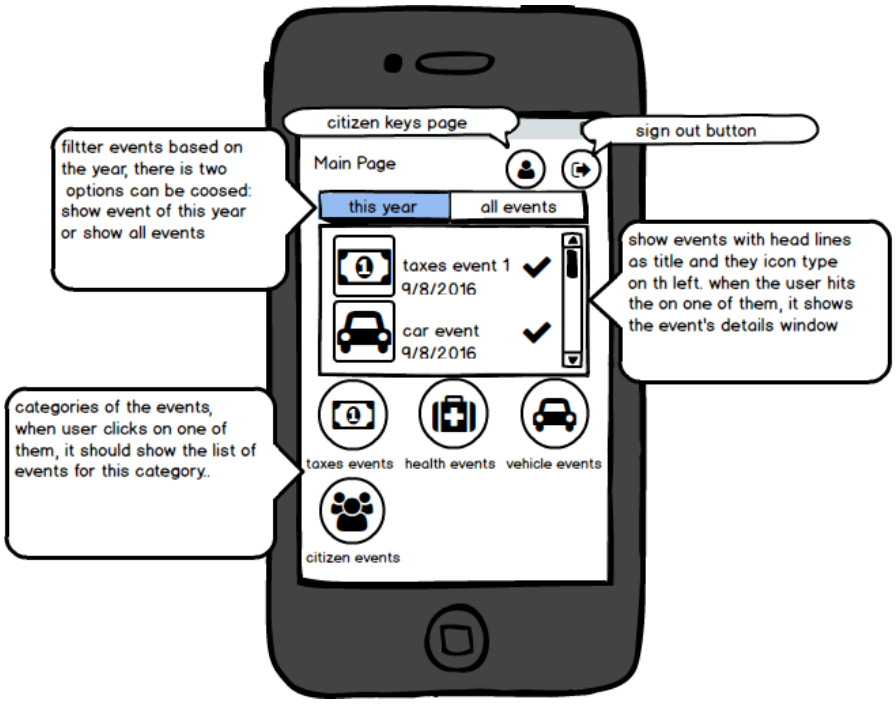

citizen keys page
sign out button
Main Page
this year
all events
taxes event 1 9/8/2016
car event 9/8/2016
taxes events
health events
vehicle events
citizen events
filtter events based on the year, there is two options can be coosed: show event of this year or show all events
show events with head lines as title and they icon type on th left. when the user hits the on one of them, it shows the event's details window
categories of the events, when user clicks on one of them, it should show the list of events for this category.

Figure 51: UI for CitizenPal smart-phone app,main page

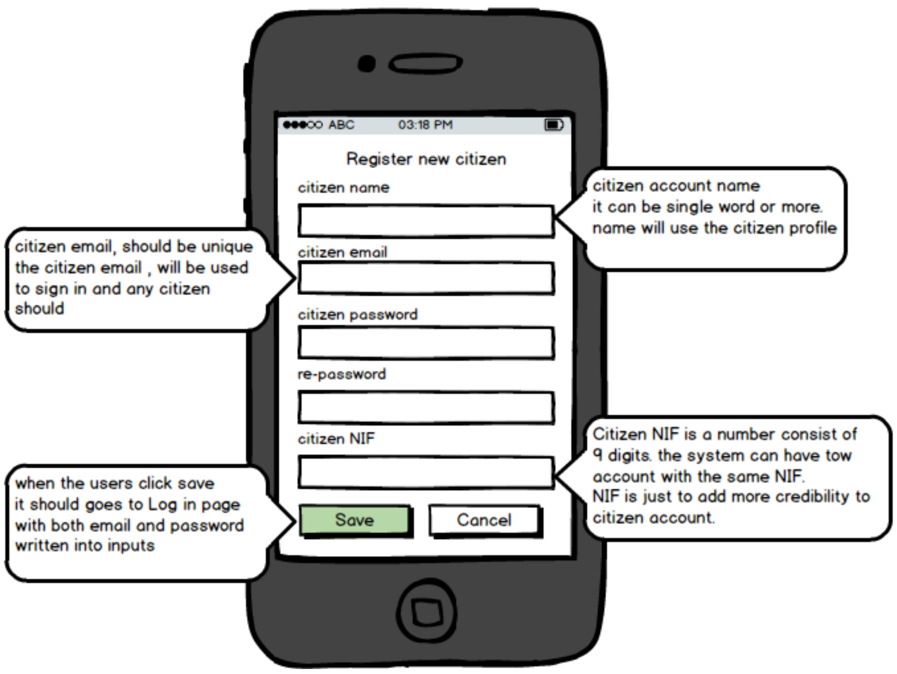

●●●○○ ABC 03:18 PM
Register new citizen
citizen name
citizen email
citizen password
re-password
citizen NIF
Save
Cancel
citizen account name it can be single word or more. name will use the citizen profile
citizen email, should be unique the citizen email , will be used to sign in and any citizen should
Citizen NIF is a number consist of 9 digits. the system can have tow account with the same NIF. NIF is just to add more credibility to citizen account.
when the users click save it should goes to Log in page with both email and password written into inputs

Figure 52: UI for CitizenPal smart-phone app, register new citizen page

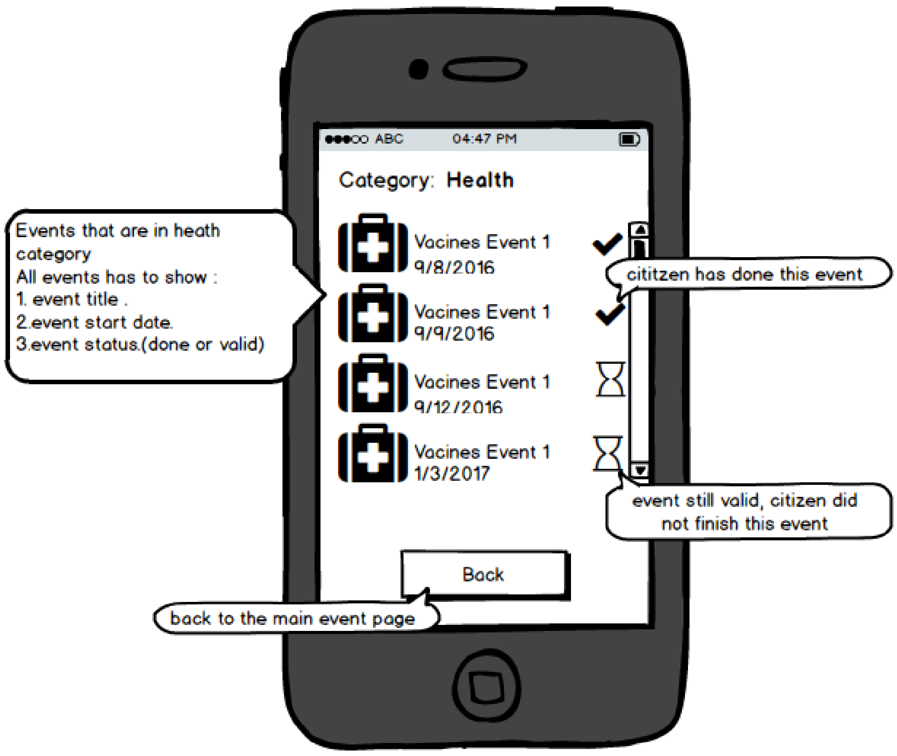

●●●○○ ABC 04:47 PM
Category: Health
Vacines Event 1 9/8/2016
Vacines Event 1 9/9/2016
Vacines Event 1 9/12/2016
Vacines Event 1 1/3/2017
Back
Events that are in heath category All events has to show : 1. event title . 2.event start date. 3.event status.(done or valid)
cititzen has done this event
event still valid, citizen did not finish this event
back to the main event page

Figure 53: UI for CitizenPal smart-phone app, categorize event page

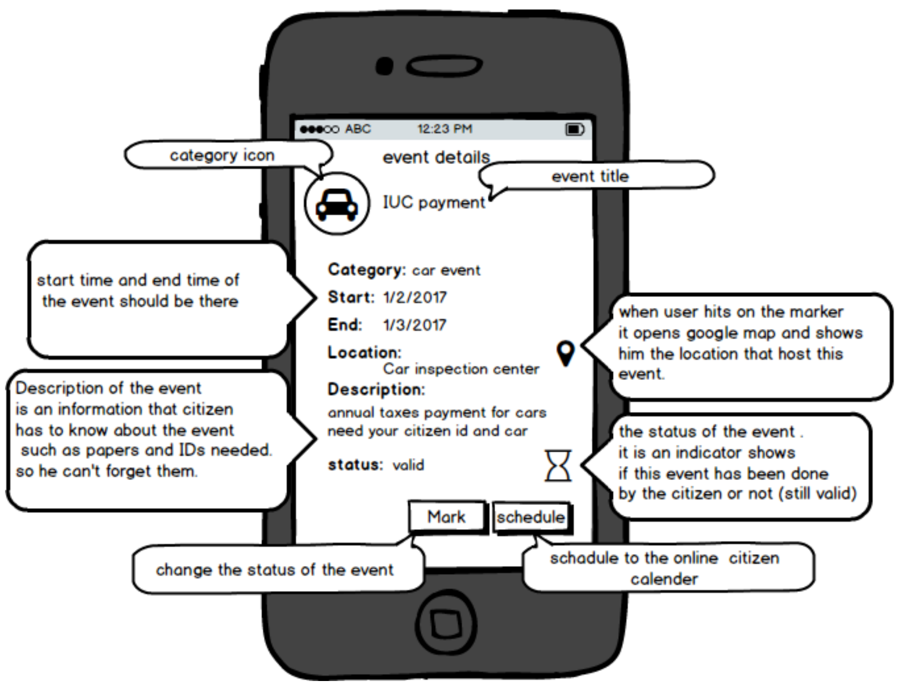

●●●○○ ABC
12:23 PM
category icon
event details
event title
IUC payment
start time and end time of the event should be there
Category: car event
Start: 1/2/2017
End: 1/3/2017
Location:
Car inspection center
when user hits on the marker it opens google map and shows him the location that host this event.
Description of the event is an information that citizen has to know about the event such as papers and IDs needed. so he can't forget them.
Description:
annual taxes payment for cars need your citizen id and car
status: valid
the status of the event . it is an indicator shows if this event has been done by the citizen or not (still valid)
Mark
schedule
change the status of the event
schadule to the online citizen calender

Figure 54: UI for CitizenPal smart-phone app, event't details page

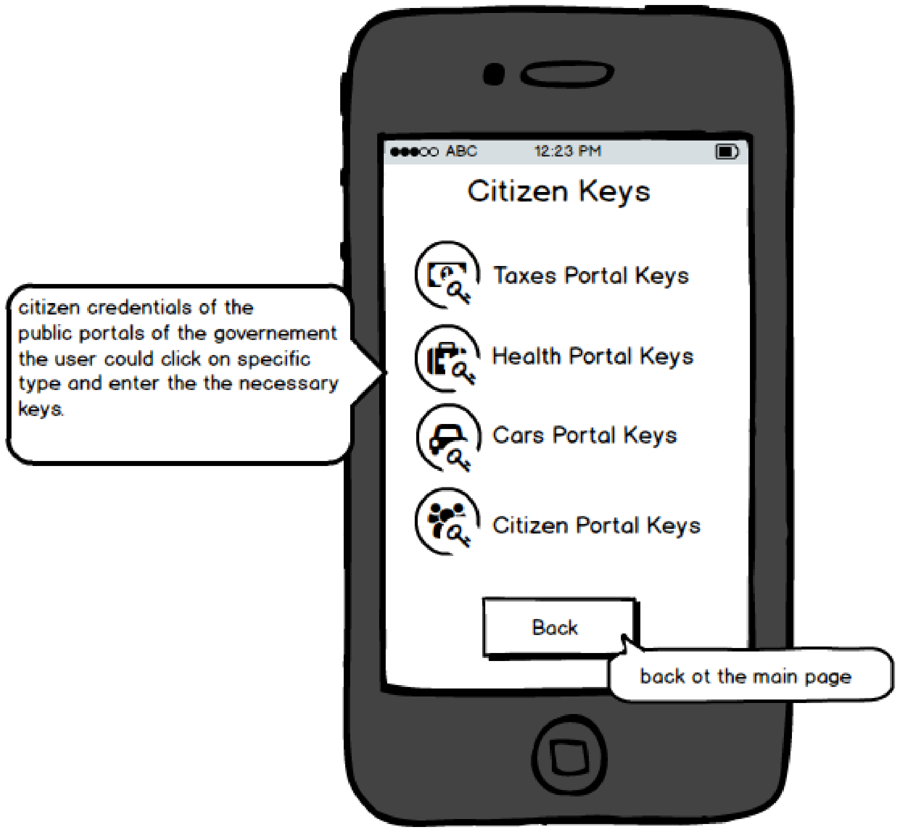

●●●○○ ABC
12:23 PM
Citizen Keys
Taxes Portal Keys
Health Portal Keys
Cars Portal Keys
Citizen Portal Keys
citizen credentials of the public portals of the governement the user could click on specific type and enter the the necessary keys.
Back
back ot the main page

Figure 55: UI for CitizenPal smart-phone app, keys entry page

10.2 User Interfaces-Back-Office App

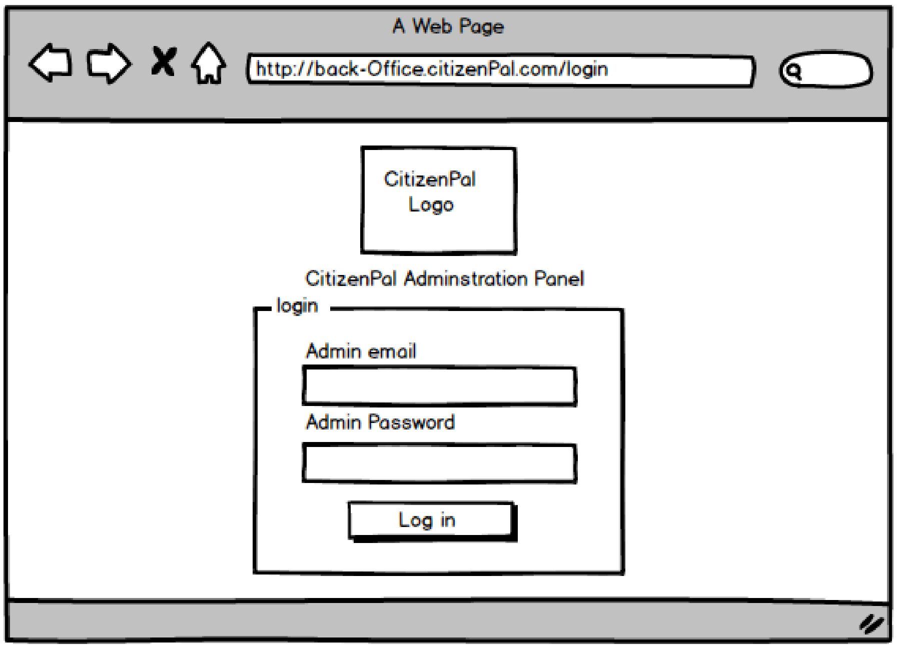

A Web Page
http://back-Office.citizenPal.com/login
CitizenPal Logo
CitizenPal Adminstration Panel
login
Admin email
Admin Password
Log in

