University of Southampton

Faculty of Physical and Applied Sciences
Electronics and Computer Science

ODLAIN
A Mobile Application for Open Data in Latin America

By
Mauricio Toapanta Castillo

5th September 2014

A Dissertation submitted in partial fulfillment of the degree of
MSc Software Engineering
By Examination and Dissertation
University of Southampton

Faculty of Physical and Applied Sciences
Electronics and Computer Science

ODLAIN

A Mobile Application for Open Data in Latino America

By
Mauricio Toapanta Castillo

5th September 2014

A Dissertation submitted in partial fulfillment of the degree of
MSc Software Engineering
By Examination and Dissertation
Abstract

Open data and mobile applications are popular topics nowadays. European countries have combined these two concepts by implementing applications based on Open Data. The purpose of this project is to evaluate the introduction of these topics in Latin America. It is achieved by creating a mobile application that uses data from a Latin American open data platform. This application offers information about indicators similar to mobile applications developed for the statistical office of the European Union.

These are the stages of the project to meet its goal. First, this report looks for an open data project in Latin America, which provides information to implement indicators. Besides, it researches the technologies used to develop the application. Second, it describes the software development cycle used to implement the mobile application which includes the following phases. It starts with the definition and planning of the project. This phase includes an analysis to mitigate the risks that the project could have faced. The next two phases are focused on the design and implementation of software. The last phase explains the testing implemented over the mobile application. In order to guarantee the quality of the application, this phase is executed using automatic testing tools. Third, this document includes a discussion composed of two sections including evaluation and reflections. The former contains the evaluation of the final product, and the later analyzes the execution of the project. The information in the different sections of this report supports this discussion. Finally, it highlights some conclusions of this project, and suggests additional features for the application.

Keywords: Android, Open Data, Mobile Application, Junar, REST, Google APIs, Android Development Tools, Open Data Latinoamérica, Latin America Open Data.

Acknowledgments

Foremost, I would like to thank Dr. Abdolbaghi Rezazadeh, for his support towards the completion of this dissertation. His feedback was pivotal in the execution of this project.

Secondly, I would like to thank to my sponsor “The government of Ecuador” and my family who gave me the opportunity to study in this prestigious university.
Contents

1. Introduction ........................................................................................................................................... 6
  1.1. Project preliminaries ......................................................................................................................... 6
    1.1.1. Purpose ........................................................................................................................................ 6
    1.1.2. Scope .......................................................................................................................................... 6
    1.1.3. Impact analysis and stakeholders .............................................................................................. 7
    1.1.4. The problem ............................................................................................................................... 7
    1.1.5. The report structure ................................................................................................................... 7
  1.2. Background ...................................................................................................................................... 8
    1.2.1. Open Data .................................................................................................................................. 8
    1.2.2. Tools and Technology ............................................................................................................... 11
2. System Analysis ..................................................................................................................................... 16
  2.1. Overview description ....................................................................................................................... 17
  2.2. System requirements ....................................................................................................................... 19
    2.2.1. Functional requirements ............................................................................................................. 19
    2.2.2. Non-Functional requirements .................................................................................................... 24
  2.3. Project Management ....................................................................................................................... 24
    2.3.1. Assumptions .............................................................................................................................. 24
    2.3.2. Constraints ............................................................................................................................... 25
    2.3.3. Risk analysis ............................................................................................................................. 25
    2.3.4. Time management ..................................................................................................................... 29
3. Design .................................................................................................................................................... 31
  3.1. Logical Design .................................................................................................................................. 31
    3.1.1. System architecture .................................................................................................................... 31
    3.1.2. Interaction Diagrams ............................................................................................................... 32
  3.2. Data Base Design ............................................................................................................................ 36
    3.2.1. Model design ............................................................................................................................ 36
  3.3. Interface Design .............................................................................................................................. 37
    3.3.1. Home screen .............................................................................................................................. 37
    3.3.2. Indicator screen .......................................................................................................................... 38
    3.3.3. Chart screen .............................................................................................................................. 39
    3.3.4. Map screen ............................................................................................................................... 40
4. Implementation ................................................................. 41
   4.1 Implementation of initial configuration ............................... 41
      4.1.1 Model layer Implementation .................................. 41
      4.1.2 Controller layer implementation ............................... 41
      4.1.3 View layer implementation ..................................... 42
   4.2 Implementation of indicators (Information per Year) .............. 43
      4.2.1 Model layer Implementation .................................. 43
      4.2.2 Controller layer implementation ............................... 44
      4.2.3 View layer implementation ..................................... 45
   4.3 Implementation of indicators (Historical Information) .......... 46
      4.3.1 Model layer Implementation .................................. 46
      4.3.2 Controller layer implementation ............................... 46
      4.3.3 View layer implementation ..................................... 47
   4.4 Implementation of chart diagrams .................................. 48
      4.4.1 View layer implementation .................................... 48
   4.5 Implementation of the map of Latin America ....................... 50
      4.5.1 Model layer Implementation .................................. 50
      4.5.2 Controller layer implementation ............................... 50
      4.5.3 View layer implementation ..................................... 50

5. Testing ............................................................................ 51
   5.1 Integration testing ..................................................... 51
   5.2 System Testing ........................................................ 52

6 Discussion ......................................................................... 53
   6.1 Evaluation .................................................................... 53
      6.1.1 Results .................................................................. 53
      6.1.2 Analysis .............................................................. 54
   6.2 Process execution ...................................................... 56
   6.3 Tools selection .......................................................... 57
   6.4 Management Plan ....................................................... 58

7 Conclusions and future work .............................................. 59
   7.1 Conclusions ............................................................... 59
   7.2 Future work .............................................................. 60
   7. References ................................................................. 62
Appendix ........................................................................................................................................................................... 64
  A. Open Data Latin America datasets .................................................................................................................................. 64
  B. Junar Latin America platforms ....................................................................................................................................... 66
  C. System communication architectures .................................................................................................................................. 67
  D. ODLAIN map implementation ........................................................................................................................................... 68
  E. Integration Testing: IT_01 ..................................................................................................................................................... 69
  F. Integration Testing: IT_02 ..................................................................................................................................................... 70
  G. Integration Testing: IT_03 ..................................................................................................................................................... 71
  H. System Testing: ST_01 ......................................................................................................................................................... 73
  I. System Testing: ST_02 ......................................................................................................................................................... 75
  J. System Testing: ST_03 ......................................................................................................................................................... 77
  K. Tasks execution .................................................................................................................................................................... 79
1. Introduction

1.1. Project preliminaries

1.1.1. Purpose

The aim of this project is to create a mobile application which consumes data from open data projects. This project uses a Latin-American open data initiative in order to analyze the status of the open data initiative in Latino America. ODLAIN (Open Data Latino America INdicators) is the name of the application developed in this project.

This project intends to create a mobile application which will offer information about indicators. The main source of data is the project called OpenData Latinoamérica. The aim of ODLAIN is transform this data into information. ODLAIN offers indicators like other applications which provide indicators for areas like banking, marketing, and planning.

This project focuses on using software development tools to implement a mobile application; on the other hand, this dissertation is to analyze the open data initiative. It studies the open data approach in Latino America. Besides, it creates a mobile application based on data from Latino-American open data repositories.

1.1.2. Scope

Section 1.1.1 cites that ODLAIN is to evaluate the status of open data in Latino America. The following list of concerns and functionalities is to define ODLAIN.

- The data for this application is gathered from Junar platforms. It is achieved using Junar APIs.
- This application is for mobile devices that use Android operating systems.
- Users are able to access to information based on indicators. Section 2.1 describes them.
- It is not compulsory that the design of the application regards the addition of more indicators in the future.
- Only requirements declared as mandatory in section 2.2 are compulsory features of ODLAIN.

Section 1.2.2 summarizes the tools and technology used to develop this application.
1.1.3. Impact analysis and stakeholders

Nowadays, the quick access to information is essential. The Open Data Latin America project is an open repository which store data from governmental institutions. It means that it is a source of truthful information. If this data becomes into information, it will be useful for people interested in the development of Latino America.

Mobile applications are popular because they can be installed in mobile devices like tablets and mobile phones. People like governmental authorities or managers of organization prefer to use their mobile devices instead of their personal computers. The development of a mobile application based on indicators provides a practical, efficient and accessible tool for governmental employees, staff of international institutions, and students. Finally, this tool enhances and promotes the usability of open data repositories in Latin America

1.1.4. The problem

Many organizations use open data concepts, particularly in Europe and North America. The transparency of the United States and United Kingdom governments has been improved by the publication of data in the official open portals [1,2]. Web and mobile applications have been developed to access to this data. Nevertheless, regions around the world do not evolve at the same pace. In order to evaluate the status of the open data initiative, it is important to know his introduction in Latino America.

A project is evaluated when users or applications use its features. For this reason, it is compulsory to find an open data project in Latino America and evaluate it. This permits state their maturity and relevance. Although there are a couple of open data projects in Latino America, there are no many applications that use their APIs.

1.1.5. The report structure

This document is composed of six chapters which include introduction, system analysis, design, implementation, testing, discussion, and conclusions and future work.

First chapter refers to a general description of the project and ODLAIN. This chapter concludes with a background section. Second chapter delineates ODLAIN. It details the features and constraints of the application. At the end of the section is mentioned the plan to execute this project. Third chapter refers to the design. It offers information about the application design phase. It considers diagrams and models for user interfaces, persistence entities, classes, and so on. Fourth chapter describes the implementation process of ODLAIN in order to create technical documentation to support this project. Fifth chapter has been used to detail the testing phase. It explains the automatic testing carried out. Sixth chapter is
composed of the evaluation performed by users and reflections about this project. The last chapter is about the conclusions of the project and ODLAIN’s future work.

1.2. Background

In this section, we divide the literature review in two parts. This lets to study the business and technical views of this project. First, we analyze the open data approach since it is the commercial concept used to promote ODLAIN. We study its definition as well as its introduction in Europe and Latin America. Second, we inspect the tools and technologies used in this application. We cite Android Development Tools since it is the development framework that we used to develop ODLAIN. Besides, we summarize Junar because it is the open data platform that provides the data to our software. Moreover, we investigate REST since it is the type of service that open data projects are using nowadays, particularly the open data platforms that we have cited in this report. Furthermore, we review Android, which is the target mobile operating system of ODLAIN.