Figure 56: UI for CitizenPal back-office app, Loging in page

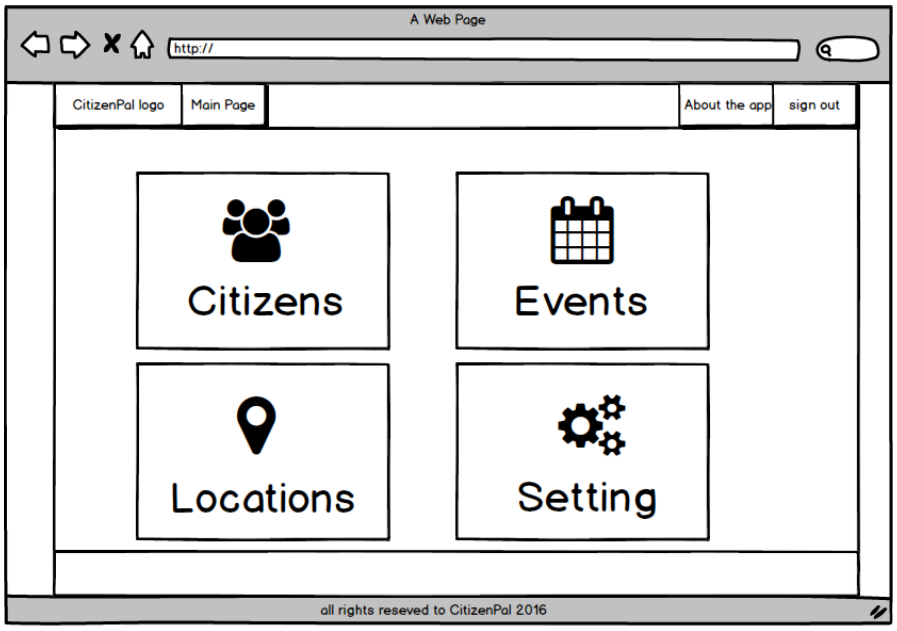
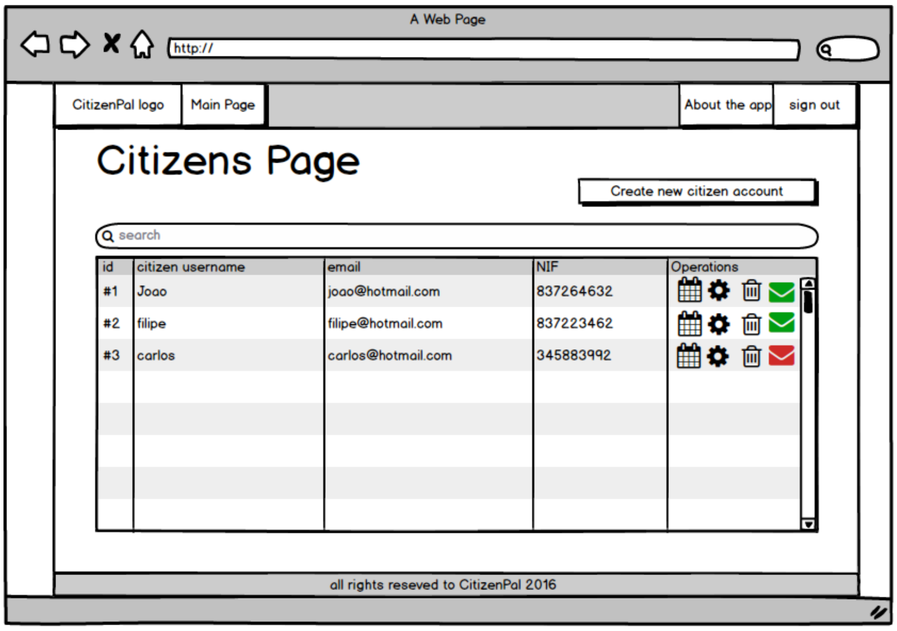


Figure 57: UI for CitizenPal back-office app, main page



Citizens Page

Create new citizen account

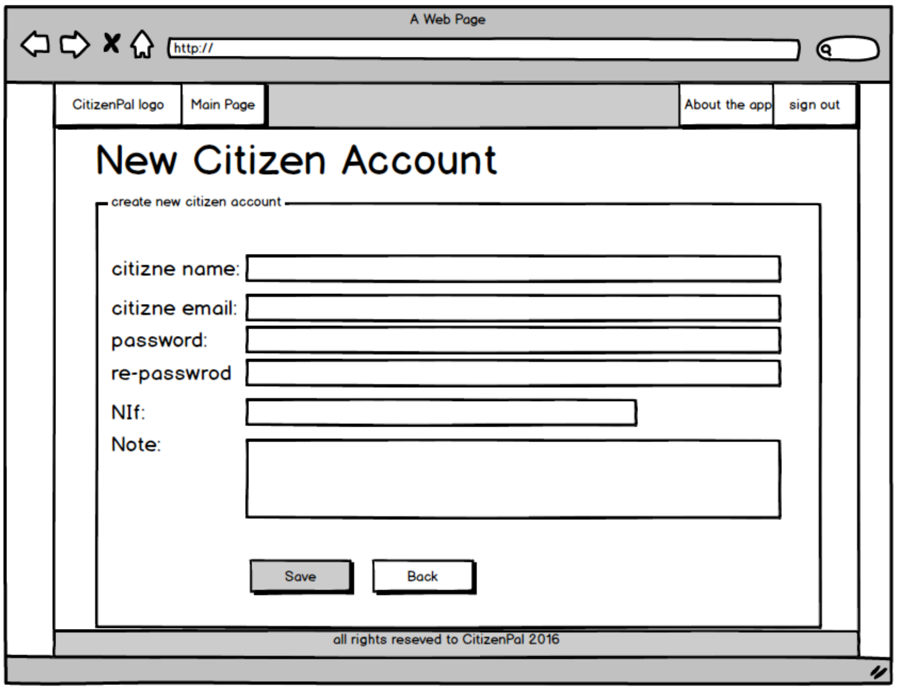
search

id	citizen username	email	NIF	Operations
#1	Joao	joao@hotmail.com	837264632	📅 ⚙ 🗑 🟩
#2	filipe	filipe@hotmail.com	837223462	📅 ⚙ 🗑 🟩
#3	carlos	carlos@hotmail.com	345883992	📅 ⚙ 🗑 🟥

all rights reseved to CitizenPal 2016

📅	Show all events of this citizen
⚙	Edit the data for this user
🗑	Delete this citizen record
🟩	all citizen events has been delevered.
🟥	some events for this citizen have been

Figure 58: UI for CitizenPal back-office app, citizens page



New Citizen Account

create new citizen account

citizne name:

citizne email:

password:

re-passwrod

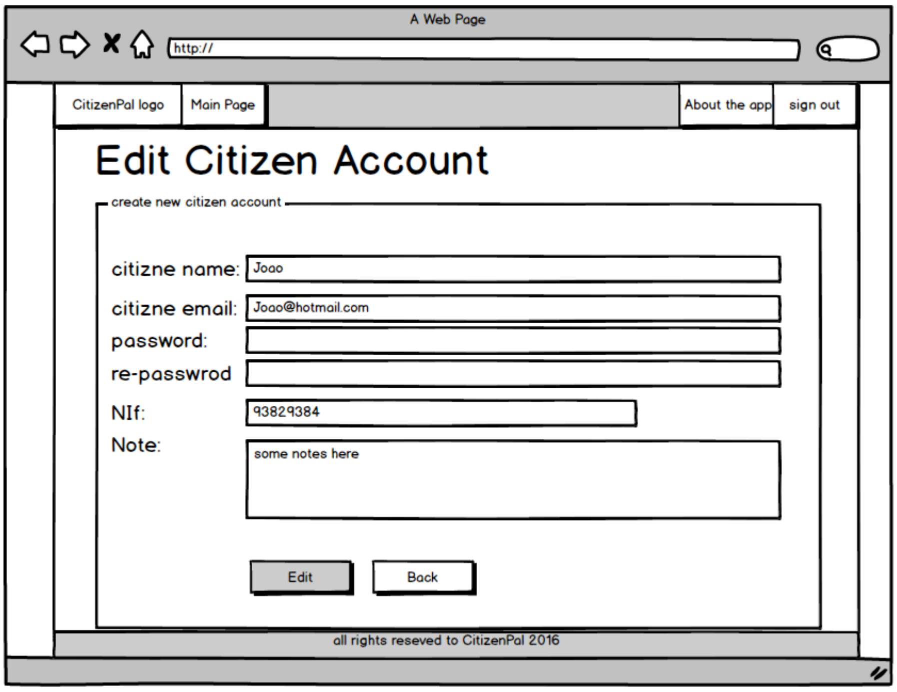
NIf:

Note:

Save Back

all rights reseved to CitizenPal 2016

Figure 59: UI for CitizenPal back-office app, new account page


A Web Page

http://

CitizenPal logo | Main Page | About the app | sign out

Edit Citizen Account

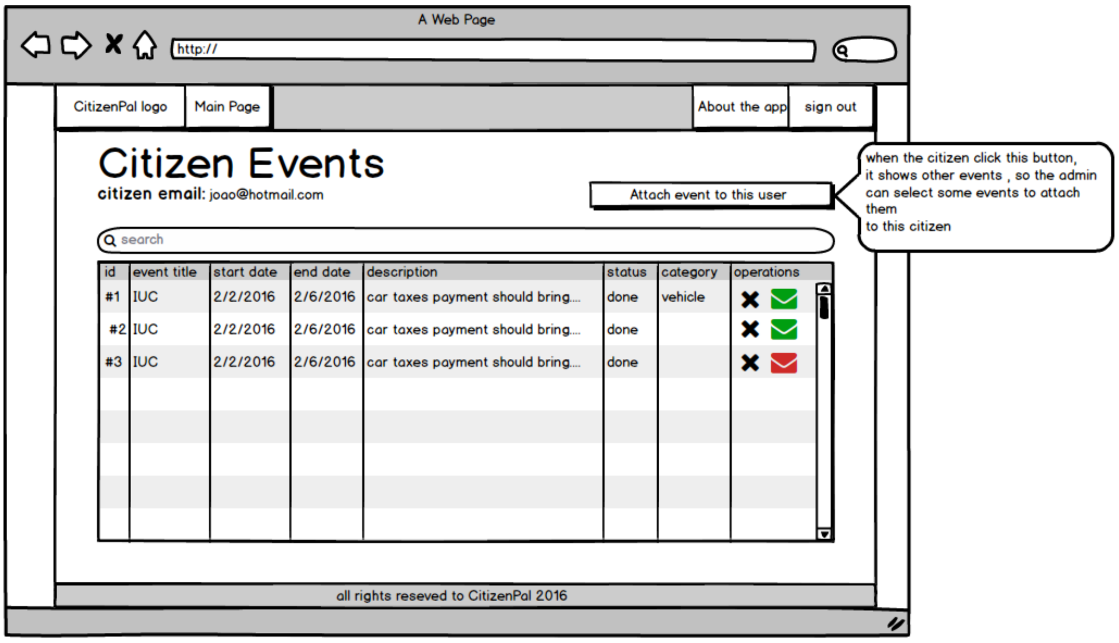
create new citizen account

citizne name:	Joao
citizne email:	Joao@hotmail.com
password:	
re-passwrod	
NIf:	93829384
Note:	some notes here

Edit | Back

all rights reseved to CitizenPal 2016

Figure 60: UI for CitizenPal back-office app, edit citizen info page


A Web Page

http://

CitizenPal logo | Main Page | About the app | sign out

Citizen Events

citizen email: joao@hotmail.com

Attach event to this user

when the citizen click this button, it shows other events , so the admin can select some events to attach them to this citizen

search

id	event title	start date	end date	description	status	category	operations
#1	IUC	2/2/2016	2/6/2016	car taxes payment should bring...	done	vehicle	✖ 🟩
#2	IUC	2/2/2016	2/6/2016	car taxes payment should bring...	done		✖ 🟩
#3	IUC	2/2/2016	2/6/2016	car taxes payment should bring...	done		✖ 🟥

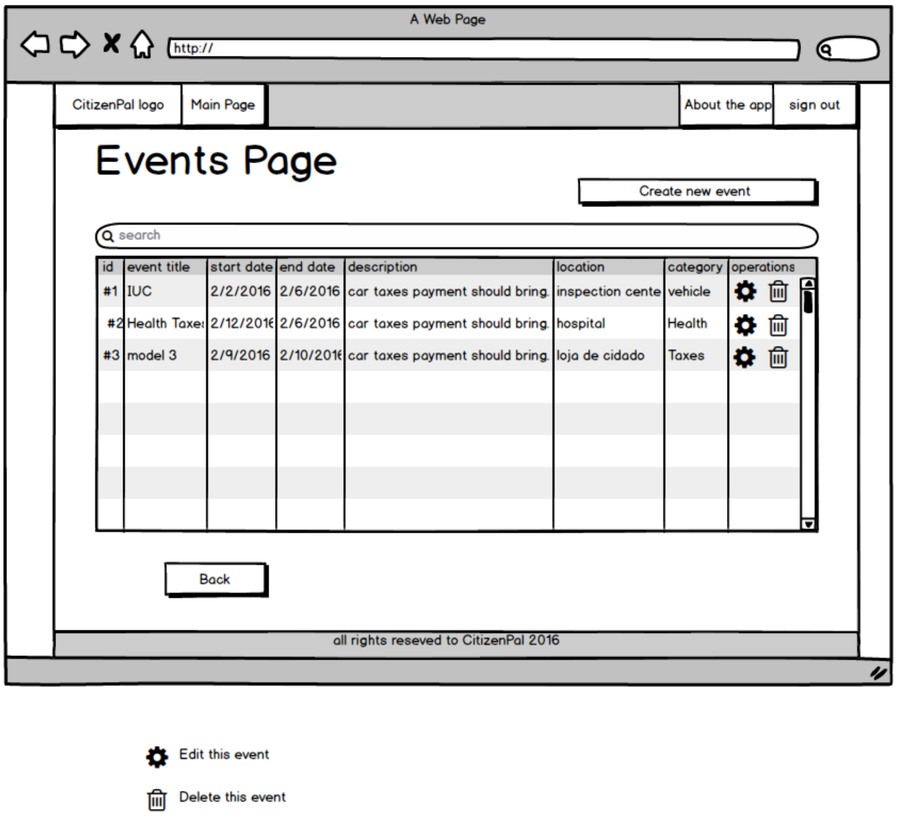
all rights reseved to CitizenPal 2016

✖ detach this event from the citizen

🟩 event has already been delivered

🟥 event has not been delivered

Figure 61: UI for CitizenPal back-office app, citizen events page

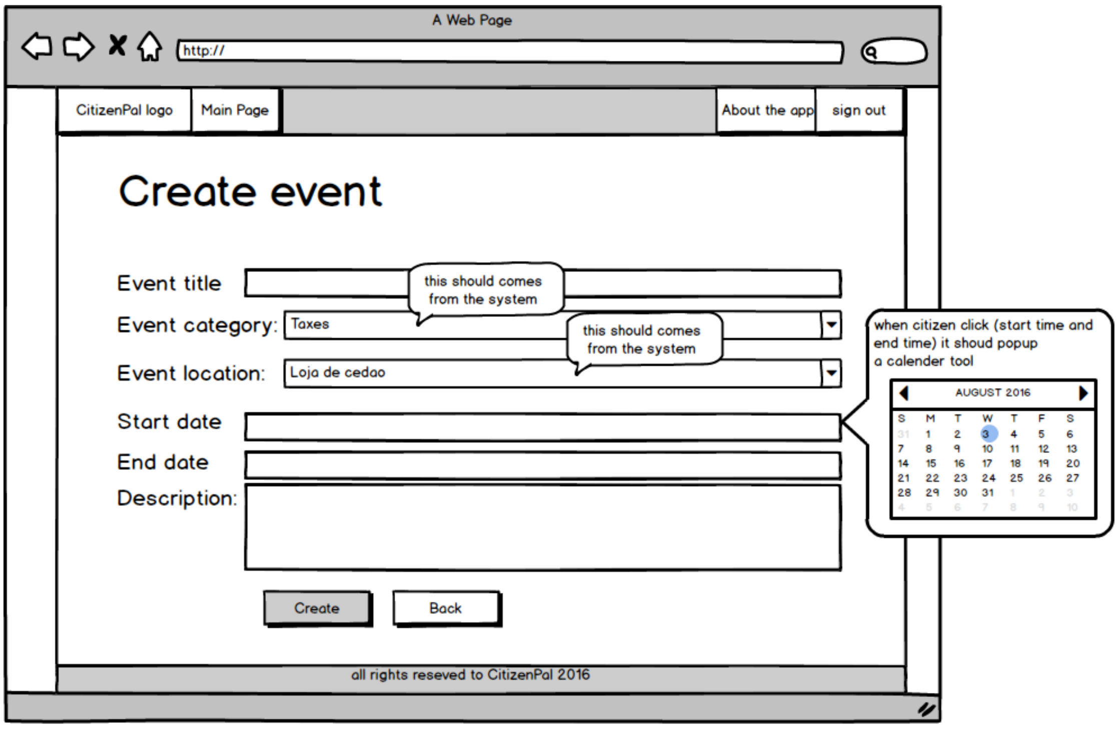


Events Page

id	event title	start date	end date	description	location	category	operations
#1	IUC	2/2/2016	2/6/2016	car taxes payment should bring	inspection cente	vehicle	⚙ 🗑
#2	Health Taxe	2/12/2016	2/6/2016	car taxes payment should bring	hospital	Health	⚙ 🗑
#3	model 3	2/9/2016	2/10/2016	car taxes payment should bring	loja de cidado	Taxes	⚙ 🗑