1.2.1. Open Data

Open Data is an approach which intends that data will be available to everybody. It also states free redistribution of data. This means to avoid copyright restrictions. Open data can be applied to data stored in academic, scientific, economical, statistical, and governmental repositories [3]. Several institutions around the world uses open data concepts particularly governmental organizations. It allows popularity of open data.

1.2.1.1. Europe as case study

United Kingdom is one of the countries that promote to open up data without restrictions. He created an Open Government license. There were many campaigns claiming the use of open data approach in the British government. Data.gov.uk site was created on 2009 as a result of them. Currently, over 9000 data sets are contained in this site. There are over 300 applications published to offer information about areas like transportation, health, location, weather, and so on [4,5].

a. Statistical office of the European Union (EUROSTAT)

The following section is based on the information of the Eurostat official web site [6]. The statistical office of the European Union is using RDF ontologies and creating mechanisms to offering updated
information. Eurostat has implemented two mobile applications, country profiles and EU economy apps, which are available for Android, iPhone and iPad.

Eurostat delivers many options to access to his data. This data is accessed statically or dynamically. Static access refers to the datasets that can be downloaded using the data base browser tool provided in Eurostat website. These datasets can be downloaded in formats such as xls, csv, HTML, pc-axis, spss, pdf, and tsv; on the other hand, dynamic access refers to the use of web services which allows to developers retrieve structured data. Data and metadata is accessed using SOAP SDMX 2.0, SOAP SDMX 2.1, or REST SDMX 2.1.

REST SDMX 2.1 API lets access to the database of Eurostat using the concept of REST and SDMX (Statistical Data and Metadata Exchange). REST defines the data as a resource. Section 1.2.2.3 contains more information about REST. SDMX is a statistical standard used to interchange information based on XML.

REST uses HTTP methods to manipulate resources. This API only allows data retrieval, so only GET method is implemented. It also lets access to information like dataset metadata, the list of datasets, datasets, and specific data of a dataset. There are many options to filter data while it is being retrieved.

The general features of the Eurostat Open Data platform are listed as follow:

- There is not data versioning because the database always contains the last version of the data.
- The number of cells in a single response is limited to one million.
- There is a file naming convention for REST responses.
- APIs carry out an asynchronous file delivery process when the limit of cells in a request has been exceeded.

1.2.1.2. Latin America as case study

a. Economical Commission for Latin America and the Caribbean (CEPAL)

CEPAL is a regional commission member of the United Nations (UN). CESPALSTAT is an API used to make public the data base of CEPAL which is called BDI-CEPALSTAT. It allows to access to this data using Open Data concepts. The data retrieved from CESPALSTAT can be gathered in XML format [7].
b. **Open Data Latin America**

Some countries in Latin America like Chile, Costa Rica and Argentina have become to publish information using Open Data initiatives. A project called Open Data Latin America\(^1\) has created a centralized storage for regional information. This project uses APIs that lets organizations collect and publish data [8]. It is sponsored by the World Bank institution which makes to this project sustainable. Data from this platform can be collected in XML, CSV, HTML and JSON formats. Open Data Latin America is presented as a stable initiative which lets to have a referent in Latin America [9].

Open Data Latin America as well as CESPALSTAT APIs offer REST services to allow users access to data using queries. Nevertheless, CEPAL differs from Open Data Latin America since the former opened its data base whilst the latter created a new data base composed of data from different sources. The Open Data Latin America project lets access their datasets. They are developed using an open data platform called Junar. Besides, the open data platform promotes and provides access to datasets from other platforms whose are also based on Junar. More information about Junar is in section 1.2.2.2. The following information is about platforms based on Junar.

**b.1. Open Data Argentina**

The initiative Open Government of the city of White Bay\(^2\) in Argentina has created a website to access to public data. It has improved the transparency and the information access time of this governmental entity. It published three types of data resources, including views, collections, and graphics. The datasets are clustered in ten categories. Some examples of these categories are budgets, purchases, health, transport, and so on. The platform is composed of over 190 data resources which are accessed through an API. A KET is the requirement to use this API which uses it as security mechanism [10].

**b.2. Open Data Peru**

The Peruvian governmental plan called “The Action Plan of the Open Government Partnership of Peru”\(^3\) tries to promote the transparency and community collaboration among public institutions. Open data platforms have been recognized as a good governmental practice on terms of transparency and information access. Information access is one of the principal aims of the plan cited above. The metropolitan municipality of Lima has developed an open data website which offers a sort of data resources. They can be of type view, graphic or a combination of them. There are ten categories of

---

1. Open Data Latin America is translated to Spanish as OpenData Latinomérica, which is the original name of the project
2. Open Government of the city of White Bay is translated to Spanish as Gobierno abierto de la ciudad de Bahía Blanca, which is the original name of the project
3. The Action Plan of the Open Government Partnership of Peru is translated to Spanish as Plan de Acción Sociedad de Gobierno Abierto del Perú, which is the original name of the project
datasets which includes business development, investment projects, health, mobility, transport, etc. The request of a KEY is compulsory to access to the API for developers [11].

b.3. Open Data Chile

Government of Chile has implemented an official open data website to allow people access his public information. The information is presented in tables and charts which can be either bar or pie charts. Similar to the open data projects cited above, this project provides collections to group data resources. There are over 1150 datasets published where data comes from public and private institutions. However, most of it comes from governmental institutions. The datasets are categorized in over 15 categories. It implements an API that lets developers access to datasets. The web site offers 18 applications that use the data sets published on the platform. These applications are either web or mobile apps [12].

1.2.2. Tools and Technology

1.2.2.1. Android Development Tools (ADT)

The following section summarizes some aspects of ADT. The most of this is based on the information of the ADT official site [13]. It is an Eclipse extension that offers a development environment to create Android applications. It can be added to an existing eclipse version or download the full eclipse android package from the official web site. As other eclipse versions, it is multi-platform since is based on Java. This IDE help developers to test, debug and package applications.

Some features of this tool include:

- The development environment lets edit and navigate easily between java and xml documents. It offers templates to build components and projects.
- In order to build graphical interfaces, it allows use drag and drop mechanisms, including compatibility control before build the application.
- It provides two debugging options. The first is creating a virtual mobile device by selecting the features of the mobile to be virtualized. The second is debugging directly on a real mobile device over a USB connection. Performance and correctness examinations can be executed because of the use of the debug option on a device. Deployment and testing can also be carried out.
- Testing is enabled to create user interface and script based tests. Finally, it lets to compile and pack C or C++ code into a single binary to allow multiple architecture compatibility.
The process to develop an application is synthetized in four steps:

a. **Set-up**

It refers to the installation and configuration of the IDE. It is compulsory to install Android SDK. If the full ADT package was downloaded then Android SDK installation is omitted. It is also mandatory create android virtual devices or activate the use of mobile devices.

b. **Development**

In this step the project is created using template wizards or from scratch. Although a project is created from scratch, the IDE provides a basic structure of an android project including its configuration and resource files.

c. **Debugging and Test**

ADT creates a debuggable file every time the user saves the project. It is an apk file which can be installed in a mobile or a virtual device. Once it was installed, the Android SDK lets to debug and test the application.

d. **Publishing**

The development environment allows configure the project to build an application to be released and distributed. It uses release mode to realize these activities.

1.2.2.2. **Junar**

The following data is based on the information of Junar official website [14]. Junar is an Open Data platform that lets organizations collecting and publishing data. It offers an easy way for opening data turning it into understandable and searchable information. This platform has been used by over 200 organizations around the world. It is a paid service which is offered in annual subscriptions (750 USD per 12 months). Junar advantages are:

a. Independence between the front end and back end applications. It can offer components for other front end applications.

b. It does not modify the original data.

c. Provide APIs to support connectivity to external as well as internal applications.

This cloud-based platform follows a simple work flow which is its business strategy. It is composed of five activities.
a. Collect
The process starts collecting data from different sources. Junar platform provides tools for gathering and managing datasets.

b. Enhance
It lets to create resources like tables, maps, and charts from the datasets. It offers to users a readable mean to interpret information. The tool called dashboard is to organize resources.

c. Publish
Junar allows his customers to select which datasets and resources are available for internal and external users. It implements Junar APIs. The purpose of these APIs is to give access to the platform.

d. Share
The purpose is to become searchable the data of the platform. Junar optimize data to carry out this feature. It shares data via Facebook, Twitter and Junar Community.

e. Analyse
It provides reports about popularity of the resources. It also informs about data consuming generated by Junar APIs. This lets to control the usability of the platform and its open data initiatives.

The annual subscription includes features such as domain hosting, 15 datasets, 50 GB of Data Storage, 250 GB of data delivery bandwidth per day, 250 K API requests per day, unlimited data views, and APIs. Additional services can be hired from the Junar team. These services are APIs customization, internationalization, project consulting, portal designing, data analyzing, user training, and mobile application customization. Besides, extra data sets, and bandwidth can be acquired. The pricing for these specifications are not available in the web site or official documentation.

1.2.2.3. REST
Defined as the acronym of REpresentational State Transfer, REST is an architectural style composed of three elements and some constraints applied to these elements. Depending on the constraints used in an
architecture, it can be stated some REST styles including null, client-server, stateless, cache, uniform interface, layered system, and code-on-demand [15].

The architectural elements are clustered in three groups such as data elements, connectors, and components. REST refers to the state and nature of the data elements of an architecture. The data elements of REST can be listed like resources, resources identifiers, representations, metadata resources and control data. Resources are abstractions of information and their identifiers are used to identify them to carry out interaction between components. Representations refer to the formats used to allow components manipulate resources.

Connectors try to encapsulate all activities executed to transfer representation or access to resources. The types of connectors are the client, server, cache, resolver, and tunnel.

The REST architecture components for a distributed hypermedia system are origin server, gateway, proxy, and user agent. Gateways and proxies are intermediary components, so they act like servers or clients in order to execute operations. The origin server is the source for representations. Finally, a user agent is the client which initiates the requests.

Views in REST are used to describe how the elements detailed above create the architecture. The architectural views include process, connector and data views.

When these constraints are applied to Web Services instead of a web architecture, it creates a new model. This model is known as the Richardson maturity model (RESTful maturity model). RESTful states four levels depending on the constraints applied. These levels go from level zero to three. These levels represent the architecture style used to build the web service.

1.2.2.4. Android

It is an operating system (OS) developed for mobile devices, particularly smartphones although it is being used in other devices like tablets, televisions, cameras, and so on. One of the principal features of this OS is to react to touch input requests. Currently, touch screen devices are popular, and that is the reason for the popularity of Android. This OS is based on the Linux kernel, follows an Open source model, and has to Google as his company owner. Since 2012 over 70% of the smartphones around the world use Android OS [16].

The android architecture is composed of four layers. On the bottom is located the Linux kernel, and on the top is the applications layer which uses the application framework layer to run itself. Over the kernel layer
is the library layer which contains the libraries used by the OS. The library layer contains a sub-layer called android runtime composed of core libraries and the Dalvik virtual machine.

Although Android source code is under Apache license 2.0, most devices which use Android OS replace the Android software by its own privative applications.

The following statements are some features of Android [17].

- Easy and friendly interface based on the use of input touch mechanism. Manipulation of objects either using the screen or a virtual keyboard.
- Applications can be installed by downloading from online stores or manually since the device storage.
- The minimum hardware requirements are 512 MB of RAM, x86 architecture processor or similar, and support to OpenGL ES 1.1 for versions after 2013.
- In contrast to desktop OS, Android tries to consume the minimum power since mobile devices are battery powered. Applications can be stopped to save power or ram memory. This memory management intends to solve the hardware and RAM limitations that mobile phones have at the moment.
2. System Analysis

This section describes the behavior of ODLAIN. The business logic of this project is summarized in figure 2.1. This figure is based on the general features of ODLAIN stated in section 1.1. Sections 2.1 and 2.2 details the activities of this workflow by defining requirements and explaining how they were established.

![System workflow diagram]

Figure 2.1 System workflow

This workflow shows how ODLAIN works. If users require information about an indicator, they must use this process. This represents the simplest navigation through the application where “Start” represents opening the application and “End” symbolizes closing it. The following defines the functionality of the application. First, it describes the available indicators in the application. Second, it details the functional
behavior of the application which is based on the workflow described above. Finally, it lists the non-functional requirements.

In order to development this project, we follow an Agile Development because it provides flexibility to modify the project development plan regarding time restriction constraints.

2.1. **Overview description**

ODLAI intends to follow the aims of Eurostats mobile applications. ODLAIN shows statistical indicators based on information of Latin America gathered from Junar repositories. Graphic charts are provided to interpret indicators. The data in Open Data Latin America\(^4\) is not enough to generate indicators as the ones in Eurostat or Cepal platforms. Tables in appendix A show a list of the data available in the Open Data Latin America repository:

Some datasets in the list of Open Data Latin America resources refers to links to other Junar repositories. It means that they are not into the Junar Latin American repository. Tables in appendix B detail these datasets.

In order to collect enough data to calculate the indicators, ODLAIN uses connections to other Junar APIs in addition to the connection to Junar Latin America API. For this reason, it must be clarified that the accuracy of the information offered by ODLAIN depends on the accuracy of the data provided by the Junar Latin American platforms. As a result, this application manages five indicators.

a. **Public investment.**

This is an economic indicator. The aim of public investment is to show how much money the governments are investing. In ODLAI, Public investment is the amount of money that a government invests every year. It is the obtained by adding the investment amounts that every governmental institution does in a year. The data provided in Junar Latin American platforms permits to determine this amount for Argentina, Chile, Peru and Uruguay. Nevertheless, Argentina and Peru amounts are partial accurate because the data regards public investment in a sector of the country instead of a national investment.

b. **Academic achievement level.**

This is an educational indicator. The purpose of academic achievement level is to show the educational level of a country based on the number of people who approve a course. It reflects the improvement of

---

\(^4\) Open Data Latin America is translated to Spanish as OpenData Latinomérica which is the original name of the project
their academic knowledge. In ODLAI, it is the rate between students that approve and students registered for a specific academic program level (approved students / registered students). The open data platforms provide data to determine the academic achievement level in Chile, including historical information from 2006 to 2012.

c. Average salary.

Average salary is an economic indicator. It is used to show the income of a person in a country. In ODLAI, it is the average salary per hour in governmental institutions. It comes from calculating the average salary in a particular governmental institution taken into account the real salary, but not the official one. Junar platforms let determine this indicator for Argentina, Bolivia and Chile.

d. Trade balance rate.

Trade balance rate is an economic indicator which shows the exportation and importation relation in a country. In ODLAI, it is the rate between the amounts of exports and imports (exports/imports). Costa Rica is available to provide this data through the Open Data Latin America\(^5\) project. It offers historical information from 2000 to 2010. This indicator is one of the most important economic indicators, but the lack of data into the open data sets do not let ODLAIN provides this information for all countries.

e. Notifiable diseases

This is a health indicator. It is important to provide indicators of health in our application. According to the data analyzed, there is not relevant data to generate this type of indicators. The datasets which refers to health topics are originated in Chile. There are two options to generate health indicators. The first is the number of notifiable diseases in a year, which is obtained by adding all the dangerous disease cases listed in a country. The second is the available pharmacies rate, which is the relation between the number of pharmacies available and the number of inhabitants in a location. The former option is selected as indicator since it provides to users more accurate information than the second. In ODLAI, this indicator represents the amount of diseases notified in a year. Notifiable diseases are diseases considered dangerous by a government. The other Junar platforms used in this project have not data to include in this indicator.

The indicators described above were stated after analyze the data available in Open Data Latin America. Some datasets are not used because their data is not relevant for the aims of this project. Although it is not included in the scope of this project, ODLAI collect data from other Junar platforms in order to support

---

\(^5\) Open Data Latin America is translated to Spanish as OpenData Latinomérica which is the original name of the project
the indicators. These platforms are official governmental projects in Latin America described in section 1.2.1.2.

The implementation of this new feature requires a reassessment of the complexity of our application. Besides, the data analyzed is not standardized. Rest APIs return a standardized structure of a resource, but the structure of the result attribute depends of the owner of the data. An attribute named as “result” contains the data of the table. It means that every country sends its data in the format they are able to provide. These facts increase the complexity of the implementation and reduce the accuracy of the indicators offered by ODLAIN.

2.2. System requirements

2.2.1. Functional requirements

Every requirement has been assigned a code of format FR_ [TYPE] _ [NUMBER] where FR represents to functional requirements, [TYPE] denotes if it is part of documentation (DOC) or part of the application (APP), [NUMBER] is the number assigned.

The importance column for the functional requirements can be one of three options including Mandatory, High Priority, or Optional. Mandatory requirements will be built first. On a second phase, high priority requirements will be developed. Finally, if the remaining time for the project deadline is good enough, the optional requirements will be implemented.

The functional requirements have been joined up in 7 groups. The first includes the main options offered by the application, the following five groups cluster the requirements for indicators stated in section 2.1. The last group refers to general aspects of the application

2.2.1.1. Select available options

This group of requirements describes the aim of this application which allows select indicators and countries to display information based on these selections. The FR_APP_01 requirement explains this functionality by stating constraints of it. The remaining requirements detail its compulsory features

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_APP_01</td>
<td>User selects options</td>
<td>The application lets users select an indicator and select up to two countries to show information based on them.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_02</td>
<td>Indicator selection</td>
<td>The user must be able to select one of the five Indicators stated on section 2.1. This</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>
Table 2.1 Select available options requirements

| **FR_APP_03** | First country selection | The user must be able to select one of the countries of the Latin American region which is stated as the first country. | Mandatory |
| **FR_APP_04** | Second country selection | The user must be able to select one of the countries of the Latin American region which is stated as the second country. | Mandatory |
| **FR_APP_05** | Show results per Country | The application must display information based on the parameters selected in FR_APP_02, FR_APP_03, and FR_APP_04. It will also let users move to FR_APP_8, FR_APP_13, FR_APP_18, FR_APP_23 or FR_APP_28. When this option is selected, FR_APP_08 is compulsory. | Mandatory |
| **FR_APP_06** | Show map of Latin America | The application must display a map of Latin America. This map displays an indicator (selected in FR_APP_02) for every country. When this option is selected, FR_APP_02 selection is compulsory. | Optional |

### 2.2.1.2. **Public investment indicator**

This section defines the functional behavior of the public investment indicator. The following table is structured to describe this behavior. FR_APP_07 states briefly what is intended about the indicator. The term “calculate” in FR_APP_07 refers to the group of activities carried out to obtain the public investment value. The remaining requirements show how this indicator information will be used.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_APP_07</td>
<td>Calculate the public investment indicator</td>
<td>In order to calculate the public investment indicator, the application will look for all the governmental institutions of a country and add the inversion done for every one of them.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_08</td>
<td>Public investment comparison of countries</td>
<td>The application displays information related to public investment for two countries of Latin America. Users select countries in FR_APP_03 and FR_APP_04. It also lets users to see FR_APP_9 for a selected country. Besides, It lets user move to FR_APP_11.</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>
2.2.1.3. **Academic achievement level indicator**

This section defines the functional behavior of academic achievement level indicator. The following table is structured to describe this behavior. FR_APP_12 states briefly what is intended about the indicator. The term “calculate” refers to the group of activities executed in order to obtain the academic achievement level. The other requirements show how this indicator information should be used.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_APP_12</td>
<td>Calculate the academic achievement level indicator</td>
<td>This is to calculate the academic achievement level indicator. Regarding the number of students registered and approved in a program.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_13</td>
<td>Academic achievement level comparison of countries</td>
<td>This is to display information related to academic achievement level for one or two countries of Latin America. The countries are selected in FR_APP_03 and FR_APP_04. It will also let users to see FR_APP_14. Finally, It allows users navigate to FR_APP_16.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_14</td>
<td>Academic achievement level per country</td>
<td>This is to display information per year about academic achievement of the country selected in FR_APP_13. It also allows users move to FR_APP_15.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_15</td>
<td>Academic achievement level bar graph</td>
<td>This is to display a bar graph for the information presented in FR_APP_14.</td>
<td>High Priority</td>
</tr>
<tr>
<td>FR_APP_16</td>
<td>Academic achievement level pie graph</td>
<td>This is to display a pie graph for the information presented in FR_APP_13.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Table 2.3 Academic achievement requirements
2.2.1.4. **Average salary indicator**