⚙	Edit this event
🗑	Delete this event

Figure 62: UI for CitizenPal back-office app, events page


A Web Page
http://
CitizenPal logo
Main Page
About the app
sign out
this should comes from the system
this should comes from the system
when citizen click (start time and end time) it shoud popup a calender tool
AUGUST 2016
all rights reseved to CitizenPal 2016

Create event

Event title:
Event category: Taxes
Event location: Loja de cedao
Start date
End date
Description:

Create Back

S	M	T	W	T	F	S
31	1	2	3	4	5	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	1	2	3
4	5	6	7	8	9	10

Figure 63: UI for CitizenPal back-office app, new event page

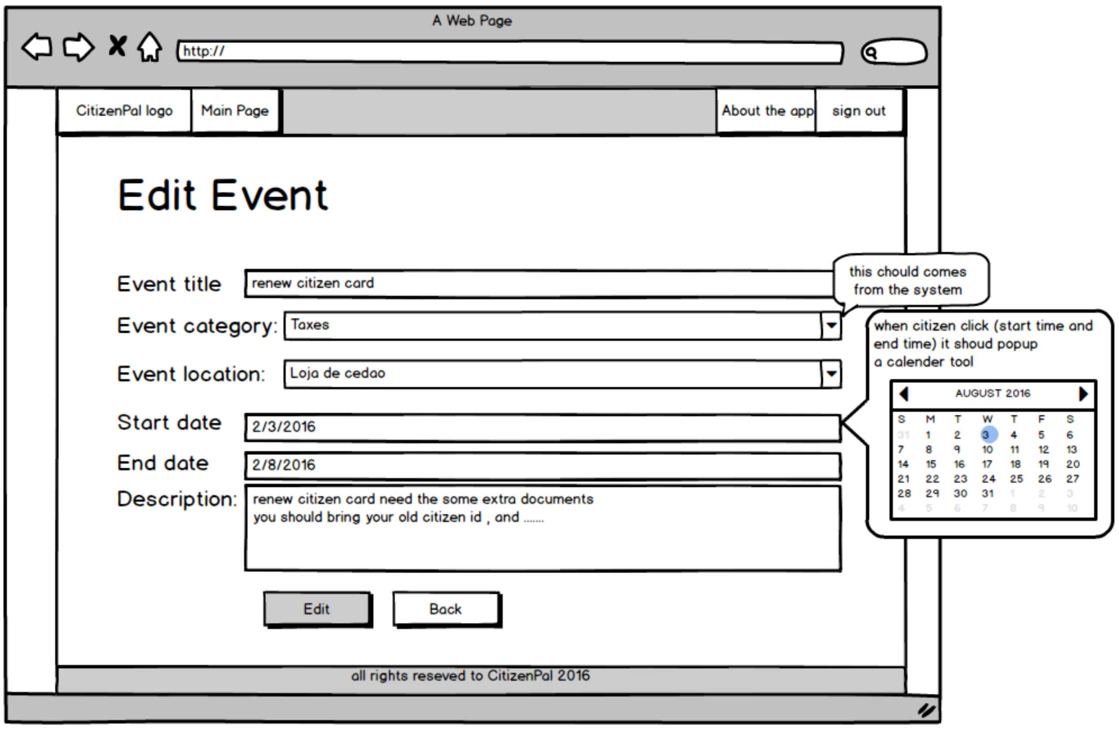
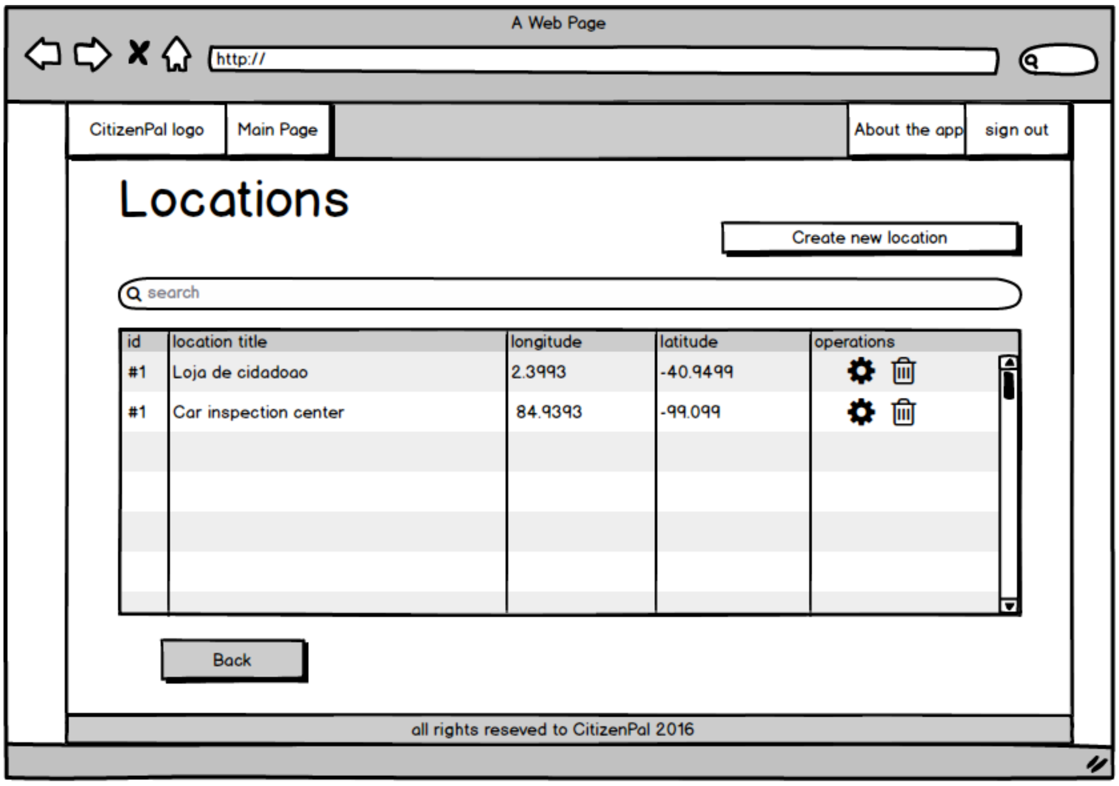

A Web Page
http://
CitizenPal logo
Main Page
About the app
sign out
Edit Event
Event title
renew citizen card
this chould comes from the system
Event category:
Taxes
Event location:
Loja de cedao
when citizen click (start time and end time) it shoud popup a calender tool
AUGUST 2016
Start date
2/3/2016
End date
2/8/2016
Description:
renew citizen card need the some extra documents you should bring your old citizen id , and ....
Edit
Back
all rights reseved to CitizenPal 2016

Figure 64: UI for CitizenPal back-office app,edit event page

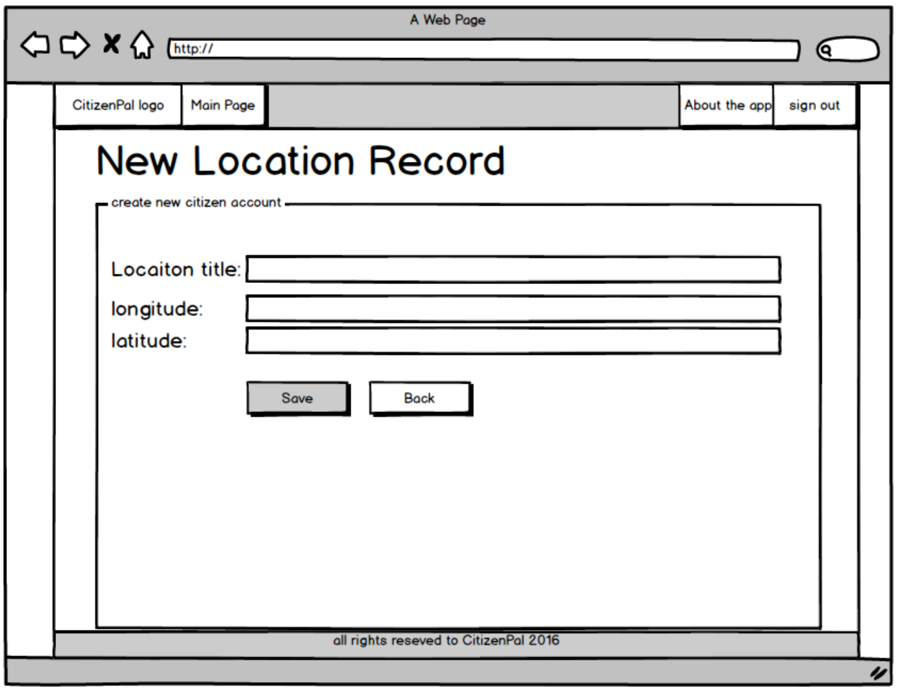


id	location title	longitude	latitude	operations
#1	Loja de cidadoao	2.3993	-40.9499	⚙ 🗑
#1	Car inspection center	84.9393	-99.099	⚙ 🗑

🗑 delete this location record

⚙ edit this location record

Figure 65: UI for CitizenPal back-office app, locations page



New Location Record

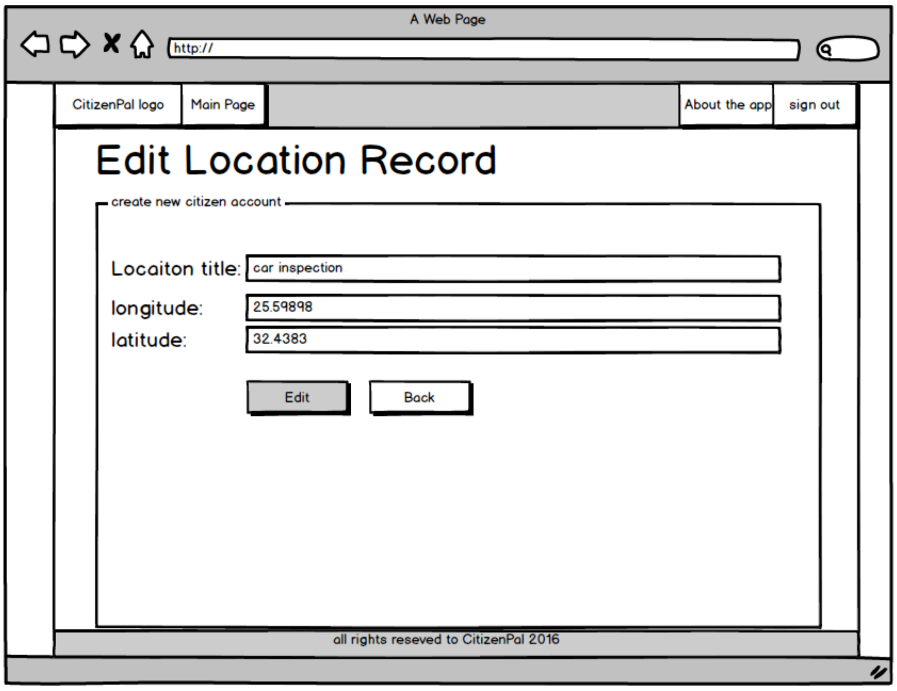
create new citizen account

Locaiton title:

longitude:

latitude:

Figure 66: UI for CitizenPal back-office app, new location page



Edit Location Record

create new citizen account

Locaiton title: car inspection

longitude: 25.59898

latitude: 32.4383

Figure 67: UI for CitizenPal back-office app, events page

10.3 Use cases

10.3.1 Smart-Phone App Use Case:

The following Use Cases explains the User goals of the mobile application. The User of the Mobile app in this case is the Citizens.

SP-UC-01

ID	SP-UC-01
Name	Register User
Description	User Can be able to register in citizenPal App using by using username, email, password and NIF(Número de Identificação Fiscal). email and NIF has to be Unique.
Actor	The Citizen
Priority Level	Essential level

SP-UC-02

ID	SP-UC-02
Name	Log in User
Description	User can be able to Log in Using his unique email and his password.
Actor	The Citizen
Level	Essential level

SP-UC-03

ID	SP-UC-03
Name	Show User Events
Description	User should be able to see his events after a success logging in CitizenPal App.
Actor	The Citizen
Level	Essential level

SP-UC-04

ID	SP-UC-04
Name	Filter Events
Description	User should be able to filter his all his events based on the the event's category.
Actor	The Citizen
Level	Essential level

SP-UC-05

ID	SP-UC-05
Name	Show Event's details
Description	User should be able to show every event's details. events details are event category, event's starting time, event's ending time, event's location name, event's description, and event's status.
Actor	The Citizen
Level	Essential level

SP-UC-06

ID	SP-UC-06
Name	Schedule event
Description	User Should be able to schedule his events to his own mobile calender.
Actor	The Citizen
Level	Essential level

SP-UC-07

ID	SP-UC-07
Name	Mark Event
Description	User should be able to Mark event. mark the event is referring that the event has been done or or it is still valid.
Actor	The Citizen
Level	Essential level

SP-UC-08

ID	SP-UC-08
Name	Save User email and password
Description	User can be able to save his username and password for the next logging in attempts without writing it again.
Actor	The Citizen
Level	Recommended level

SP-UC-09

ID	SP-UC-09
Name	keep logging in
Description	Use can be able to automatically log in when he opens the application. User should have and a successfully logging in.
Actor	The Citizen
Level	Recommended level

SP-UC-10

ID	SP-UC-10
Name	Show Location Map
Description	User should be able see the Location event on google Map.
Actor	The Citizen
Level	Recommended level

SP-UC-11

ID	SP-UC-11
Name	Notify for new event
Description	User shall receive an Notification for the new events that user might have. -User logged in SP-UC-09 . -Notification should be received even if the app is not open
Actor	The Citizen
Level	Essential level

10.3.2 Back-Office App

The Back office is an web client applications that can connect to the Server app and control the data. The major actor of the back office is the Admin.