This section defines the functional behavior of average salary indicator. The following table is structured to describe this behavior. Requirement FR_APP_17 states briefly what is intended about this indicator. The term “calculate” in FR_APP_17 refers to the activities to obtain the average salary value. The remaining requirements in the table show how this indicator information is used.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_APP_17</td>
<td>Calculate the average salary indicator</td>
<td>In order to calculate the average salary indicator, the application will look for a governmental institution and calculate the average salary among its employees.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_18</td>
<td>Average salary comparison of countries</td>
<td>The application displays information related to the average salary of two countries of Latin America. The countries are selected by users in FR_APP_03 and FR_APP_04. It also lets users to see FR_APP_19 for a selected country. Besides, It allows users navigate to FR_APP_21.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_19</td>
<td>Average salary per country</td>
<td>The application displays information per year related to average salary of the country selected in FR_APP_18. It will allow user navigate to FR_APP_20.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_20</td>
<td>Average salary bar graph</td>
<td>The application displays a bar graph for the data gathered in FR_APP_19.</td>
<td>High Priority</td>
</tr>
<tr>
<td>FR_APP_21</td>
<td>Average salary pie graph</td>
<td>The application displays a pie graph for the information show in FR_APP_18.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Table 2.4 Average salary indicator requirements

2.2.1.5. **Trade balance rate indicator**

The bellow table defines the functional behavior of trade balance rate indicator. It is structured to describe it as follow. FR_APP_22 states briefly what is intended about the indicator. The term calculate in FR_APP_22 refers to the activities that should be done in order to obtain the public investment value. The remaining requirements show how this indicator information will be used.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_APP_22</td>
<td>Calculate the trade balance rate indicator</td>
<td>This is to calculate the trade balance rate indicator. Regarding the imports and exports done.</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>
Table 2.5 Trade balance rate indicator requirements

2.2.1.6. Notifiable diseases indicator

In order to define the functional behavior of notifiable diseases indicator, it is specified the following table.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_APP_27</td>
<td>Calculate the notifiable diseases indicator</td>
<td>In order to calculate the notifiable diseases indicator, the application will add all occurrences of dangerous disease cases found.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_28</td>
<td>Notifiable diseases comparison of countries</td>
<td>It should be possible to display the information related to notifiable diseases for two countries of Latin America. User select countries in FR_APP_03 and FR_APP_04. It will also let users to see FR_APP_29 for a selected country. It also allows users navigate to FR_APP_31.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_29</td>
<td>Notifiable diseases per country</td>
<td>It should be possible to display information per year related to notifiable diseases of the country selected in FR_APP_28. It will allow users navigate to FR_APP_30.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>FR_APP_30</td>
<td>Notifiable diseases bar graph</td>
<td>It should be possible to display a bar graph for the information showed in FR_APP_29.</td>
<td>High Priority</td>
</tr>
<tr>
<td>FR_APP_31</td>
<td>Notifiable diseases pie graph</td>
<td>It should be possible to display a pie graph for the information showed in FR_APP_29.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Table 2.6 Notifiable diseases indicator requirements
FR_APP_27 states briefly what is intended about the indicator. The term “calculate” refers to the group of activities performed to obtain the number of notifiable diseases generated in a year. How this indicator information is used is into the remaining requirements.

2.2.2. Non-Functional requirements

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF_01</td>
<td>The availability of the application is 98%.</td>
<td>Availability</td>
</tr>
<tr>
<td>NF_02</td>
<td>User will be familiar with the application after three time of use it.</td>
<td>Usability</td>
</tr>
<tr>
<td>NF_03</td>
<td>The application response to user requests within 4 seconds</td>
<td>Performance</td>
</tr>
<tr>
<td>NF_04</td>
<td>Number of target operating systems is one which is Android versions 4.x</td>
<td>Portability</td>
</tr>
<tr>
<td>NF_05</td>
<td>Everybody who has access to the application by installing it must be allowed to use it.</td>
<td>Security</td>
</tr>
<tr>
<td>NF_06</td>
<td>The interface of this application will support two languages which includes English and Spanish</td>
<td>Usability</td>
</tr>
</tbody>
</table>

Table 2.7 Non-Functional requirements

2.3. Project Management

In this section we plan the project development. It starts defining assumptions and constraints for the application. Later, it is analyzed the risks that can influence over the project execution and their impact.

2.3.1. Assumptions

The following assumptions are stated in order to develop ODLAIN mobile application.

f. The availability of Junar platforms is 99.50%
g. Basic versions of Junar APIs have connectivity limitation.
h. APIs used in this project are basic Junar APIs.
i. Spanish and English are assumed as official languages for the application.
j. Target mobile devices for this app are touch-screen smart phones.
2.3.2. Constraints

Four categories are defined for this project with the aim of join up and organize its concerns.

2.3.2.1. Software

Junar APIs have connectivity limitations. Particularly, Open Data Latin America is limited to 10,000 requests/month and 1 request/second. Although it is possible to ask the improvement technic features of the APIs, the development of this application does not follow the process required to improve these features because of time limitations. The accuracy of the information offered in ODLAIN depends on the accuracy of the data gathered from Junar APIs. The accuracy of indicators depends on how updated the datasets of third-part projects is.

Interface development is limited to the use of free software tool. No paid tools are used so a limitation in usability of the app can be found. Android version limitations about compatibility produced by development frameworks force us to create an interface limited to compatibility for a group of Android versions.

2.3.2.2. Hardware

Hardware constraints are not defined by our application, but it is by the version of Android OS used in the mobile device where ODLAIN app is going to be installed. The requirement NF_04 of section 2.2.2 details requirements about Android, besides section 1.2.2.4 describes mobile hardware requirements for Android.

2.3.2.3. Language

English language is taken into consideration to any phase of this project, as well as the application development cycle. It includes the code generated, diagrams, database meta-data, etc. Countries of Latin America have their own official languages. This makes difficult standardize the language of the application. For this reason, data which is not stored in English language will remain in their original language.

2.3.3. Risk analysis

This section is composed of three parts. It starts identifying the project risks. On a second phase, risks are analyzed. Finally, it is defined a list of strategies to prevent risks.

2.3.3.1. Risk Definition

In order to define the risks of this project, the research is divided in activities and sub activities. Besides, we declare the risks for the sub activities. A number of 18 risks are outlined for the project.
<table>
<thead>
<tr>
<th>Project activities</th>
<th>Sub activities</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. System Analysis and Research</strong></td>
<td>1.1. System definition</td>
<td>1.1.1. Scope and restrictions not well defined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.2. Lack of experience to estimate the project phases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.3. Application not delivered on time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1.4. High implementation cost</td>
</tr>
<tr>
<td></td>
<td>1.2. Tools and Technology Research</td>
<td>1.2.1. Incompatibility of software development tools</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.2.2. Learning software development tools takes so much time</td>
</tr>
<tr>
<td></td>
<td>1.3. Requirements specification</td>
<td>1.3.1. Business logic gaps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.3.2. Functional requirements not well defined or detailed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.3.3. Non-Functional requirements underestimated</td>
</tr>
<tr>
<td></td>
<td>1.4. Project Management</td>
<td>1.4.1. Underestimating timing in the time management plan</td>
</tr>
<tr>
<td><strong>2. Design of the system</strong></td>
<td>2.1. Logical Design</td>
<td>2.1.1. Changes in the architectural logic</td>
</tr>
<tr>
<td></td>
<td>2.2. Interface Design</td>
<td>2.2.1. Dismiss target market when interfaces are designed</td>
</tr>
<tr>
<td><strong>3. Implementation</strong></td>
<td>3.1. Logical implementation</td>
<td>3.1.1. Connection to Junar platforms unavailable</td>
</tr>
<tr>
<td></td>
<td>3.2. Interface implementation</td>
<td>3.2.1. Inadequate interface implementation because of development tools limitations</td>
</tr>
<tr>
<td><strong>4. Testing and Evaluation</strong></td>
<td>4.1. Unit testing</td>
<td>4.1.1. There is not enough data to test all possible cases</td>
</tr>
<tr>
<td></td>
<td>4.2. Integration testing</td>
<td>4.2.1. Incremental testing is not carried out.</td>
</tr>
<tr>
<td></td>
<td>4.3. Acceptance testing</td>
<td>4.3.1. Members of the target market have not tested the application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.3.2. Examiners have not tested the application before the final prototype generation</td>
</tr>
</tbody>
</table>

Table 2.8 Project risks
### 2.3.3.2. Analyzing Risks

The aim of this phase is to assign a probability and a severity to the risks. Probability refers to how frequently the risk can occur. It can take a value from 1 to 5 where 1 is the lowest which means that the risk can occur rarely. Severity mentions how dangerous can be this risk for the success of the project. A risk can take a value from 1 to 5. Higher is the value most dangerous are the consequences of the risk.

<table>
<thead>
<tr>
<th>Risk Code</th>
<th>Risk</th>
<th>Probability (1 low – 5 high)</th>
<th>Severity (1 low – 5 high)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.</td>
<td>Scope and restrictions not well-defined</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1.1.2.</td>
<td>Lack of experience to estimate the project phases</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1.1.3.</td>
<td>Application not delivered on time</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1.1.4.</td>
<td>High implementation cost</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1.</td>
<td>Incompatibility of software development tools</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2.</td>
<td>Learning software development tools takes so much time</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1.</td>
<td>Business logic gaps</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1.3.2.</td>
<td>Functional requirements not well defined or detailed</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1.3.3.</td>
<td>Non-Functional requirements underestimated</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1.4.1.</td>
<td>Underestimating timing in the time management plan</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1.</td>
<td>Changes in the architectural logic</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2.2.1.</td>
<td>Dismiss target market when interfaces are designed</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3.1.1.</td>
<td>Connection to Junar platforms unavailable</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3.2.1.</td>
<td>Inadequate interface implementation because of development tools limitations</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4.1.1.</td>
<td>There is not enough data to test all possible cases</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4.2.1.</td>
<td>Incremental testing is not carried out.</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4.3.1.</td>
<td>Members of the target market have not tested the application</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4.3.2.</td>
<td>Examiners have not tested the application before the final prototype generation</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2.9 Risk analysis
2.3.3.3. Preventing and managing risks

This section states some mitigation strategies for risks. They are detailed only for risks with probability and severity of three to five.