BO-UC-01

ID	BO-UC-01
Name	Log In Admin
Description	User should be able to log in app as with user eamil and password. The User's email and the password is constants can't be change. and there is only one admin in the web application.
Actor	The Admin
Level	Essential level

BO-UC-02

ID	BO-UC-02
Name	Show all Users
Description	User should be able to browse all citizens that have been registered in the citizenPal App . citizens should be listed with Username, email, NIF. User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-03

ID	BO-UC-03
Name	Add New citizen
Description	User should be able to add new citizen to the CitizenPal App. - Citizen information the required are username, email, password,NIF number. - Notes field for citizen information is not required to complete registration. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-04

ID	BO-UC-04
Name	Edit citizen Information
Description	User should be able to Edit new citizen to the CitizenPal App. - Citizen information that can be edited are username, email, password,NIF number. - Notes field for citizen information is not required to complete registration. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-05

ID	BO-UC-05
Name	Delete citizen Information
Description	User should be able to delete all citizen Record. - Deleting Citizen Record should detach him from all his connection the the events. . - Deleting Citizen Record will not delete the its event information - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-06

ID	BO-UC-06
Name	Show Citizen Events
Description	User should be able to show all Events that belong to specific user. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-07

ID	BO-UC-07
Name	Detach citizen event
Description	User should be able to drop (detach) Event from specific User. Detaching from event should not affect the event information. - Detaching from event should not affect other users that have the same event. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-08

ID	BO-UC-08
Name	Attach Event
Description	User should be able to attach event to specific user. - Attaching event to user should not affect other users that have the same event. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-09

ID	BO-UC-09
Name	Filter Events records
Description	User should be able to search for specific events based on event title. - User should be logged in before.
Actor	The Admin
Level	Recommended level

BO-UC-10

ID	BO-UC-10
Name	Filter Citizens records
Description	User should be able to search for specific User based citizen's email. - User should be logged in before.
Actor	The Admin
Level	Recommended level

BO-UC-11

ID	BO-UC-11
Name	Show Events records
Description	User should be able Browse all events records in the citizenPal project. The Events should be listed by Event's title, event's startTime, event's End Time, event's description , event's location, event's category. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-12

ID	BO-UC-11
Name	Edit Event Information
Description	User should be able to Edit User Information. The Events info that can be edited are, event's title, event's start time, event's end time , event's description, event's category and event's location . - User should be logged in before.
Actor	The Admin
Level	Essential level

UC-13

ID	BO-UC-13
Name	Add New Event
Description	User should be able to Add new Event. The Events info that should be added to continue this use case , event's title, event's start time, event's end time , event's description, event's category and event's location . - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-14

ID	BO-UC-14
Name	Delete Event
Description	User Should be able to delete event record. - Deleting event detaches all users that already linked to it. - Deleting events will not delete the users that are attached to it. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-15

ID	BO-UC-15
Name	Browse event users
Description	User Should be able to browse all users that are attached to specific event. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-16

ID	BO-UC-16
Name	Deliver event
Description	User should be able to know if the a specific events is already delivered to the citizen - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-17

ID	BO-UC-17
Name	deliver user events
Description	User should be able to know if the a specific citizen has an event is not delivered , or all his events are delivered. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-18

ID	BO-UC-18
Name	Show Locations
Description	User should be able to see all locations records that are in CitizenPal's system. Locations information are , Location's Title , Location Latitude and Location Longitude . - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-19

ID	BO-UC-19
Name	Edit Location Title
Description	User should be able to Edit locations information. Locations information that can be edited are , Location's Title , Location Latitude and Location Longitude . - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-20

ID	BO-UC-20
Name	Add Location Title
Description	User should be able to Add locations record. Location information the required to be added are Location's title , Location's latitude and Location Longitude. - User should be logged in before.
Actor	The Admin
Level	Essential level

BO-UC-21

ID	BO-UC-21
Name	Portals Setting.
Description	User should be able to control the portal links and Urls. - User should be logged in before.
Actor	The Admin
Level	Recommended level

Server Use Cases

Server is an Application that does not have any User interface. It application that receives requests from a client. This client could be smart-phone application or back-office.

SA-UC-01

ID	SA-UC-01
Name	Register new citizen account.
Description	It is a request to register new citizen account that comes to sever. The request should carry out the necessary citizen information. - client does not need any User authentication.
Actor	Client (Mobile application)
Level	Essential

SA-UC-02

ID	SA-UC-02
Name	add new citizen account.
Description	It is a request to add new citizen account. - This requests needs User authentication.
Actor	Client (Admin)
Priority level	Conditional

SA-UC-03

ID	SA-UC-03
Name	Log in admin.
Description	It is a request log in the admin to the back office. logging in is including the authentication of the admin. This request will return true if Admin has successfully authenticate
Actor	Client (Admin)
Priority level	Essential

SA-UC-04

ID	SA-UC-04
Name	Delete Citizen Account
Description	It is a request to delete an specific citizen account. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Conditional

SA-UC-05

ID	SA-UC-05
Name	Edit Citizen account info
Description	It is a request to edit the citizen info. the request should carry the new citizen info account . - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Conditional

SA-UC-06

ID	SA-UC-06
Name	Edit citizen's flag.
Description	Change the citizen flag to 1 every time the user had new event. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-07

ID	SA-UC-07
Name	Retrieve citizens info.
Description	get all citizens account info. The info that should retrieved are username, citizen email, NIF, citizen flag . - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-08

ID	SA-UC-08
Name	Retrieve citizen events
Description	get all citizen events info. The event info that should be retrieved are: event title, event start time, event end time , event description, event status, event category title , category location title. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-09

ID	SA-UC-09
Name	Add an event to citizen
Description	It is and request that send to server to add specific event to citizen account. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-10

ID	SA-UC-10
Name	Detach an event from citizen account
Description	It is and request that sends to sever to detach specific event from citizen account. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-11

ID	SA-UC-11
Name	Retrieve all locations info.
Description	It is and request that sends to sever to retrieve all location info. The info that should retrieved are location title , location longitude, location latitude . - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Optional

SA-UC-12

ID	SA-UC-13
Name	Edit location locations info.
Description	It is and request that sends to server in order to edit a location record. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Optional

SA-UC-13

ID	SA-UC-13
Name	Delete location info
Description	It is and request that sends to server in order to edit a location record. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Optional

SA-UC-14

ID	SA-UC-14
Name	Retrieve all events info record
Description	It is and request that sends to server to get all events records. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-15

ID	SA-UC-15
Name	Edit event info
Description	It is a request that sends to server to Edit specific event. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-16

ID	SA-UC-16
Name	Retrieve citizens of specific event
Description	It is a request that sends to server to get all citizens that belong to specific event. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-17

ID	SA-UC-17
Name	Authenticate Admin requests.
Description	The requests that carried out from the admin back-office should be authenticate by admin email and password.
Actor	Client (Admin)
Priority level	Essential

SA-UC-17

ID	SA-UC-17
Name	Authenticate citizen requests.
Description	The requests that carried out from the admin back-office should be authenticate by admin email and password.
Actor	Client (Admin)
Priority level	Essential

SA-UC-19

ID	SA-UC-19
Name	Add new event record.
Description	It is a request that sends to server to add new event from to database. - Admin should be authenticated successfully
Actor	Client (Admin)
Priority level	Essential

SA-UC-20

ID	SA-UC-20
Name	log all request to file.
Description	every request shold be logged to external file with time date and request title.
Actor	Client (Admin)
Priority level	Optional

10.4 Screen shots

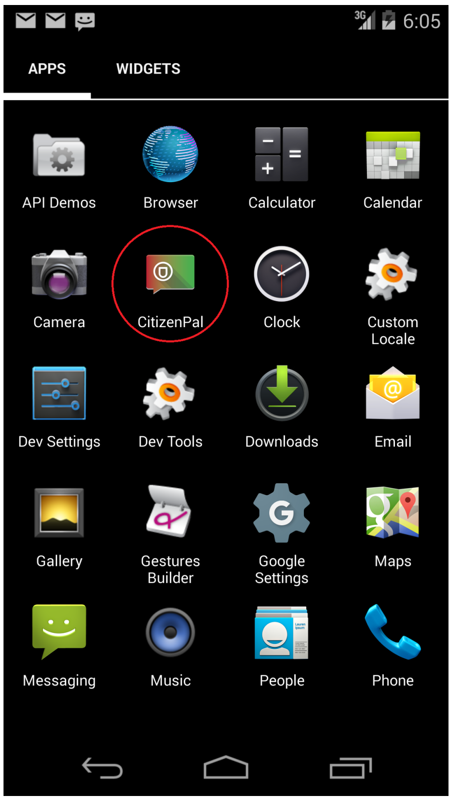


Figure 68: CitizenPal app among the Android applications

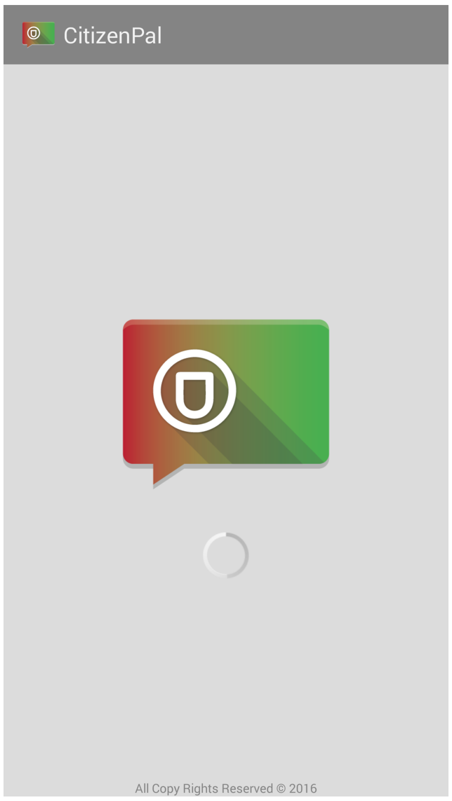


Figure 69: Screenshot of Android app, Start page.

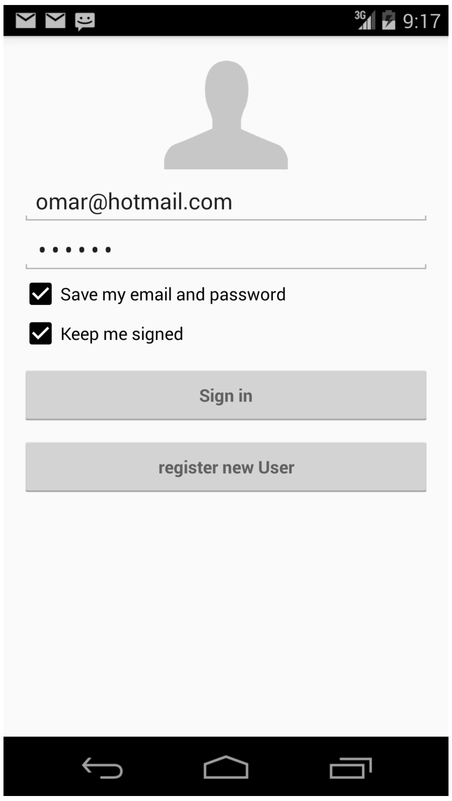
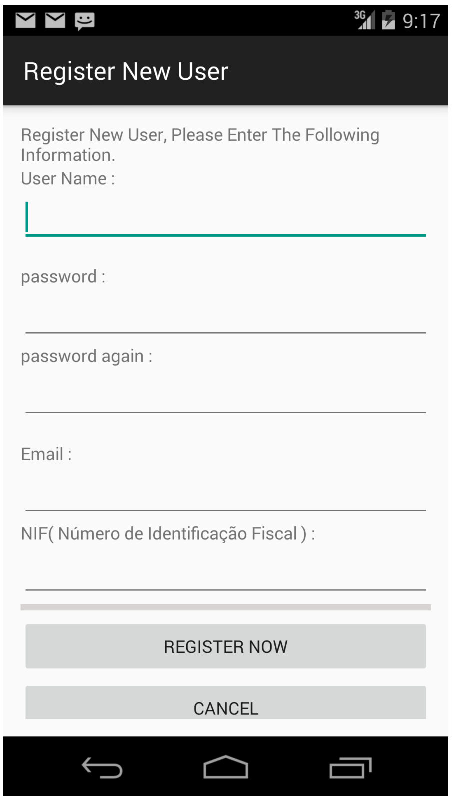


Figure 70: Screenshot of Android app, sign in page


9:17

Register New User

Register New User, Please Enter The Following Information.

User Name :

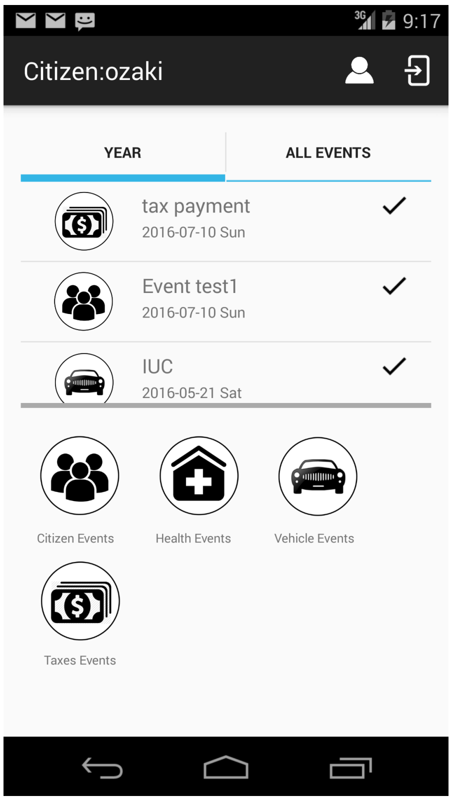
password :

password again :

Email :

NIF(Número de Identificação Fiscal) :

Figure 71: Screenshot of Android app, Registering Page.


9:17

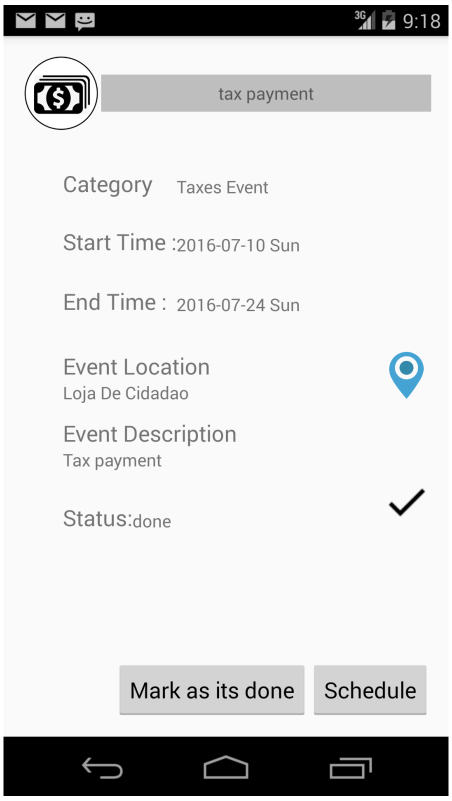
Citizen:ozaki

tax payment
2016-07-10 Sun ✓

Event test1
2016-07-10 Sun ✓

IUC
2016-05-21 Sat ✓

Figure 72: Screenshot of Android app, main page


9:18

tax payment

Category Taxes Event

Start Time :2016-07-10 Sun

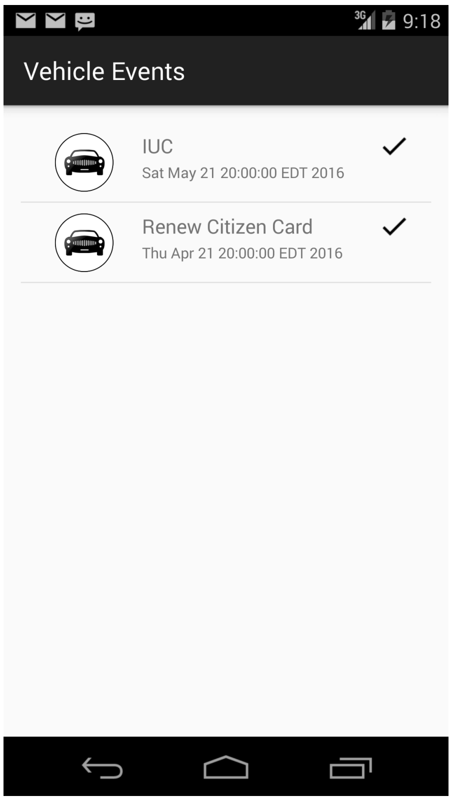
End Time : 2016-07-24 Sun

Event Location
Loja De Cidadao

Event Description
Tax payment

Status:done

Figure 73: Screenshot of Android app, event details.


9:18

Vehicle Events

IUC
Sat May 21 20:00:00 EDT 2016

Renew Citizen Card
Thu Apr 21 20:00:00 EDT 2016

Figure 74: Screenshot of Android app, filtered events.

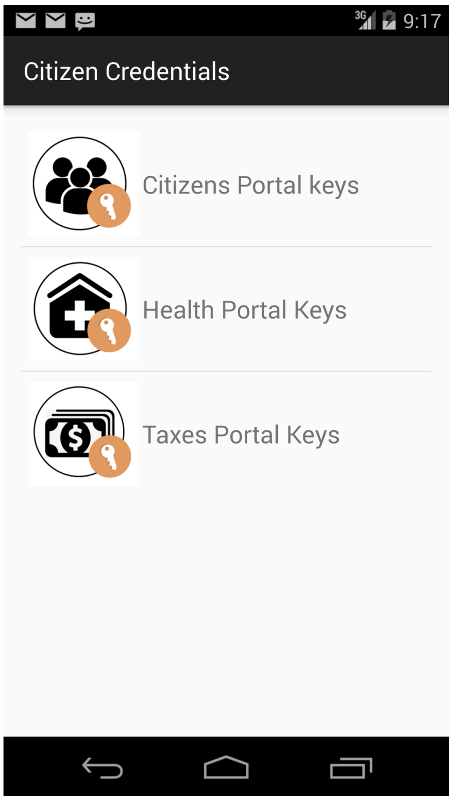
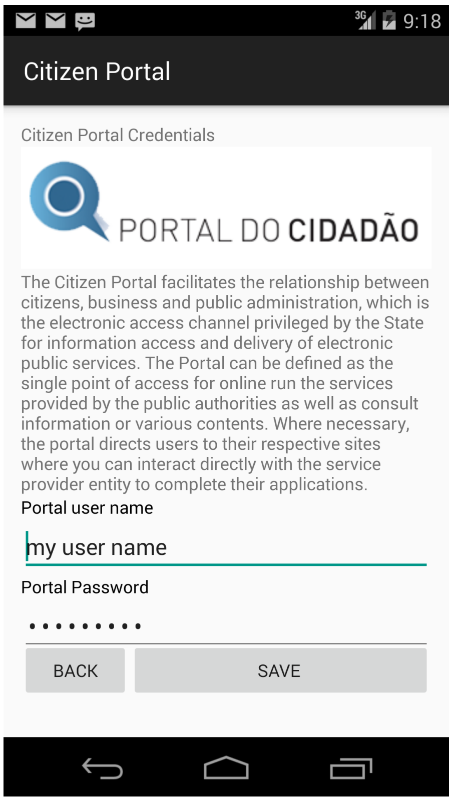


Figure 75: Screenshot of Android app, keys entry page



The Citizen Portal facilitates the relationship between citizens, business and public administration, which is the electronic access channel privileged by the State for information access and delivery of electronic public services. The Portal can be defined as the single point of access for online run the services provided by the public authorities as well as consult information or various contents. Where necessary, the portal directs users to their respective sites where you can interact directly with the service provider entity to complete their applications.

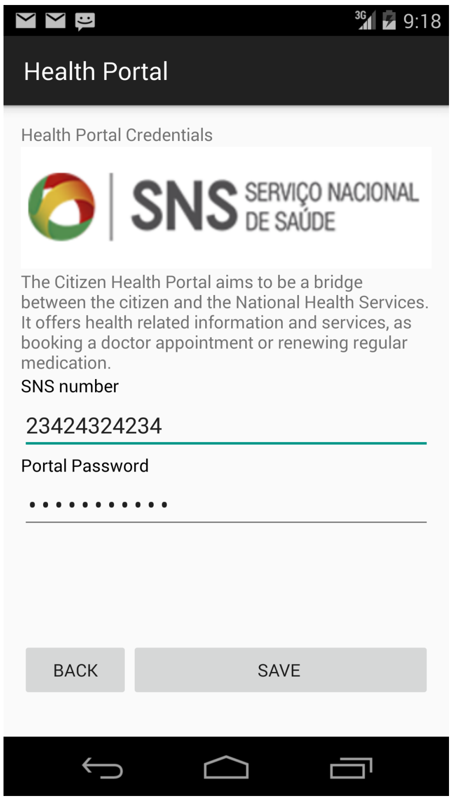
Portal user name

my user name

Portal Password

•••••••••

Figure 76: Screenshot of Android app, citizens portal credentials entry page



The Citizen Health Portal aims to be a bridge between the citizen and the National Health Services. It offers health related information and services, as booking a doctor appointment or renewing regular medication.

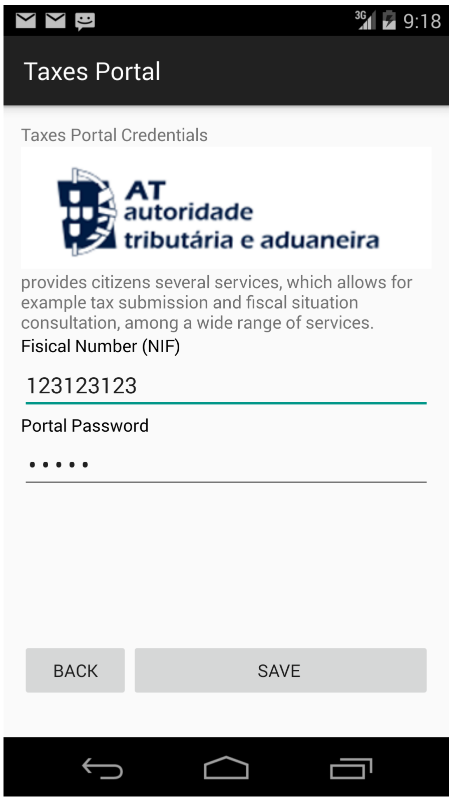
SNS number

23424324234

Portal Password

•••••••••••

Figure 77: Screenshot of Android app, health portal credentials entry page



provides citizens several services, which allows for example tax submission and fiscal situation consultation, among a wide range of services.

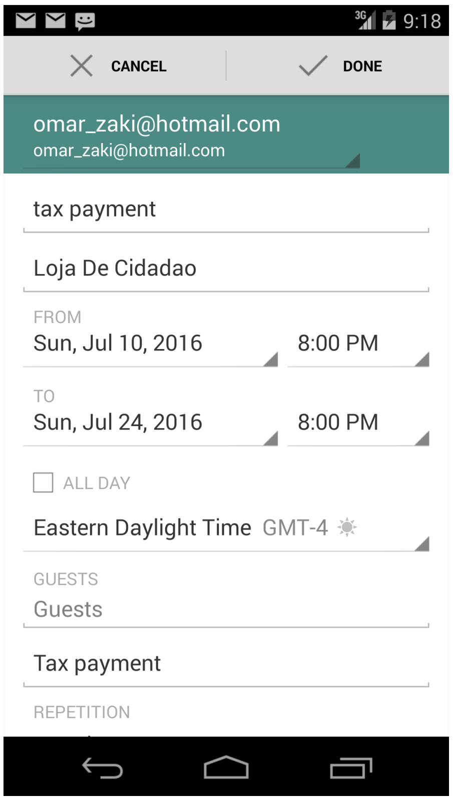
Fisical Number (NIF)

123123123

Portal Password

•••••

Figure 78: Screenshot of Android app, Taxes portal credentials entry page.


9:18

CANCEL DONE

omar_zaki@hotmail.com
omar_zaki@hotmail.com

tax payment

Loja De Cidadao

FROM
Sun, Jul 10, 2016 8:00 PM

TO
Sun, Jul 24, 2016 8:00 PM

ALL DAY

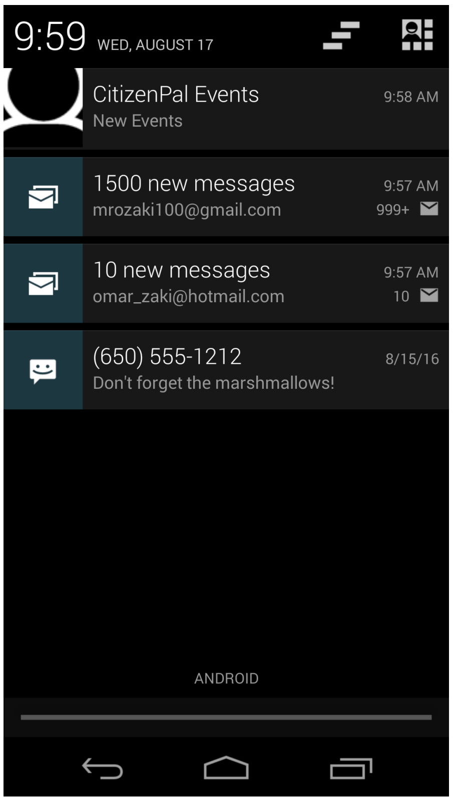
Eastern Daylight Time GMT-4

GUESTS
Guests

Tax payment

REPETITION

Figure 79: Screenshot of Android app , scheduling event to citizen's calender


9:59 WED, AUGUST 17

CitizenPal Events 9:58 AM
New Events

1500 new messages 9:57 AM
mrozaki100@gmail.com 999+

10 new messages 9:57 AM
omar_zaki@hotmail.com 10

(650) 555-1212 8/15/16
Don't forget the marshmallows!

ANDROID

Figure 80: Screenshot of Android app , notify a new event

Back-office interface

page.png

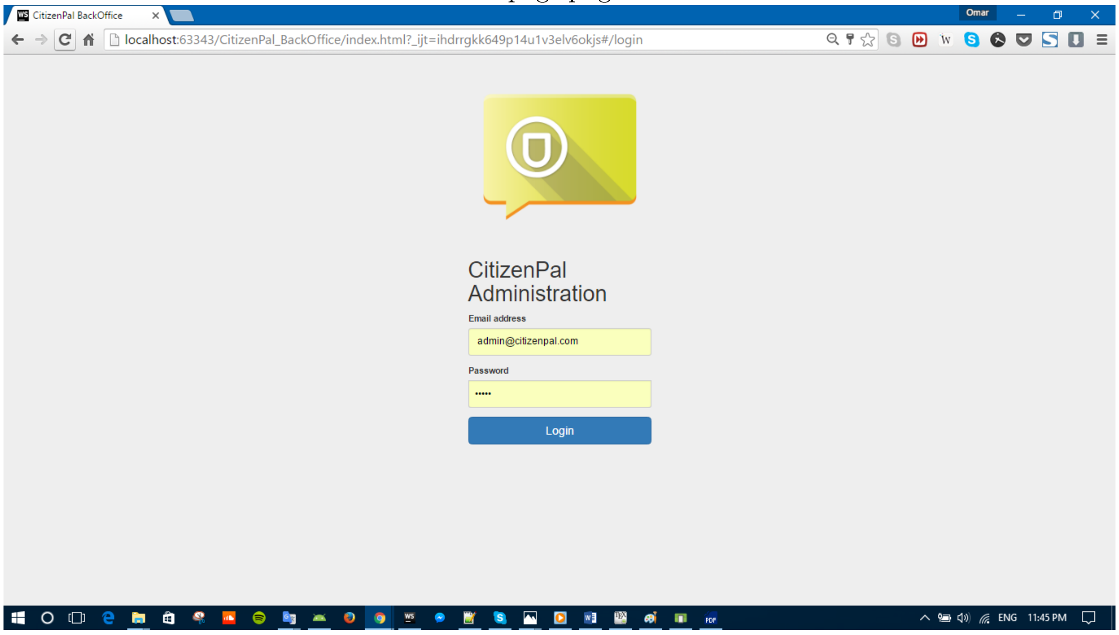

CitizenPal BackOffice
localhost:63343/CitizenPal_BackOffice/index.html?_ijt=ihdrrgkk649p14u1v3elv6okjs#/login
CitizenPal Administration
Email address
admin@citizenpal.com
Password
•••••
Login

Figure 81: Screenshot of back-office app, admin login page

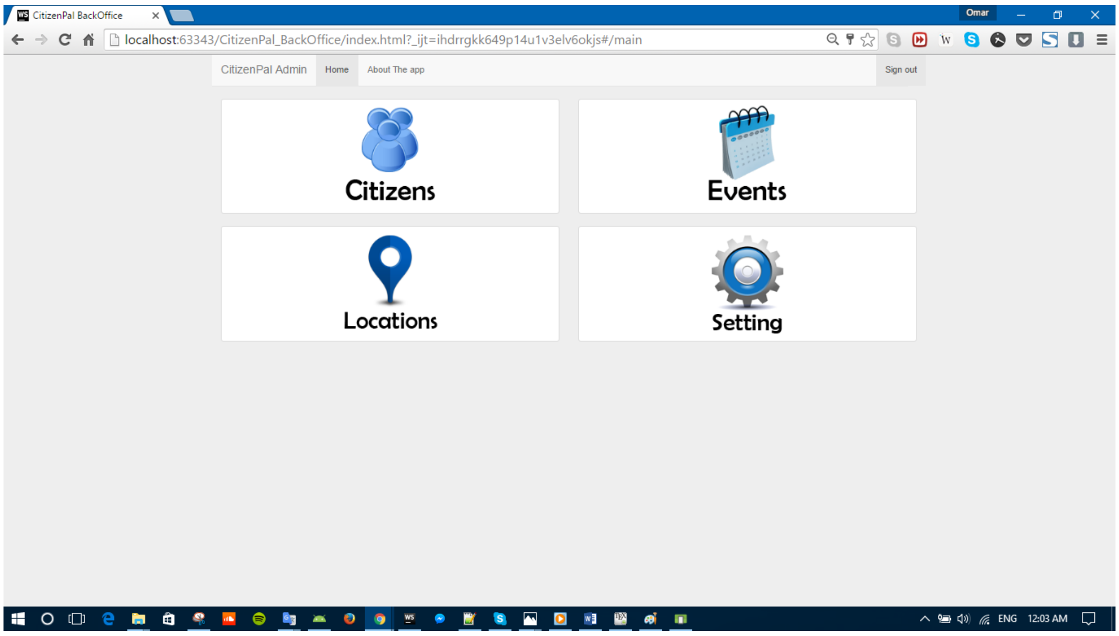
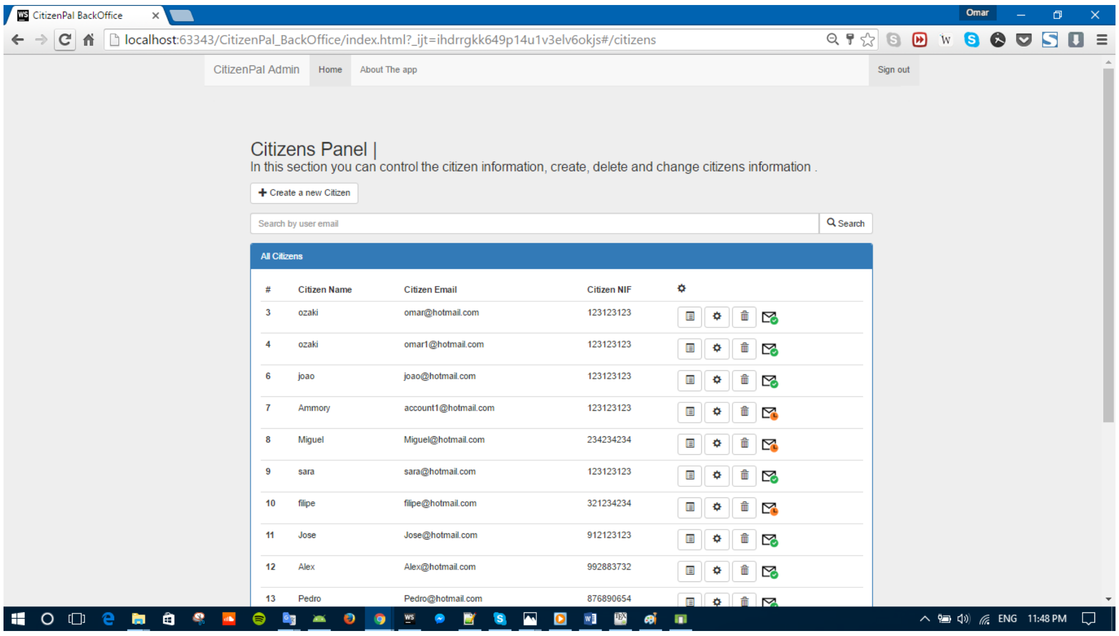


Figure 82: Screenshot of back-office app, citizen main page


CitizenPal Admin | Home | About The app | Sign out

Citizens Panel |

In this section you can control the citizen information, create, delete and change citizens information .