<table>
<thead>
<tr>
<th>Risk Code</th>
<th>Risk</th>
<th>Mitigation strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.</td>
<td>Scope and restrictions not well defined</td>
<td>Write two drafts of the project definition document which will be revised by an expert before the presentation of the final document.</td>
</tr>
<tr>
<td>1.1.2.</td>
<td>Lack of experience to estimate the project phases</td>
<td>Ask for suggestions to people with more experience in the area.</td>
</tr>
<tr>
<td>1.1.3.</td>
<td>Application not delivered on time</td>
<td>Evaluate progress of the project after each phase of the project cycle.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Evaluate the progress of the project three weeks before deadline delivery.</td>
</tr>
<tr>
<td>1.1.4.</td>
<td>High implementation cost</td>
<td>Use free development tools and technology.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use tools and technology offered by the university of Southampton technological service</td>
</tr>
<tr>
<td>1.2.1.</td>
<td>Incompatibility of software development tools</td>
<td>Test the software compatibility at early phases of the project, particularly in the project definition phase.</td>
</tr>
<tr>
<td>1.2.2.</td>
<td>Learning software development tools takes so much time</td>
<td>Learning software development tools since early phases of the project cycle.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select tools with references of easy learning and good usability.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Add extra learning hours to the plan.</td>
</tr>
<tr>
<td>1.3.1.</td>
<td>Business logic gaps</td>
<td>Research success projects that use similar business logic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ask for advice to experts to cover more aspects of business logic.</td>
</tr>
<tr>
<td>1.3.2.</td>
<td>Functional requirements not well defined or detailed</td>
<td>Include a requirement section into the documentation of the project, which specifies the functional features of the application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ask for revision of functional requirements to experts.</td>
</tr>
<tr>
<td>1.3.3.</td>
<td>Non-Functional requirements underestimated</td>
<td>The requirements must be based on metrics from official sites.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clarify that the metrics were estimated based on an specific execution environment.</td>
</tr>
</tbody>
</table>
1.4.1. Underestimating timing in the time management plan

First, we should create an initial plan. Then increase 10% of time to every activity. Finally, fit the plan to the limit of time stated (three months).

2.1.1. Changes in the architectural logic

Create an architecture that supports multiplatform connections

2.2.1. Dismiss target market when interfaces are designed

Create a testing plan that ensures to cover basic test cases in order to ensure the usability and accuracy of the software developed.

4.1.1. There is not enough data to test all possible cases

Create temporal data for test cases that are not covered with data from Junar platforms

4.2.1. Incremental testing is not carried out.

The testing plan must take into account that the software must be tested after every prototype release

4.3.2. Examiners have not tested the application before the final prototype generation

After the first prototype release, a schedule for testing the application must be proposed.

Table 2.10 Risk mitigation strategies

2.3.4. Time management

Time management plan describes a project of 14 weeks divided in seven sections. These sections can be grouped in three phases including starting, development and closing.

The starting phase is composed of first (scope definition and background), second (system analysis) and third (design) section of the diagram specified in figure 2.2. It takes six weeks and defines two milestones. The first milestone is to delivery documentation about the introductory information of the project. The second milestone is to build the system analysis and the design documents.

Implementation and testing sections are members of development phase. They and the evacuation section spends seven weeks where the first week is used to work in parallel with the last activities of phase one. Three milestones are used to delivery prototypes of the application. The first is after three weeks of work, and the others are in the last two weeks of this phase.

Finally, around two weeks are used to execute the closing phase. This phase defines the conclusions of the project. This project ends with a space to carry out corrections to documentation or the application developed. This phase states two milestones which includes a conclusion document and the final product as last milestone. The final product includes the application, and documentation according to the requirements defined in section 2.2.1.
<table>
<thead>
<tr>
<th>Week #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>week beginning:</td>
<td>3/6</td>
<td>10/6</td>
<td>17/6</td>
<td>24/7</td>
<td>1/7</td>
<td>8/7</td>
<td>15/7</td>
<td>22/7</td>
<td>29/7</td>
<td>5/8</td>
<td>12/8</td>
<td>19/8</td>
<td>26/8</td>
<td>2/9</td>
</tr>
<tr>
<td>1. Defining scope of the project and background topics</td>
<td>Milestone – Introduction documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. System analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Design</td>
<td>Milestone – System analysis and design documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Implementation</td>
<td>Milestone – present first application prototype</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Testing &amp; evaluation</td>
<td>Milestone – present second application prototype</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Milestone – demonstration of application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Conclusion</td>
<td>Milestone – Conclusion Documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Final corrections</td>
<td>Milestone – Delivery of product</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.2 Project Management Plan
3. Design

The aim of this section is to provide the design of the application which is used in the implementation phase, and it is based on functional requirements. Agile software development best practices cite to agile modeling as recommended methodology in the design phase of the development cycle [18]. It defines among its best practices that just barely good enough artifacts must be built and lookahead modeling must be used to control overall risk [19]. Based on these statements, the project presents artifacts like class, sequence and interface diagrams.

3.1. Logical Design

In this section is detailed the architectural definition of the application as well as the logic design used to implement the requirements stated in section 2.2.1.

3.1.1. System architecture

In order to develop ODLAIN, it was designed a communication architecture (see appendix C). This architecture indicates that the application is installed in the mobile device, and it connects to Junar Open Data Latin America server directly. There are not servers or any other applications between them. Rq label represents a request call from ODLAIN, and Rs a response from Junar server call in the above figure.

System analyses phase defined the indicators to be implemented. Junar Open Data Latin America provides data available to generate these indicators, but part of this data is not stored in this Junar platform. This data is stored in other Junar platforms. The new system architecture (see appendix C) is designed to access to these platforms. We are able to use the same connection logic because all are Junar platforms.

ODLAIN uses some tools to interact with the servers. For example:

- Junar Key. - Connection to Junnar platforms uses Junar keys. ODLAIN has a Junar key for every platform that it is going to interact with.
- Junar REST. - The application connects to Junar REST services provided by the Junar platforms.
- JSON. - Acknowledgment responses (Rs) return data in JSON format.
The above figure explains how the system interacts internally. Application communicates with users and servers. The former is through the Interface layer and the latter is through a connection controller. This figure represents the communication logic which shows the three layers of the Model view controller (MVC) design pattern. This is to provide independence between the layers. Besides, this project uses this strategy to let to reuse code, ease the maintenance of the application as well as simplify the migration of ODLAIN to other platforms [20,21]. The following sections describes the design of these layers.

3.1.2. Interaction Diagrams

3.1.2.1. Class Diagrams

As was cited above the application design was divided in three layers. The classes which are member of the com.soton.mauricio.odlain.logic package represent the model layer. The com.soton.mauricio.odlain.controller package is composed of the controller layer classes. The interface classes are part of the view layer. They are into the first level of the com.soton.mauricio.odlain package. The model layer is detailed in section 3.2; on the other hand, controller and view layers are described as follow.

a. Controller design
This design diagram is composed of nine classes where eight of them correspond to the controller layer. An additional class (CupboardSQLiteOpenHelper) is included to show how this layer interacts with the model layer (logic package in the application). More information about this interaction can be found in section 3.2.

Dao class allows building a connection to the model layer through the CupboardSQLiteOpenHelper class. All Dao classes that inherit from Dao class are allowed to access to the model layer. InitialConfiguration class is in charge of creating the configuration data including countries, indicators, Junar servers, and connection configurations. CountryDao class represents the structure of dao classes. The other dao classes have this structure although it is not reflected in figure 3.2. Controller class is in charge of creating the indicator values required recovering this data from Junar servers or the local data base.
Controller class details the structure to adapt the calculation of the indicators. It can be noted that it is defined a method for every indicator. It is not able to create a global method to manage all indicators calculation because the complexity of the calculation of indicators.

InitialConfigurtarion and controller classes are the interaction link between the view and controller layer.

b. View design

**Figure 3.3 Class Diagram – View Design**

The class diagram showed above includes classes stated to ODLAIN interface design. It is composed of five classes which are described as follow:

- **MainActivity**. - It is used to state the home interface functionality, including the country and indicator selector widgets as well as the navigation to IndicatorComparisonActivity and PieGraphActivity graph.

- **IndicatorComparisonActivity**. - It is designed to show the comparison table. Besides, it provides navigate to IndicatorDeatiledActivity and PieGraphActivity.

- **IndicatorDeatiledActivity**. –It is designed to display the indicator per year table. Besides, it provides navigate to BarGraphActivity.

- **BarGraphActivity.**- It is used to show a bar chart, which is a representation of the data provided by IndicatorDeatiledActivity.
- PieGraphActivity. It is used to show a pie chart, which is a representation of the data provided by IndicatorComparisonActivity.

All classes in this package uses to InitialConfigutarian and Controller classes in order to interact with the controller layer.

3.1.2.2. Sequence Diagrams

In this section is defined the information flow that the application must implement in order to solve the user demands. Sequence diagrams are used to describe the interaction logic of the system.

a. Retrieve indicators by country

Retrieve indicators by country calculates and return the values of an indicator for one or two countries. First, the user asks for catalogs of indicators and countries. Second, he selects the indicator, and one or two countries. Third, user requests to calculate an indicator. Fourth, the controller recovers data from Junar platforms, and calculates the indicator value for the countries selected. Finally, return the calculated values.

![Sequence Diagram](image)

Figure 3.4 Retrieve indicators by country - Sequence Diagram

b. Retrieve Indicator

It returns the values of an indicator for all countries of Latin America. First, the user asks for the indicator catalog. Second, the user requests to retrieve the indicator data. Third, the controller recovers data from Junar platforms, and calculates the indicator. Finally, return the calculated values.
c. Retrieve graph information

It returns the data needed to create a bar or a pie graph resource. The user sends the indicator id and data per year he recovered earlier. Graph indicator calculates values and returns it to the user.

3.2. Data Base Design

ODLAIN uses Cupboard tool to manage the persistence model. This section describes the model layer and the data base model used in this application. The design of this layer following the MCV pattern eases the database maintenance in future versions of ODLAIN [22].

3.2.1 Model design

The persistence layer represents the model part of MVC. The following figure represents the data layer designed for the application.
The above figure contains six classes. CupboardSQLiteOpenHelper class uses Cupboard library to create the database connection. The others are part of the persistence layer which can be detailed as follow:

**Indicator class.** It is the abstract representation of an indicator. It is composed of id, name, description and measure attributes which are stated as private. Metric is the unit used to provide meaning to indicators. It symbolizes the five indicators stated for the application.

- **QueryConnection class.** It is used to store the configuration of connections to Junar Servers in order to gathering data. QueryUrl is used to store the URL needed to invoke a RestFull service in Junar APIs.
- **Country class.** It is used to represent the countries of Latin America that the application uses.
- **OpenDataServer class.** It represents the Junar Open Data servers. ProjectName refers to the name of the Open Data platform, and key is the security key needed to connect to the server.
- **IndicatorDetail.class.** It is the value of an indicator for a country in a specific year.

### 3.3. Interface Design

#### 3.3.1. Home screen

Figure 3.8 specifies the design of the interface used to implement FR_APP_01, FR_APP_02, FR_APP_03, FR_APP_04, FR_APP_05, and FR_APP_06. This interface is the start default page to any information research.
3.3.2. Indicator screen

Indicator Comparison Screen (Figure 3.9) represents the design interface for the indicator comparison functionality. This design is used to implement FR_APP_08, FR_APP_13, FR_APP_18, FR_APP_23, FR_APP_28 which refers to indicator comparison.

Indicator historical information screen in figure 3.10 shows the design for detailed indicator information per country requirements cited in section 2.2.1. Requirements FR_APP_09, FR_APP_14, FR_APP_19, FR_APP_24, FR_APP_29 denote indicator information per country requirements.

In the case of figures 3.9 and 3.10, in the top there are four elements. Home option is used to redirect to Home Screen. In the central area is located a data grid which presents information depending of the kind of requirement the application is responding at that moment. At the bottom, it can be found two action elements. Show graph is to navigate to the chart screen, and Show information to display information about an indicator.

The screen is divided into three areas. In the top area there are three elements. “Languages” option implements the requirement NF_06. At the central area are located the option elements used to select indicators and countries. At the bottom, it can be found two action elements to navigate to indicator screen or map screen.

Figure 3.8 Home Screen
3.3.3. Chart screen

Figure 3.9 Indicator Comparison Screen

Figure 3.10 Historical Information Screen

Figure 3.11 Pie graph screen

Figure 3.12 Bar graph Screen
Interface designs in figures 3.11 and 3.12 represents an outline for the presentation of graph resources detailed in section 2.2.1. The pie graph screen in figure 3.11 is used to implement FR_APP_10, FR_APP_15, FR_APP_20, FR_APP_25 and FR_APP_30 which refers to bar graphs. The bar graph screen in figure 3.12 is the design to implement FR_APP_11, FR_APP_16, FR_APP_21, FR_APP_26, FR_APP_31 which specify the visualization of pie graphs.

In both figures, there are four elements in the top area which includes title, home, languages, and about. In the central area is located a panel which contains a chart diagram. It presents a graphical representation based on information which depends on the requirement requested.

3.3.4. Map screen

The following figure represents the design interface for the Latin America map indicator resource provided to implement FR_APP_06. The interface can be divided in two areas. At the top there are four elements which includes title, home, settings, and about. In the central area is located the map resource which presents information depending on the indicator request as is stated in FR_APP_01.

![Figure 3.10 Map Screen](image-url)
4. Implementation

This section will detail technical features of ODLAIN implementation as well as some design modifications that will be required to implement the system requirements defined in Section 2.2. Although in section 2.2.1 was stated that the application will present a comparison between two countries for the latest year, we believe it is useful to modify this requirement to implement this comparison for the years from 2009 to 2013. According to requirement analysis, the development phase will be split in five sections.

4.1 Implementation of initial configuration

(FR_APP_01, FR_APP_02, FR_APP_03, FR_APP_04) ODLAIN uses a SQLite database to store data (see Section 3.2). In order to populate the country, indicator, open data server, and query connection tables, it is developed an initial configuration. It allows to ODLAIN provides the elements to implement the requirements listed above.

4.1.1 Model layer Implementation

In this phase, the persistence classes stated in section 3.2 which include Country, Indicator, OpenDataServer, and QueryConnection are implemented as simple java classes. Besides, a class called CupboardSQLiteOpenHelper extends from SQLiteOpenHelper to uses cupboard library methods. This class is in charge of creating the data base structure in SQLite as well as providing the persistence layer to ODLAIN.

4.1.2 Controller layer implementation

In order to implement the requirements cited above, five dao classes are implemented including Dao, CountryDao, IndicatorDao, OpendataServerDao, and QueryConnectionDao. The dao class is in charge of creating an instance of CupboardSQLiteOpenHelper to allows to access to the persistence layer of ODLAIN. As was cited in figure 3.2, the remaining dao classes are subclasses of Dao in order to inherit access to the model layer.

The InitialConfiguration class is used to create countries, indicators, open data servers, and the query connections that ODLAIN needs. This class gathers the data from enumeration and stores it into the database using dao classes. Figure 4.1 details the structure of InitialConfiguration.
package com.soton.mauricio.odlain.controller;
import java.util.ArrayList;

public class InitialConfiguration {
    private List<Country> countryList = new ArrayList<Country>();
    private List<Indicator> indicatorList = new ArrayList<Indicator>();
    private List<OpenDataServer> openDataServeList = new ArrayList<OpenDataServer>();
    private List<QueryConnection> queryConncetionList = new ArrayList<QueryConnection>();
    Context context;

    public InitialConfiguration(Context context) {
    }

    public void createInitialConfiguration() {
    }

    public List<Country> getCountryList() {
    }

    public List<Indicator> getIndicatorList() {
    }

    public List<OpenDataServer> getOpenDataServeList() {
    }

    public List<QueryConnection> getQueryConnectionList() {
    }

    public void setCountryList(List<Country> countryList) {
    }

    public void setIndicatorList(List<Indicator> indicatorList) {
    }

    public void setOpenDataServeList(List<OpenDataServer> openDataServeList) {
    }

    public void setQueryConnectionList(List<QueryConnection> queryConncetionList) {
    }
}

Figure 4.1 InitialConfiguration class – Controller Layer

4.1.3 View layer implementation

The elements developed to implement the requirements cited in this section include an android activity (MainTabActivity), an android fragment (MainActivity) and two xml layout documents (activity_tab_main, activity_main). MainTabActivity instantiates an InitialConfiguration object which allows carrying out the initial data creation. MainActivity recovers the information to populate the user interface elements. Activity_tab_main contains MainActivity using a TabAdapter widget. This xml file also declares a TextView and Spinner widgets. Activity_main declares three TextView and three Spinner elements. The most of this interface corresponds to design specified in section 3.3.1. This implementation is in figure 4.2.
4.2 Implementation of indicators (Information per Year)

(FR_APP_05, FR_APP_07, FR_APP_08, FR_APP_12, FR_APP_13, FR_APP_17, FR_APP_18, FR_APP_22, FR_APP_23, FR_APP_27, FR_APP_28) ODLAIN shall provide indicators based on a year.

In order to collect the data required, this application must connect to different Junar REST APIs. The design considered that ODLAIN will connect to several Junar platforms, but it was not considered to connect to various Junar servers to calculate an indicator. The following sections explain the calculation of indicators to provide up-to-date information, and the modifications applied to the original design defined in the section 3.

4.2.1 Model layer Implementation

IndicatorDetail class is implemented in this section based on the original design (see figure 3.2). The table IndicatorDetail is created because the use of the CupboardSQLiteOpenHelper class. There are two modifications to the former design.

The IndicatorDetail class structure changed since the QueryConnection attribute was deleted. Besides, it is added a new class called QConnectionIDetail. It refines the design and lets calculate an indicator value after connect to several Junar APIs. Figure 4.3 shows the updated persistence diagram for this section
4.2.2 Controller layer implementation

It explains the changes in the design of section 3.1.2.1. It will help researchers to develop next versions of ODLAIN.

4.2.2.1 Data gathering

GenericController class is created to manage the connections that ODLAIN must execute in order to retrieve the data from Junar APIs. Originally this feature should have been implemented into the controller class, but we decided to create a second class in order to improve the modifiability of this application. GenericController will be in charge of managing the network connections and Controller in charge of manage the calculations using the data recovered from these connections.

This class extends from AsyncTask class. It means that is used to manage asynchronous calls. ODLAIN use them to recover data from Junar servers.

4.2.2.2 Indicator Calculation

Figure 4.4 controller classes – Controller Layer
This section implements a Controller class. The structure of this class remains to the original design. Nevertheless, additional classes are developed because of the complexity of the calculations to obtain the indicators. We created five classes which were named according to the name of the indicators. They are used to implement the calculation of the indicator for countries.

Figure 4.4 shows the class diagram for this section which is based on the controller view diagram of figure 3.2.

4.2.3 View layer implementation

This section resources respond to the requests and design stated in section 3.3.2 principally referring to figure 3.9. These resources include an activity (IndicatorComparisonTabActivity), a fragment (IndicatorComparisonActivity) and an xml layout document (activity_tab_indicator_comparison, activity_indicator_comparison). The former instantiate a Controller object in its onCreate method to calculate the indicator values requested. The fragment fills the widget elements stated in activity_indicator_comparison. The layout documents implements widgets according to the interface design.