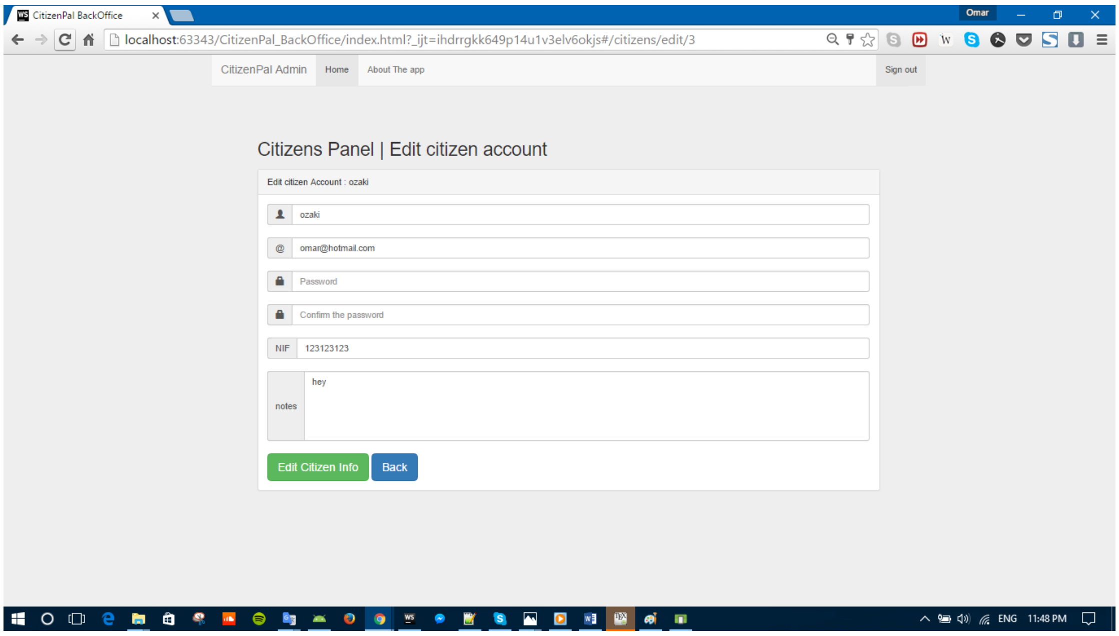
+ Create a new Citizen

Search by user email | Search

All Citizens

#	Citizen Name	Citizen Email	Citizen NIF	⚙
3	ozaki	omar@hotmail.com	123123123	
4	ozaki	omar1@hotmail.com	123123123	
6	joao	joao@hotmail.com	123123123	
7	Ammory	account1@hotmail.com	123123123	
8	Miguel	Miguel@hotmail.com	234234234	
9	sara	sara@hotmail.com	123123123	
10	filipe	filipe@hotmail.com	321234234	
11	Jose	Jose@hotmail.com	912123123	
12	Alex	Alex@hotmail.com	992883732	
13	Pedro	Pedro@hotmail.com	876890654	

Figure 83: Screenshot of back-office app, citizens page


CitizenPal Admin | Home | About The app | Sign out

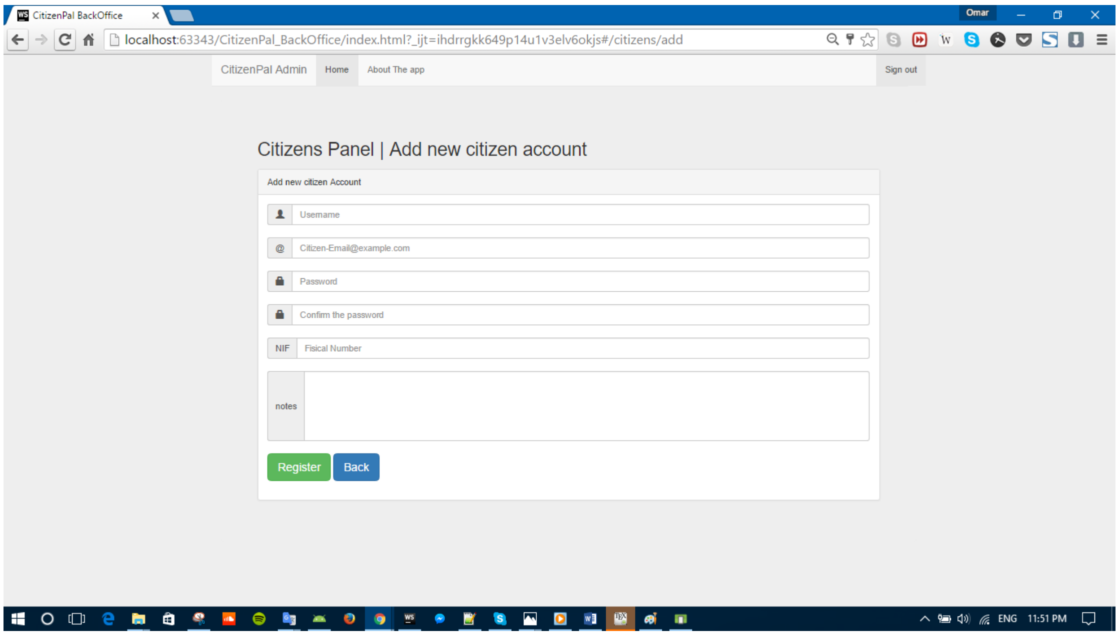
Citizens Panel | Edit citizen account

Edit citizen Account : ozaki

	ozaki
@	omar@hotmail.com
	Password
	Confirm the password
NIF	123123123
notes	hey

Edit Citizen Info | Back

Figure 84: Screenshot of back-office app, edit citizen info


CitizenPal BackOffice
localhost:63343/CitizenPal_BackOffice/index.html?_ijt=ihdrrgkk649p14u1v3elv6okjs#/citizens/add
CitizenPal Admin
Home
About The app
Sign out

Citizens Panel | Add new citizen account

Add new citizen Account

Username

Citizen-Email@example.com

Password

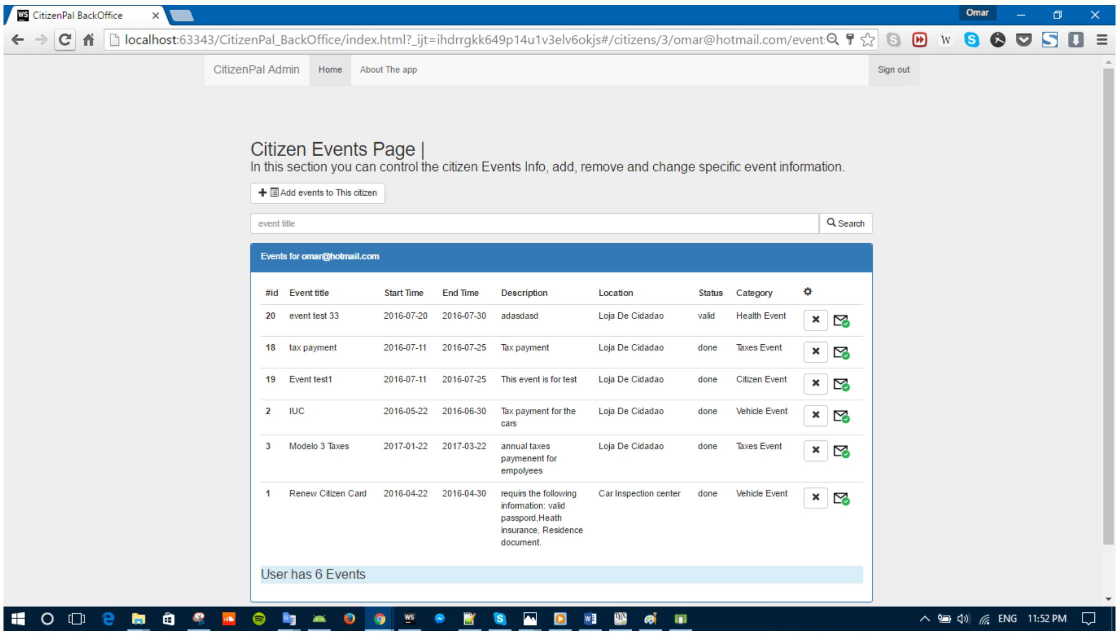
Confirm the password

NIF Fiscal Number

notes

Register Back

Figure 85: Screenshot of back-office app, create new citizen


CitizenPal BackOffice
localhost:63343/CitizenPal_BackOffice/index.html?_ijt=ihdrrgkk649p14u1v3elv6okjs#/citizens/3/omar@hotmail.com/event
CitizenPal Admin
Home
About The app
Sign out

Citizen Events Page |

In this section you can control the citizen Events Info, add, remove and change specific event information.

Add events to This citizen

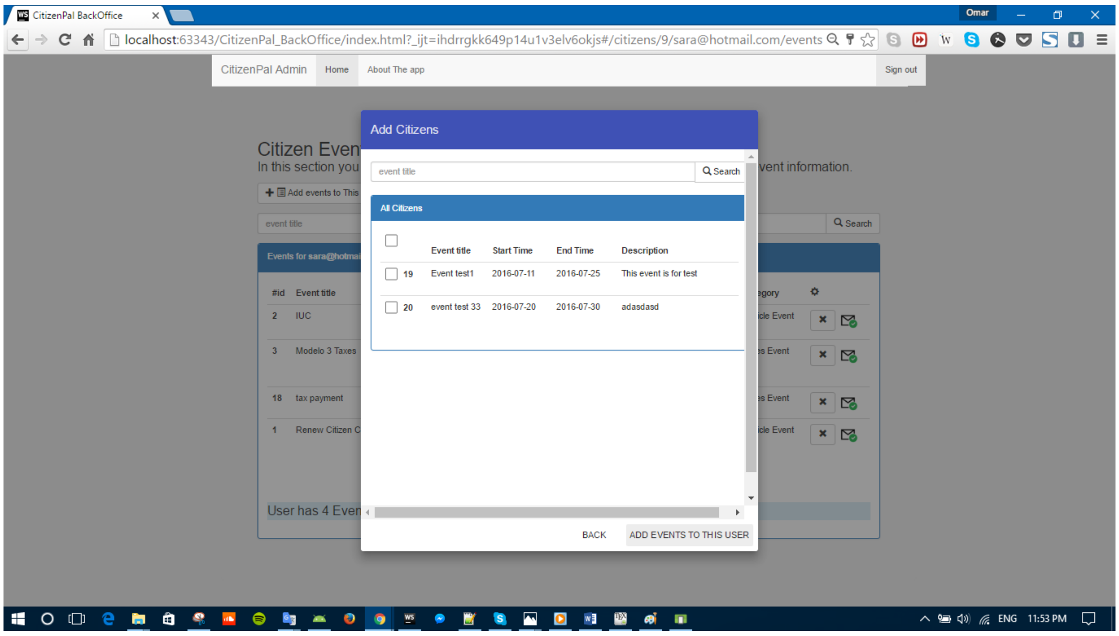
event title Search

Events for omar@hotmail.com

#id	Event title	Start Time	End Time	Description	Location	Status	Category	⚙
20	event test 33	2016-07-20	2016-07-30	adasdasd	Loja De Cidadao	valid	Health Event	✖ ✉
18	tax payment	2016-07-11	2016-07-25	Tax payment	Loja De Cidadao	done	Taxes Event	✖ ✉
19	Event test1	2016-07-11	2016-07-25	This event is for test	Loja De Cidadao	done	Citizen Event	✖ ✉
2	IUC	2016-05-22	2016-06-30	Tax payment for the cars	Loja De Cidadao	done	Vehicle Event	✖ ✉
3	Modelo 3 Taxes	2017-01-22	2017-03-22	annual taxes paymenent for empolyees	Loja De Cidadao	done	Taxes Event	✖ ✉
1	Renew Citizen Card	2016-04-22	2016-04-30	requirs the following information: valid passport,Heath insurance, Residence document.	Car Inspection center	done	Vehicle Event	✖ ✉

User has 6 Events

Figure 86: Screenshot of back-office app, events that belong to specific citizen



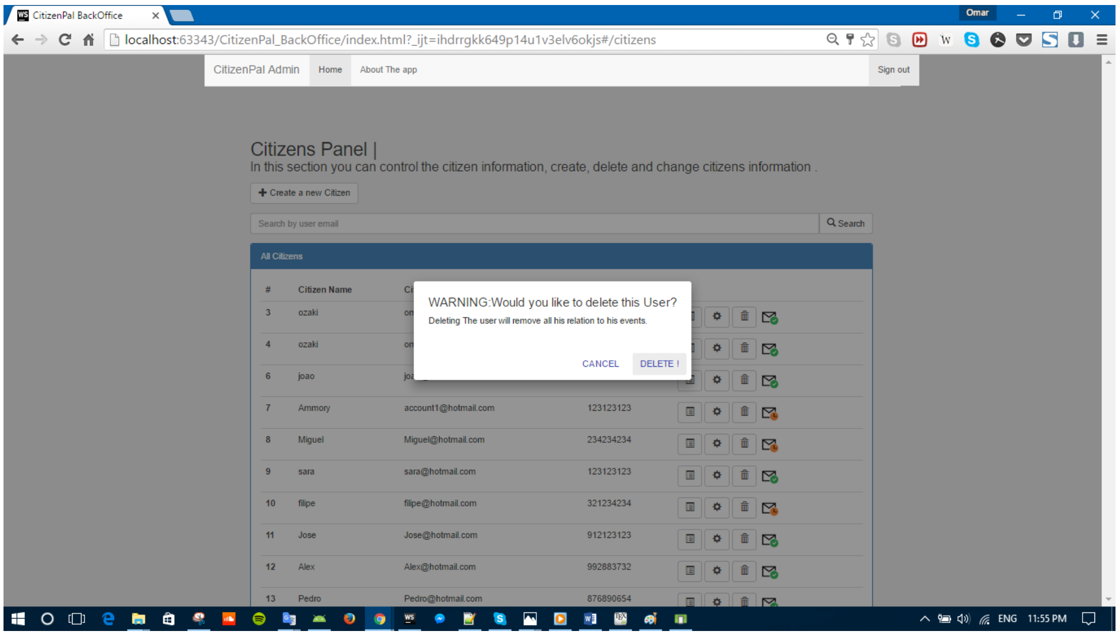
Citizen Even
In this section you … vent information.

	Event title	Start Time	End Time	Description
19	Event test1	2016-07-11	2016-07-25	This event is for test
20	event test 33	2016-07-20	2016-07-30	adasdasd

#id	Event title	…	egory
2	IUC		cle Event
3	Modelo 3 Taxes		s Event
18	tax payment		s Event
1	Renew Citizen C		cle Event

User has 4 Even

Figure 87: Screenshot of back-office app, attach event to citizen

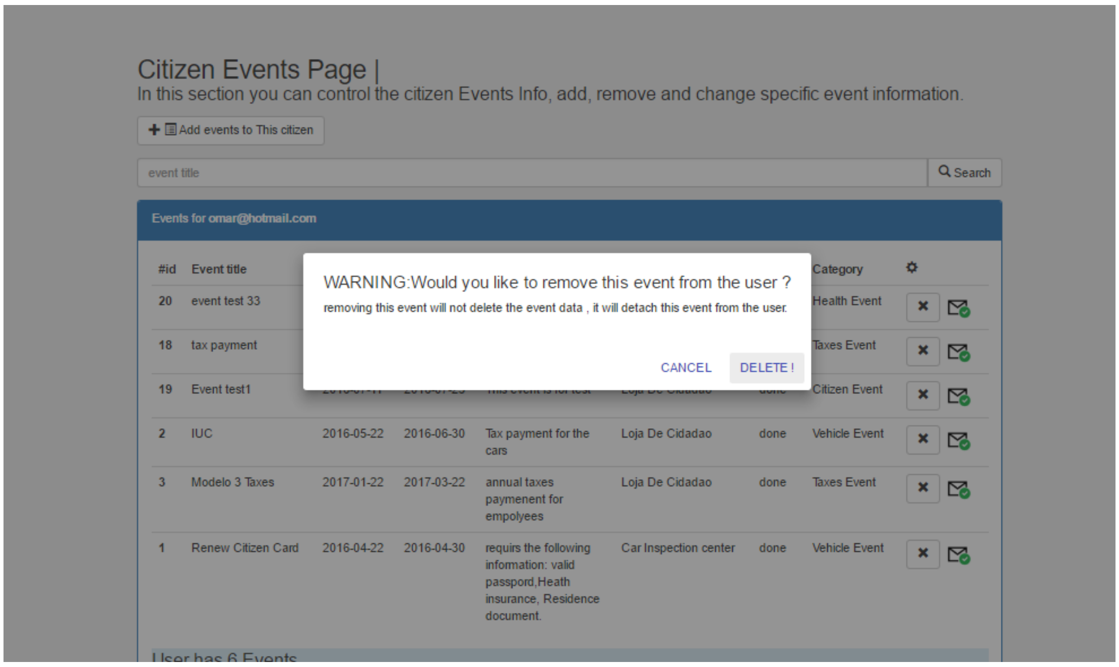


Citizens Panel |
In this section you can control the citizen information, create, delete and change citizens information .