![ODLAIN App Interface](image)

Figure 4.5 Information per year interface
FR_APP_05 states the communication between MainActivity and IndicatorComparisonActivity. We implement a button element in MainActivity which calls a method to solve this requirement. This method instantiates a GenericController instance to gathering data and sent it to IndicatorComparisonActivity. ODLAIN send a json string between MainActivity and IndicatorComparisonActivity. This json string is a representation of a ConnectionResult object. This class contains information to fill the user interface elements. The result of this section implementation is in figure 4.5

4.3 Implementation of indicators (Historical Information)

(FR_APP_05, FR_APP_07, FR_APP_09, FR_APP_12, FR_APP_14, FR_APP_17, FR_APP_19, FR_APP_22, FR_APP_24, FR_APP_27, FR_APP_29) This section shall explain how ODLAIN generates historical information. It also details the changes in the design made to implement the requirements of this phase.

4.3.1 Model layer Implementation

It was realized two modifications to the persistence design of figure 3.2. These changes update the QueryConnection persistence class. First, it adds the attribute year. This is to determinate the year which this connection refers. This change is useful to implement section 4.2. Second, it adds the attribute selections. This is to specify the relevant columns used to calculate the indicators.

4.3.2 Controller layer implementation

Some functionalities of ODLAIN are complex, which becomes difficult reuse their code fragments. We can cite that sometimes the best option is to skip some patterns in order to ease the implementation [23]. For this reason, we modify the design in section 3.1.2 as follow.

4.3.2.1 Data gathering

The controller class differentiates when ODLAIN needs to retrieve connections for a year or a sort of them. The controller invokes methods of the QueryConnectionDao class to list these connections.

The selections attribute is to filter the information retrieved from Junar API’s. Junar APIs allows filtering data by rows but not by columns. It means that all the columns of a table are gathered. It reduces the efficiency of ODLAIN. In order to reduce the traffic of information in asynchronous calls, the method reduceResultString uses the selections attribute. This method filters the columns of the result in order to deliver only the relevant data through the asynchronous calls.
4.3.2.2 Indicator Calculation

The five classes developed in section 4.2.2.2 are used to implement the calculation of historical information. The methods with format calculate[Country Name]HistoricalValue implemented in these classes are used to implement the calculation of these requirements. The following figure details the structure of a class modified to calculate historical information.

```java
package com.soton.mauricio.odlain.controller.indicator;

import java.text.DecimalFormat;

public class AverageSalary extends IndicatorController {
    private static float workingHoursChile = 180;
    private static float workingHoursBolivia = 180;

    public AverageSalary(ConnectionResult connectionResult, Context context) {
        // Constructor

    }

    public ConnectionResult selectCountryValue() {
        // Method to select country value

    }

    private void calculateUruguayValue() {
        // Method to calculate Uruguay Value

    }

    private String calculateUruguayHistoricalValue() {
        // Method to calculate Uruguay Historical Value

    }

    private void calculatePeruValue() {
        // Method to calculate Peru Value

    }

    private String calculatePeruHistoricalValue() {
        // Method to calculate Peru Historical Value

    }

    private void calculateCostaRicaValue() {
        // Method to calculate Costa Rica Value

    }

    private String calculateCostaRicaHistoricalValue() {
        // Method to calculate Costa Rica Historical Value

    }

    private void calculateBrazilValue() {
        // Method to calculate Brazil Value

    }

    private String calculateBrazilHistoricalValue() {
        // Method to calculate Brazil Historical Value

    }

    private void calculateArgentinaValue() {
        // Method to calculate Argentina Value

    }

    private String calculateArgentinaHistoricalValue() {
        // Method to calculate Argentina Historical Value

    }

    private void calculateBoliviaValue() {
        // Method to calculate Bolivia Value

    }

    private String calculateBoliviaHistoricalValue() {
        // Method to calculate Bolivia Historical Value

    }

    private void calculateChileValue() {
        // Method to calculate Chile Value

    }

    private String calculateChileHistoricalValue() {
        // Method to calculate Chile Historical Value

    }
}
```

Figure 4.6 AverageSalary class – Controller Layer

4.3.3 View layer implementation

This section implements the design of the interface stated in figure 3.10. The artifacts implemented in this section are two xml layout documents (activity_tab_indicator_detail, activity_indicator_detail), a fragment IndicatorDetailActivity and an activity IndicatorDetailTabActivity.

Activity_indicator_detail defines the user interface elements including texView elements and a tableLayout. The table layout is filled with information from the controller class which is instantiated in the IndicatorDetailActivity class. The user navigates from IndicatorComparisonActivity fragment to IndicatorDetailTabActivity through activating an event declared in IndicatorComparisonTabActivity. The method instantiates GenericController which will recover data. It will send data to IndicatorDetailedTabActivity in order to fill the elements of this activity. The result of this implementation is represented in figure 4.7.
4.4 Implementation of chart diagrams

(FR_APP_10, FR_APP_11, FR_APP_15, FR_APP_16, FR_APP_20, FR_APP_21, FR_APP_25, FR_APP_26, FR_APP_30, FR_APP_31) This section details the implementation of charts in ODLAIN. It shall detail changes applied to design stated in sections 3.1.2 and 3.3.3.

4.4.1 View layer implementation

A modification to ODLAIN’s interface design was implement tab adapters to improve the interface navigability for users. Android provides this type of resources for activities of the application but not for fragments.

In this section are implemented two android fragments and two xml layout files. Figure 4.16 stipulates the structure of the updated view design. These classes remain attributes and methods stated in figure 3.3. Although in the requirements was stated to create a pie chart to compare two countries and a bar chart to show the evolution of historical information, after implement these artifacts we realize there is other types of chart diagrams to present the information. For this reason ColumnGraphActivity is implemented to compare an indicator for two countries, and LineGraphActivity fragment to display historical information.
The fragments contain charts based on the use of WebView widgets. Google charts API allows to render a graph based on html contents which are displayed in the WebView elements. Figure 4.9 and 4.10 shows the result of this implementation.

Figure 4.9 Column chart interface

Figure 4.10 Line chart interface
4.5 Implementation of the map of Latin America

(FR_APP_06) This section implements a map of Latin America. It also details the design modifications as well as technologies used to develop this artefact.

4.5.1 Model layer Implementation

The `Country` class of the model layer is modified in this section. We add two new attributes to this class. They are `lat` which refers to latitude and `lng` to longitude. They will store the location of a country.

4.5.2 Controller layer implementation

The `CountryDao` class reflects the changes in the `Country` class. The `QueryConnectionDao` class implements the method `retrieveByIndicatorByYear`. This method recovers all connections for all countries in a specific year. These are the relevant changes made to the design stated in figure 3.2.

4.5.3 View layer implementation

This section builds a fragment (`MapActivity`) and an XML layout document (`activity_map`). Google Maps Android API v2 is used in ODLAIN as a library which provides the elements to display interactive maps. `MapActivity` uses the widget `MapView` to display the map of Latin America. The following figure shows this implementation. Appendix D outlines the process to enable this map in ODAIN which is based on the information of the google maps API v2 official web site [24].

![Map interface – View layer](image)

Figure 4.11 Map interface – View layer
5. Testing

This section shall provide information about the testing carried out which support the quality of this software product. This phase will use the box approach particularly white-box. The testing levels covered are integration and system testing. The Android testing framework is the tool used to create and execute test cases. These test cases are tested in a mobile device which runs an Android 4.3 OS and an Android virtual device (AVD) which runs an Android 4.4 OS.

5.1 Integration testing

The following table is a summary of the tests executed. In this table, the column code assigns a code to the test. The classes column specifies the classes used in the execution. Description is an outline of the case. Requirement list the codes of the requirements related to the test. Reference provides the name of the appendix which contains the execution of the test.

<table>
<thead>
<tr>
<th>Code</th>
<th>Classes</th>
<th>Description</th>
<th>Requirement</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT_01</td>
<td>MainTabActivity, MainActivity, InitialConfiguration</td>
<td>This case is to test the selection of years, indicators, and countries as well as their use in MainActivity fragment.</td>
<td>FR_APP_01, FR_APP_02, FR_APP_03, FR_APP_04</td>
<td>Appendix E</td>
</tr>
<tr>
<td>IT_02</td>
<td>Controller, ComparisonIndicator, ComparisonDetailTabActivity, ComparisonDetailActivity</td>
<td>This case is to test the gathering of data from a Junar API.</td>
<td>FR_APP_02, FR_APP_03, FR_APP_05, FR_APP_09</td>
<td>Appendix F</td>
</tr>
<tr>
<td>IT_03</td>
<td>Controller, ComparisonDetailTabActivity, ComparisonDetailActivity</td>
<td>This case is to test the gathering of data from multiple Junar API's.</td>
<td>FR_APP_02, FR_APP_03, FR_APP_05, FR_APP_09</td>
<td>Appendix G</td>
</tr>
</tbody>
</table>

Table 5.1 Integration testing cases

The response time to user requests was 8 seconds (the slowest case) in the early start of this phase because of the multiple connections to Junar servers. Nevertheless, the implementation of the method reduceResultString into the GenericController class reduced this time to 5.8 seconds.
5.2 System Testing

This section describes the system testing executed in order to check the functionality of ODLAIN. The following table has the same structure that section 5.1 table.

<table>
<thead>
<tr>
<th>Code</th>
<th>Classes</th>
<th>Description</th>
<th>Requirement</th>
<th>Reference</th>
</tr>
</thead>
</table>
| ST_01 | MainTabActivity  
MainActivity  
ComparisonIndicator  
- TabActivity  
ComparisonIndicator  
- Activity | This case is to test the process to compare the indicator values for two countries in a specific year. | FR_APP_05  
FR_APP_08  
FR_APP_13  
FR_APP_18  
FR_APP_23  
FR_APP_28 | Appendix H |
| ST_02 | ComparisonIndicatorTab  
Activity  
ComparisonIndicator  
- Activity  
ComparisonDetailTab  
Activity  
ComparisonDetailActivity | This case is to test the process to display the indicator values for a country from 2009 to 2013. | FR_APP_05  
FR_APP_09  
FR_APP_14  
FR_APP_19  
FR_APP_24  
FR_APP_29 | Appendix I |
| ST_03 | MainTabActivity  
MapActivity | This case is to test the process to display the indicator values for all countries in a specific year. | FR_APP_01  
FR_APP_06 | Appendix J |

Table 5.1 System testing cases
6 Discussion

6.1 Evaluation

This section details the results of developing ODLAIN and using it as an application that provides indicators. In order to achieve this, we asked to a group of students use ODLAIN to complete four tasks. When they completed the tasks, they were asked to mark four features of the application, and answer three questions. This evaluation is composed of two sub-sections. The former is to present the results of the execution, and the latter is to state an analysis of these results.