WARNING:Would you like to delete this User?
Deleting The user will remove all his relation to his events.

#	Citizen Name	Ci	
3	ozaki	on	
4	ozaki	on	
6	joao	jo	
7	Ammory	account1@hotmail.com	123123123
8	Miguel	Miguel@hotmail.com	234234234
9	sara	sara@hotmail.com	123123123
10	filipe	filipe@hotmail.com	321234234
11	Jose	Jose@hotmail.com	912123123
12	Alex	Alex@hotmail.com	992883732
13	Pedro	Pedro@hotmail.com	876890654

Figure 88: Screenshot of back-office app, delete a citizen



Citizen Events Page |

In this section you can control the citizen Events Info, add, remove and change specific event information.

Events for omar@hotmail.com

WARNING:Would you like to remove this event from the user ?

removing this event will not delete the event data , it will detach this event from the user.

#id	Event title						Category
20	event test 33						Health Event
18	tax payment						Taxes Event
19	Event test1						Citizen Event
2	IUC	2016-05-22	2016-06-30	Tax payment for the cars	Loja De Cidadao	done	Vehicle Event
3	Modelo 3 Taxes	2017-01-22	2017-03-22	annual taxes paymenent for empolyees	Loja De Cidadao	done	Taxes Event
1	Renew Citizen Card	2016-04-22	2016-04-30	requirs the following information: valid passport,Heath insurance, Residence document.	Car Inspection center	done	Vehicle Event

User has 6 Events

Figure 89: Screenshotof back-office app, detach event from citizen

10.5 Test Cases

10.5.1 Smart-Phone app Test Cases

- **Test Case ID:** SP-TC-01
- **Use Case ID:** SP-UC-01
- **Description:** Registering User
- **Pre-condition:** should not have a citizen already registered with the same email.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Username: joao Password: qwe123 Re-Password: qwe123 Email: joao1092@hotmail.com NIF: 738293213	success	success	Accepted
02	Username: joao33 Password: qwe123 Re-Password: qww123 Email: joao1092@hotmail.com NIF: 738293213	Fail password does not match	Fail	Accepted
03	Username: joao33 Password: qwe123 Re-Password: qww123 Email: joao1092hotmail.com NIF: 738293213	Fail password does not match	Fail	Accepted
04	Username: joao33 Password: qwe123 Re-Password: qww123 Email: joao1092hotmail.com NIF: 7383dsfsdsw	Fail Non-valid NIF format.	Fail	Accepted

---

- **Test Case ID:** SP-TC-02
- **Use Case ID:** SP-UC-02
- **Description:**Log in User
- **Pre-condition:** Citizen should be successfully registered.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Email: omar@hotmail.com Password:123123	Success	Success	Accepted
02	Email: omarhotmail.com Password:123123	Fail Non-valid email	Fail	Accepted
03	Email: omarddfddf@hotmail.com Password:123123	Fail Email/Password is not correct	Fail	Accepted

---

- **Test Case ID:** SP-TC-03
- **Use Case ID:** SP-UC-03
- **Description:** Show User events
- **Pre-condition:** User should be logged in successfully, User should have some events.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** SP-TC-04
- **Use Case ID:** SP-UC-03
- **Description:** Show User events
- **Pre-condition:** User should be logged in successfully, User does not have any events.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success no data is showed	Success	Accepted

- **Test Case ID:** SP-TC-05
- **Use Case ID:** SP-UC-04
- **Description:** Filter events based on category
- **Pre-condition:** User should be logged in successfully, User does not have any events.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** SP-TC-06
- **Use Case ID:** SP-UC-05
- **Description:** Show Event's detail
- **Pre-condition:** User should be logged in successfully, User does have an event.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** SP-TC-07
- **Use Case ID:** SP-UC-06
- **Description:** Schedule an Event.

- **Pre-condition:** User should be logged in successfully, User should sign in to his calendar account into android system.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** SP-TC-08
- **Use Case ID:** SP-UC-07
- **Description:** Mark an Event as done.
- **Pre-condition:** User should be logged in successfully, User has a least one event, The event's status is valid.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** SP-TC-09
- **Use Case ID:** SP-UC-08
- **Description:** Save email and password.
- **Pre-condition:** User should be logged in successfully.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** SP-TC-10
- **Use Case ID:** SP-UC-09
- **Description:** Keep logging User in.
- **Pre-condition:** User should be logged in successfully.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** SP-TC-11

- **Use Case ID:** SP-UC-10
- **Description:** Show Location Map. .
- **Pre-condition:** User should be logged in successfully.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** SP-TC-12
- **Use Case ID:** SP-UC-11
- **Description:** Notify User for new events.User should have at least one new event.
- **Pre-condition:** User should be logged in successfully.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

10.5.2 Back-Office app Test Cases

- **Test Case ID:** BO-TC-01
- **Use Case ID:** BO-UC-01
- **Description:** Log in Admin
- **Pre-condition:** no pre-conditions.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Email: admin@citizenpal.com Password: admin	Success	Success	Accepted
02	Email: otheremail@citizenpal.com Password: admin	Fail wrong email or password	Fail	Accepted
03	Email: admin@citizenpal.com Password: otherpass	Fail wrong email or password	Fail	Accepted
04	Email: citizenpal.com Password: otherpass	Fail wrong email or password	Fail	Accepted
05	Email: Password: otherpass	Fail Email is required	Fail	Accepted
06	Email: Password: otherpass	Fail password is required	Fail	Accepted

---

- **Test Case ID:** BO-TC-01
- **Use Case ID:** BO-UC-01
- **Description:** Log in Admin
- **Pre-condition:** no pre-conditions.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Email: admin@citizenpal.com Password: admin	Success	Success	Accepted
02	Email: otheremail@citizenpal.com Password: admin	Fail wrong email or password	Fail	Accepted
03	Email: admin@citizenpal.com Password: otherpass	Fail wrong email or password	Fail	Accepted
04	Email: citizenpal.com Password: otherpass	Fail wrong email or password	Fail	Accepted
05	Email: Password: otherpass	Fail Email is required	Fail	Accepted
06	Email: Password: otherpass	Fail password is required	Fail	Accepted

- **Test Case ID:** BO-TC-02
- **Use Case ID:** BO-UC-02
- **Description:** Show All Users
- **Pre-condition:** Admin Login successfully .
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	No data is required	Success	Success	Accepted

- **Test Case ID:** BO-TC-03
- **Use Case ID:** BO-UC-03
- **Description:** Add new citizen account
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Not Accepted , 1 error needs to be fixed

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Username: Alex Email: Alex@hotmail.com Password: asd123 Re-Password: asd123 NIF: 992883732 Note: this is my test for test	Success	Success	Accepted
02	Username: Email: Alex@hotmail.com Password: asd123 Re-Password: asd123 NIF: 992883732 Note: this is my test for test	Fail (user name required)	Fail	Accepted
03	Username:Alex Email: Alex@hotmail.com Password: asd123 Re-Password: asd123 NIF: 992883732 Note: this is my test for test	Fail User email already registered	Fail	Accepted
04	Username:Alex Email: dfdhotmail.com Password: asd123 Re-Password: asd123 NIF: 992883732 Note: this is my test for test	Fail Email not valid	Fail	Accepted
04	Username:Alex Email: dfdhotmail.com Password: asd Re-Password: asd123 NIF: 992883732 Note: this is my test for test	Fail Password does not match	Fail	Accepted
04	Username:Alex Email: dfdhotmail.com Password: asd Re-Password: asd123 NIF: dfds Note: this is my test for test	Fail NIF should be a number	Fail	Not Accepted

- **Test Case ID:** BO-TC-04
- **Use Case ID:** BO-UC-04
- **Description:** Add new citizen account
- **Pre-condition:** Admin Logged in successfully .

- **Acceptance:** Not Accepted , 1 error needs to be fixed (similar to BO-TC-3)

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** BO-TC-05
- **Use Case ID:** BO-UC-05
- **Description:** Delete Citizen Info
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted.

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** BO-TC-06
- **Use Case ID:** BO-UC-06
- **Description:** Show the events that belong to specific citizen
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted.

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** BO-TC-07
- **Use Case ID:** BO-UC-07
- **Description:** Detach an event from specific citizen, citizen account should have some events attached to it.
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted.

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** BO-TC-08
- **Use Case ID:** BO-UC-08

- **Description:** Attach an event to specific citizen
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted.

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** BO-TC-09
- **Use Case ID:** BO-UC-09
- **Description:** Filter events records.
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted.

#no	Test Data	Expected Result	Actual Result	Acceptance
01	no data needed	Success	Success	Accepted

- **Test Case ID:** BO-TC-10
- **Use Case ID:** BO-UC-10
- **Description:** Filter citizens records.
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted.

#no	Test Data	Expected Result	Actual Result	Acceptance
01	omar@hotmail.com (any citizen email)	Success	Success	Accepted

---

- **Test Case ID:** BO-TC-11
- **Use Case ID:** BO-UC-11
- **Description:** Show all events records
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted.

#no	Test Data	Expected Result	Actual Result	Acceptance
01	No data is required	Success	Success	Accepted

- **Test Case ID:** BO-TC-13
- **Use Case ID:** BO-UC-13
- **Description:** Add new event record
- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted. 1 error

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Title: Renew Car License Category: Vehicle Event Location: Car Inspection center StartTime: 2016-08-17 EndTime: 2016-08-24 Description: should bring several documents	Success	Success	Accepted
02	Title: Category: Vehicle Event Location: Car Inspection center. StartTime: 2016-08-17 EndTime: 2016-08-24 Description: should bring several documents	Fail (title required)	Fail	Accepted
03	Title:Renew Car License Category: Vehicle Event Location: Car Inspection center. StartTime: 2016-08-24 EndTime: 2016-08-17 Description: should bring several documents	Fail (start time should be less than end time)	Fail	Not Accepted
04	Title: Renew Car License Category: Vehicle Event Location: Car Inspection center StartTime: EndTime: 2016-08-24 Description: should bring several documents	Fail (Starttime and Endtime are required)	Fail	Accepted
05	Title: Renew Car License Category: Vehicle Event Location: Car Inspection center StartTime: 2016-08-17 EndTime: 2016-08-24 Description:	Fail (Description required)	Fail	Accepted

---

- **Test Case ID:** BO-TC-12
- **Use Case ID:** BO-UC-12
- **Description:** Edit specific event record

- **Pre-condition:** Admin Logged in successfully .
- **Acceptance:** Accepted, 1 error (similar to BO-TC-13)

- **Test Case ID:** BO-TC-14
- **Use Case ID:** BO-UC-14
- **Description:** Delete event record.
- **Pre-condition:** Admin Logged in successfully.

#no	Test Data	Expected Result	Actual Result	Acceptance
01	No data is required	Success	Success	Accepted

- **Test Case ID:** BO-TC-15
- **Use Case ID:** BO-UC-15
- **Description:** Browse event users.
- **Pre-condition:** Admin Logged in successfully.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	No data is required	Success	Success	Accepted

- **Test Case ID:** BO-TC-16
- **Use Case ID:** BO-UC-16
- **Description:** Deliver sign event.
- **Pre-condition:** Admin Logged in successfully.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Select one event	Success	Success	Accepted

- **Test Case ID:** BO-TC-17
- **Use Case ID:** BO-UC-17
- **Description:** Deliver sign events on user record.
- **Pre-condition:** Admin Logged in successfully.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Select a citizen record	Success	Success	Accepted

- **Test Case ID:** BO-TC-18
- **Use Case ID:** BO-UC-18
- **Description:** show location records.
- **Pre-condition:** Admin Logged in successfully.
- **Acceptance:** Accepted

#no	Test Data	Expected Result	Actual Result	Acceptance
01	No data is required	Success	Success	Accepted

- **Test Case ID:** BO-TC-19
- **Use Case ID:** BO-UC-19
- **Description:** Add new Location record
- **Pre-condition:** Admin Logged in successfully.
- **Acceptance:** Not Accepted,1 errors

#no	Test Data	Expected Result	Actual Result	Acceptance
01	Title: citizen spot Latitude: -9.172264 Longitude: 121.3434	Success	Success	Accepted
01	Title: Latitude: -9.172264 Longitude: 121.3434	Fail (Title is required)	Fail	Accepted
01	Title: citizen spot Latitude: erd Longitude: dsfs	Fail (Latitude and Longitude should be numbers)	Fail	Not Accepted

---

- **Test Case ID:** BO-TC-20
- **Use Case ID:** BO-UC-20
- **Description:** Edit Location record.
- **Pre-condition:** Admin Logged in successfully.
- **Acceptance:** Not Accepted , 1 errors (similar to BO-TC-19)

- **Test Case ID:** BO-TC-21
- **Use Case ID:** BO-UC-21
- **Description:** Portal Setting.
- **Pre-condition:** Admin Logged in successfully.
- **Acceptance:** Not Accepted , (Need to be continued when proxy app is developed)

10.5.3 Testing Accounts

These accounts can be used to test the system:

- Admin back-office account:
  - **email:** admin@citizenPal.com
  - **password:** admin
- Android CitizenPal app:
  - **email:**omar@hotmail.com
  - **password:** 123123

10.6 API Resources

CitizenPal RESTful API is a set of resources. Each resources has properties. The following Tables go throw these resources in details.
The application root path is "/app". Then every resource has it own path. In order to use the path of a specific resource, you should add the root path. For example: user resources is "/app/users" and events resoruce is "/app/events". The following tables explain more:

Path	"/users/hello"
Method	GET
Produces/Consumes	Produces
Role	Permit all
Description	Resource used for testing, it produces a massage "Hello world API !" . It can be accessed by anyone

Path	"/users/login"
Method	POST
Role	citizen
Produces	OperationResoponse Object.
Consumes	User Object (email and password)
Description	Resource used for Resources used for log in, The Client send and User object loaded with email and password, The response is OperationResponse. If the log in success, OperationResponse will be loaded with full User Info
Path	"/users/{userId}/flag"
Method	GET
Role	citizen
Produces	OperationResoponse Object.
Consumes	null
Description	Retrieve user's flag . user flag indicates if the users has new events or not. The flag has the following values, 1 indicate there are new events, 0 indicate there are not new events.
Path	"/users/register"
Method	PUT
Role	PermitAll(because the citizen is not registers yet so any one can able to register new account)
Produces	OperationResoponse Object.
Consumes	User Object
Description	Register new citizen account. It returns the OperationResponse to tell if the operation is success or not.
Path	"/users/add"
Method	PUT
Role	Admin
Produces	OperationResoponse Object.
Consumes	User Object
Description	create new user account.

Path	”/users/admin”
Method	POST
Role	Admin
Produces	OperationResoponse Object.
Consumes	nothing
Description	Log in the admin, this does not need to consume any data because the the admin credentials should be sent on the header request. Http Request will not be permitted to access this resources if id does not pass the Authentication filter
Path	”/users/all”
Method	GET
Role	Admin
Produces	OperationResoponse Object. with list of citizens info in the output section
Consumes	nothing
Description	Used to retrieve citizens accounts info information.
Path	”/users/{userId}”
Method	DELETE
Role	Admin
Produces	OperationResoponse Object.
Consumes	nothing
Description	Used to delete citizen record info.
Path	”/users/{userId}”
Method	POST
Role	Admin
Produces	OperationResoponse Object.
Consumes	User Object
Description	Used to edit citizen info.

Path	”/users/{userId}”
Method	GET
Role	Admin
Produces	OperationResoponse Object (User Object on its output) .
Consumes	nothing
Description	Get Citizen account by citizen reocord id
Path	”/users/user/{userId}/events”
Method	GET
Role	Citizen
Produces	OperationResoponse Object. with list of events
Consumes	nothing
Description	Used to retrieve events of the citizen.
Path	”/users/{userId}/other/events”
Method	GET
Role	Admin
Produces	OperationResoponse Object.
Consumes	nothing
Description	used to retrieve all events that does not belong to the citizen.
Path	”/users/{userId}/events”
Method	POST
Role	Admin
Produces	OperationResoponse Object.
Consumes	Event Object
Description	Add Events to specific Citizen account.

Path	”/users/{userId}/events/{eventId}”
Method	DELETE
Role	Admin
Produces	OperationResoponse Object (User Object on its output) .
Consumes	nothing
Description	used to detach specific event from specific user.

Path	"/events/{userId}/flag"
Method	PUT
Role	Citizen
Produces	OperationResoponse Object. (output is Integer)
Consumes	nothing
Description	change the user Events flag as zero. Setting flag as zero means that these events has been already downloaded on the user device.
Path	"/events/user/{userId}"
Method	GET
Role	Citizen
Produces	OperationResoponse Object.(output is list of Event objects)
Consumes	nothing
Description	Used to retrieve citizen events by user id
Path	"/events/category/{userId}"
Method	GET
Role	Citizen
Produces	OperationResoponse Object.(output contains list of category)
Consumes	nothing
Description	Retrieve all citizen Category.

Path	”/events/status/{userId}”
Method	GET
Role	Citizen
Produces	OperationResoponse Object. (output contains list of Status objects)
Consumes	nothing
Description	Used to retrieve events Status.
Path	”/events/event/{eventId}/user/{userId}”
Method	GET
Role	Citizen
Produces	OperationResoponse Object.(output is Event object)
Consumes	nothing
Description	Used to retrieve specific event that belong to specific citizen
Path	”/events/add”
Method	PUT
Role	Admin
Produces	OperationResoponse Object.
Consumes	Event object
Description	Used to add new Event.
Path	”/events/{eventId}”
Method	GET
Role	Admin
Produces	OperationResoponse Object(output contains Event object)
Consumes	nothing
Description	Used to Retrieve event information by its id

Path	”/events/{eventId}”
Method	POST
Role	Admin
Produces	OperationResoponse Object.
Consumes	Event Object
Description	Used to Update the event information
Path	”/events/{eventId}”
Method	DELETE
Role	Admin
Produces	OperationResoponse Object.(output is Event object)
Consumes	nothing
Description	Remove event by its id. * WARNING: This Service deletes the events and all its relation to user.
Path	”/events/add”
Method	PUT
Role	Admin
Produces	OperationResoponse Object.
Consumes	Event object
Description	Used to add new Event.
Path	”/events/all”
Method	GET
Role	Admin
Produces	OperationResoponse.(output contians list of Events objects)
Consumes	nothing
Description	Used to Retrieve all events

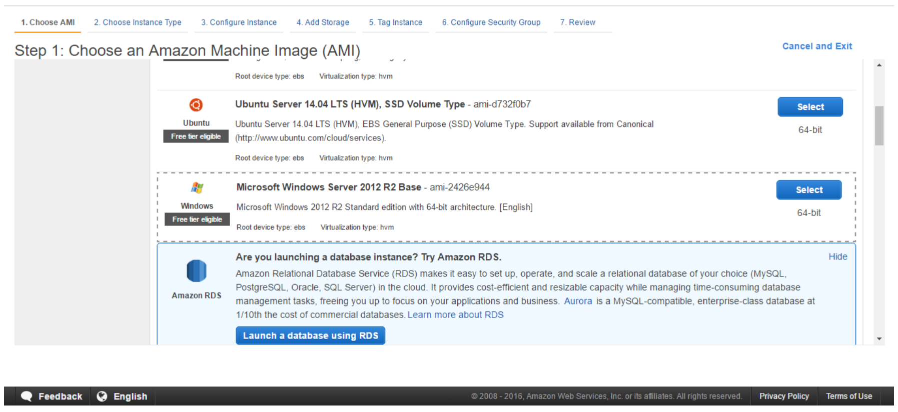
Path	”/events/{eventId}/users”
Method	GET
Role	Admin
Produces	OperationResoponse Object (output contains list of users)
Consumes	Event Object
Description	Used to retrieve users that belong to specific event
Path	”/events/{eventId}/users/{userId}”
Method	DELETE
Role	Admin
Produces	OperationResoponse Object.
Consumes	nothing
Description	Used to detach the citizen from being attached to specific event
Path	”/locations/all”
Method	GET
Role	Admin
Produces	OperationResoponse Object.(Output contains list of location objects)
Consumes	nothing
Description	Used to retrieve all locations records.
Path	”/locations/{locationId}”
Method	DELETE
Role	Admin
Produces	OperationResoponse.
Consumes	nothing
Description	Used to delete and specific location object.

Path	"/locations/add"
Method	PUT
Role	Admin
Produces	OperationResoponse Object.
Consumes	Location Object
Description	Used to add new location.
Path	"/location/{locationId}"
Method	GET
Role	Admin
Produces	OperationResoponse Object(output contains Location object)
Consumes	nothing
Description	Used to detach the citizen from being attached to specific event
Path	"/locations/{locationId}"
Method	POST
Role	Admin
Produces	OperationResoponse Object.
Consumes	location object
Description	Used to retrieve all locations records.
Path	"/status/all"
Method	GET
Role	Admin
Produces	OperationResoponse. (output contains list of status objects)
Consumes	nothing
Description	Used to retrieve all location records.

10.7 Deployments Steps

The deployment steps are going to expressed here with some important decisions.

1. Create AMI (Amazon Machine Image) : In able to create a AMI we should create an account and valid credit card. Lunching Instance of EC2. Windows Free tier has been chosen for the AMI. As the figure down below shows.



Step 1: Choose an Amazon Machine Image (AMI)

Root device type: ebs Virtualization type: hvm

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d732f0b7

Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs Virtualization type: hvm

Microsoft Windows Server 2012 R2 Base - ami-2426e944

Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]

Root device type: ebs Virtualization type: hvm

Are you launching a database instance? Try Amazon RDS.

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database of your choice (MySQL, PostgreSQL, Oracle, SQL Server) in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database management tasks, freeing you up to focus on your applications and business. Aurora is a MySQL-compatible, enterprise-class database at 1/10th the cost of commercial databases. Learn more about RDS

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 90: First step to choose the AMI

Create the AMI involves 7 steps. As it can be seen in the figure 90:

(a) Choose AMI.

(b) Choose Instance Type.

(c) Configure Instance Details

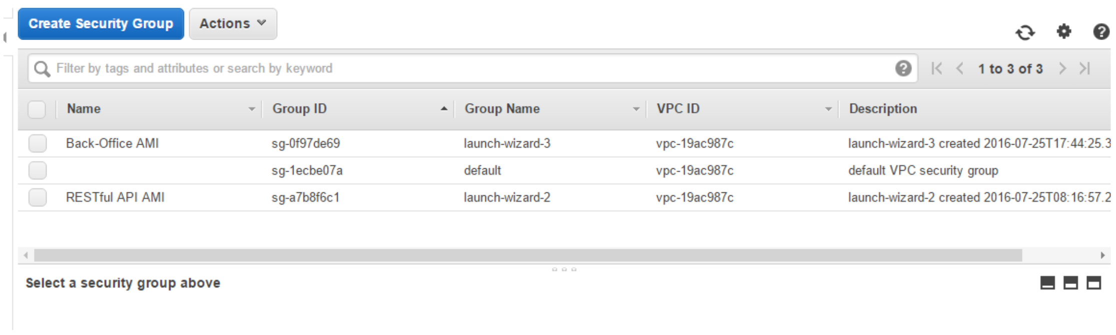
(d) Add Storage

(e) Tag Instance

(f) Configure Security Group .

(g) Review Instance Launch

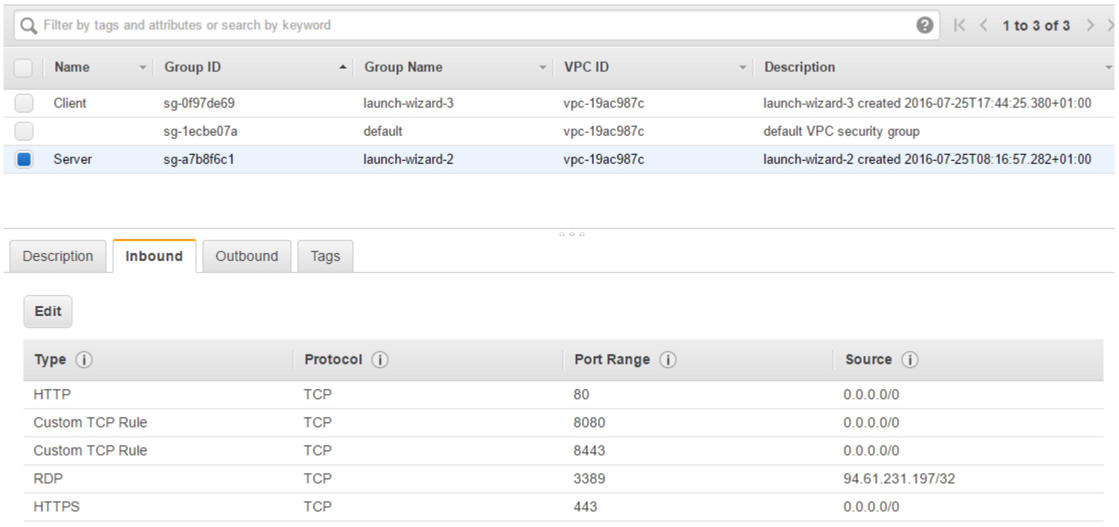
One of the important step is Configuring Security Group. Security group is working like a firewall for AMI. It has to be configured carefully.
Every Security Group has Inbound and Outbound. The Inbound has rules for for the ports and the protocols that should be allowed to access the AMI. Two Secruity Group should be created. One for every machine as it can be seen in figure 92 blow


Create Security Group
Actions
Filter by tags and attributes or search by keyword
1 to 3 of 3
Select a security group above

Name	Group ID	Group Name	VPC ID	Description
Back-Office AMI	sg-0f97de69	launch-wizard-3	vpc-19ac987c	launch-wizard-3 created 2016-07-25T17:44:25.3
	sg-1ecbe07a	default	vpc-19ac987c	default VPC security group
RESTful API AMI	sg-a7b8f6c1	launch-wizard-2	vpc-19ac987c	launch-wizard-2 created 2016-07-25T08:16:57.2

Figure 91: Security groups for CitizenPal AMIs

The rules for API RESTful API AMI should allow HTTPs, HTTP, with ports 80, 8080, 8443, 443 .



Filter by tags and attributes or search by keyword — 1 to 3 of 3

Name	Group ID	Group Name	VPC ID	Description
Client	sg-0f97de69	launch-wizard-3	vpc-19ac987c	launch-wizard-3 created 2016-07-25T17:44:25.380+01:00
	sg-1ecbe07a	default	vpc-19ac987c	default VPC security group
Server	sg-a7b8f6c1	launch-wizard-2	vpc-19ac987c	launch-wizard-2 created 2016-07-25T08:16:57.282+01:00

Description | Inbound | Outbound | Tags

Edit

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
Custom TCP Rule	TCP	8080	0.0.0.0/0
Custom TCP Rule	TCP	8443	0.0.0.0/0
RDP	TCP	3389	94.61.231.197/32
HTTPS	TCP	443	0.0.0.0/0

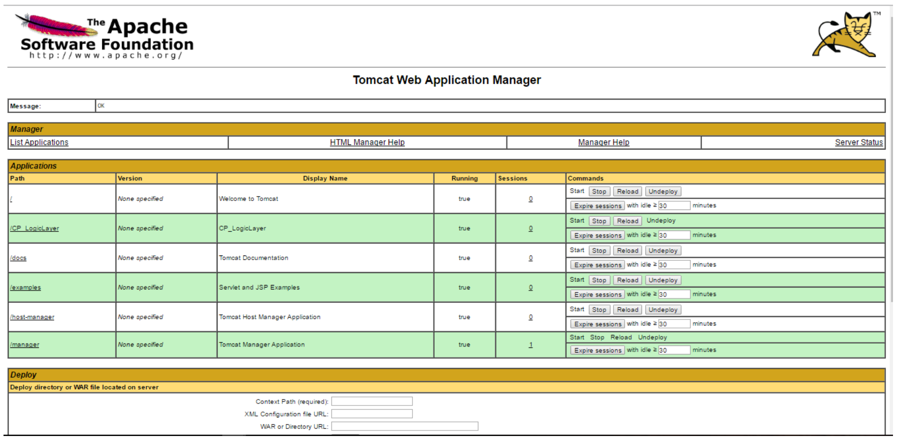
Figure 92: Security inbound roles for RESTful API machine

Note: RDP (Remote Desktop Protocol) used for remotly control the AMI. The back-Office AMI security group should add a role for HTTP with port 80, 8080 .

2. Installs Appache 7.0 And MYSQL Server on RESTful AMI. and configure the HTTPs and self-signed certificate.

   Note: The database can be deployed using RDS (Relational Database Service) . I chosed to Install the database server in the same machine of RESTful API because deploying on RDS cost extra money.

3. Deploying WAR file: Dynamic Java Web project can be extracted as WAR file. WAR files can be deployed using Appache app manager figure 93.



The Apache Software Foundation http://www.apache.org/

Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/CP_LogicLayer	None specified	CP_LogicLayer	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):
XML Configuration file URL:
WAR or Directory URL:

Figure 93: Appache app Manager

WAR file can be uploaded then deploy the app. As we can see the in the table of Applications in figure 93. we can see all the applications.

Since the Back-Office is an JavaScript app, Deploying the Back-office can be Copying the project to Appcache/webapps/[projectFolder] and restart the server.

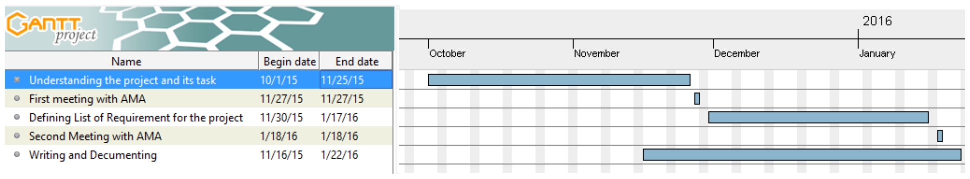
4. Change the IPs of the HTTP to match the Public IPs of the AMIs.

Testing all applications in order to make sure that they are responding to each other is an important too to make sure there is no interface does not work.

10.8 Project Planning

During the First Semester. The planning of the project, it was several attempt to get access to the public e-services portals by meeting with ama Agency which is located Lisbon. We had several meeting to present the idea. They showed acceptance of the idea it self. but a delay and postponement have been sensed.Therefor, The company changed the concept to build a prototype of the project to represent the idea.

The following graph explains the overview view plan for the first semester:



Name	Begin date	End date
Understanding the project and its task	10/1/15	11/25/15
First meeting with AMA	11/27/15	11/27/15
Defining List of Requirement for the project	11/30/15	1/17/16
Second Meeting with AMA	1/18/16	1/18/16
Writing and Decumenting	11/16/15	1/22/16

Figure 94: Gantt for first semester plan

1. **Understanding the project's context**
   Understand the overview of CitizenPal proposal, the main goals and its motivation.

2. **First meeting with AMA**
   First meeting with AMA the agency that is responsible for the e-Goverment Portals (Agência para a Modernização Administrativa) to explain the project and to collaborate with them.

3. **Defining List of Requirements**
   Defining list of concrete requirements and tasks that can be included in CitizenPal App.

4. **Second meeting with AMA**
   The second meeting with AMA was to confirm and collaborate with them over the project requirements and tasks.

5. **Writing and documenting**
   It is the process to describe and documents every activities and facts and write the report.

During the second Semester the project will have the following activities:

1. Specifying requirements in the project.
2. Build Architecture of the project.
3. Testing and evaluating.
4. Documenting.