6.1.1 Results

In order to introduce the results we divided them into three components. The first is related to the execution of the four tasks, the second and third are about the answers from users.

The first component is to display the results of the use of ODLAIN. The tasks executed in the first component of the evaluation are:

- Task 1 asked to find the public investment amount for Argentina and Chile in 2011.
- Task 2 asked to find the average salary in Bolivia on 2010.
- Task 4 asked to find all countries which have information about the notifiable diseases indicator on 2012.

There are four graphics that show the results of this part, these graphs are in appendix K. The tasks provided information to evaluate the following three variables:

- Correctness of the result. It lets to know if the result of the indicator was the expected value.
- Time used to execute the task. It evaluates if the time is into the range of time permitted.
- Correctness of the use of ODLAIN. It lets to know if the user used the ODLAN feature that we were expecting the user was going to use.

The second component is to evaluate the user interface based on four characteristics. Users marked the following features of ODLAIN.

- Feature 1. Interface color combination.
- Feature 2. Icons size.
- Feature 3. Characters readability.
Feature 4. Navigability between screens

In all cases, the request was to mark the feature with a number from one to five where one represents inadequate and five adequate. The following graph displays the result of the marking.

![User interface evaluation results graph]

Figure 6.1 User interface evaluation results

The third component is to assess the user satisfaction. We asked the users to express their opinion of this application requesting the following three judgments.

- Judgment 1. The most negative aspects of ODLAIN.
- Judgment 2. The most positive aspects of ODLAIN.
- Judgment 3. Features for the next version of ODLAIN.

6.1.2 Analysis

The porpoise of this section is analyses the results presented in section 6.1.1. In the first component of the results the user executed four tasks where every task evaluated a feature of ODLAIN. The purpose of these tasks is to assess the functionality of the application and register information about not functional features. In order to prove the non-functional requirement NF_02, users used the application for one minute before communicate them which tasks they had to execute.
The first parameter is used to evaluate the correctness of the application. The results show that ODLAIN often returns the right value when users execute a query. It shows that this application offers five indicators for six countries based on data collected from Junar servers. This evaluation ensures that ODLAIN returns the expected value, but it does not mean that this value is officially accurate. ODLAIN accuracy depends on the accuracy of the data provided for the open data servers.

A second parameter assesses the task execution time. This evaluation took as base time seven seconds. These seven seconds include the time the user spent using the software and the response time of the application. The results show that around 90% of the users spent less than seven seconds to execute a task. If we consider the refinement explained in section 5.1 and appendix G, we can say that ODLAIN offers a reasonable execution time. The most complex use of the software lets users expend less than 7 seconds. This time is acceptable since it was increased because of the time spent in gathering data from servers but not in processing data inside ODLAIN. This result shows that develop an android application is not as complex because of the features that android provides to interact with other architectures. In our case, we talk about open data projects that use REST servers. However, researchers should investigate the standards that open data servers use to publish data. This is because the standards can increase or decrease the complexity of the development in android.

The third parameter is usability correctness. There are different ways to recover information using ODLAIN. This parameter evaluates the accuracy at the moment of using ODLAIN. The results shows that task one and two were executed using the feature that we expected they were going to use. In the case of tasks three and four, 10% of the users executed the task using a feature different to the expected. Nevertheless, they obtained the correct result. Based on these results, we support the following characteristics of this tool. ODLAIN provides to users a bunch of alternatives to find information which make it easy to use as well fast and trustable

The second component is used to evaluate the interface of ODLAIN. User interfaces are important in mobile applications. Some applications do not take into consideration the opinion of the final. Some project teams ask people related to software development to evaluate the final product. It is more difficult that developers take the role of users to find errors, so use them is a bad option. For this reason, we selected people which are not related to software development to evaluate ODLAIN. In order to analyze the results of this component, the values five and four are stated as acceptable. Features one and three have gotten an acceptance of over 90%. Features two and four have obtained an acceptance of over 80%. It means that ODLAIN acceptance is around 85%. Besides, the second judgment (The most positive aspects of ODLAIN) of the third component provides information to support these results. It allows us to list the following features of our tool.
• It offers a modern interface based on fresh colors.
• ODLAIN’s information is easy to read.
• Its interface is intuitive and easy to navigate based on icons and help information.

The fourth feature of the second component is about navigability between screens. Although the result of this feature is acceptable (80% acceptance), it is the lowest score in the section. It means that ODLAIN could improve this characteristic. It is useful to consider this result in subsequent versions of this software application.

The judgments presented in the third component of this evaluation helped us to understand the results of the first and second components. Furthermore, the opinions stated by users in judgments one and three lets to set the content of chapter seven.

6.2 Process execution

This section is an evaluation of the process carried out to implement ODLAIN. It includes highlighting the hits and problems figured out during this process as well as the techniques used to offer solutions. It analyzes the four phases of the development and concludes listing the refinements executed to ODLAIN.

First, the requirement section was pivotal for this project. In this section we identified the limitations of OpenData Latin America project. In order to solve the lack of relevant data for indicators, we expanded the scope of the project. The research found other Junar platforms managed by official government institutions which provided additional data for ODLAIN.

Second, the architecture of ODLAIN stated multi-connections to servers. This feature was implemented because of changes in the scope of the project. ODLAIN structure became stronger because of it. This was a problem in the development process, but the advantages of this change made it endurable. The new architecture makes to ODLAIN flexible to the generation of new indicator. This was not declared as feature in the project definition.

Third, the implementation phase was delayed because of many reasons including design changes, new features, and external factors. The implementation of a design for multiple connections and multiple servers increased the complexity of the application. On the second prototype presentation, a new requirement was introduced. This requirement lets to ODLAIN offers a comparison between the two countries for a specific year. The lack of standardization on the data generated by Junar servers also raises the complexity of the implementation. Although Junar states a format to upload data to their servers, data management is the responsibility of the country’s governments. The notion of centralizing data in Latin
America is interesting, but the project should improve two areas which include the generation of information for users and the development of refined APIs for developers. We implemented the map of Latin America because we realized it is something attractive for the user since the other applications we have researched do not offer integration with google map API.

Moreover, the test section tried to solve the non-functional requirement NF_03. The response time is still over the value stated in the requirement. According to the risk analysis in section 2.3.3, we can cite two risks related to this case which includes code risks 1.3.3 and 2.1.1. The mitigation strategies planned for these cases have been executed. Besides, it was implemented the filter method detailed above, but it was not enough to achieve the requirement stated in section 2.2.2. Nevertheless, the results of the evaluation in section 6.1 show that the response time is acceptable.

In order to improve the quality of ODLAIN, the following refinements have been implemented. First, the implementation of comparison of countries in a year. It lets to compare the value of the two countries in the year requested by the user. Initially, it was requested comparison only for the year 2013. This was figured out in after the first prototype presentation. The implementation of this new requirement is detailed in the section 4. Besides, the implementation of the filtering method to improve the response time. This requirement was figured out in the testing phase, and it is described in section 5.1. Finally, the size of icons, font color, and the grammar of titles changed in order to improve the user interface. This is based on the result of the evaluation specified in section 6.1 where users listed the most negative aspects of using ODLAIN.

### 6.3 Tools selection

We analyses the use of two tools in this section, the use of android development framework (ADT), including all its framework extensions and the use of the Junar APIs. It is because of the importance of these tools in this project. The former contains all the software tools that we used to develop ODLAIN. The latter is the platform that generates the REST services for the open data projects that ODLAIN uses. These tools are the basis of our software and for this reason we must evaluate them.

As we have cited early, Android provides a number of applications to develop software. This is an advantage since we have many options to develop something, but it is also a disadvantage when a developer has not enough experience with the android environment. Android is popular and for this reason its tools evolve rapidly. During the development of ODLAIN, Android development tools published two versions where the second was a fixed version of the first because of bugs. This delayed the development of ODLAIN two days. When this project started, we selected google image charts to
implement the chart graphs, and the map of Latin America. Nevertheless, after implement these artifacts, we realized that the final result was not the expected. Developing mobile applications differs from web applications since the user interface must be very attractive as well as interactive. For this reason we researched new tools. We selected google charts because it lets to create the visual effects that we were looking for ODLAIN. Besides, we selected google maps android API v2 which let us create a map which is even better than the map implemented in Eurostat mobile applications. We also used the android testing framework in the testing phase of this project. This tool is useful for people who are accustomed to use JUnit or Java implementations. This tool was useful to improve the quality of the application. It must be noted that the time spent in coding test cases is recovered since the number of iteration that can be executed over these test cases which ensure the correctness of applications.

Furthermore, we also have to evaluate the use of open data APIs. The Latin American APIs are new and for this reason are still not mature. Even though they are a good start for the area, they are forced to evolve quickly because of the popularity they can have in a future. These are some features that Junar APIs should implement in future versions.

- Allows to the client to choose the columns that he needs to recover in order to reduce network data traffic.
- Include more information about advanced filtering in the official web site.
- Add more filtering options particularly for the management of sequences of characters.

We have listed these features based on our experience using these APIs to develop the indicators.

6.4 Management Plan
The original management plan changed in their structure, but it does not have a negative effect on the final milestone dates. The reasons for the changes in the time estimation are detailed in previous sections of this chapter. The scope definition phase took less time that it was expected. For this reason the extra week consumed for implementation did not delay the final product delivery date. The analysis in sections 6.2 and 6.3 implies that the most challenging milestones of this project have been the calculation of indicators and the implementation of the map of Latin America. The first refers to the business logic, and it is because of the complexity to generate indicators. The second was a technical challenge since the process to implement the map was complex. It must be clarified that the study of the software tools was carried out during all the process due to it was a mitigation risk strategy stated in the risk plan. It was useful to ease the implementation of the map of Latin America, which required a considerable amount of time to research the use of google maps API v2.
7 Conclusions and future work

7.1 Conclusions

This section includes the contributions that this project provides in terms of academic research. The goal of this project was to create a mobile android application based on the idea of Eurostat indicators. The following statements are improvements that ODLAIN makes to the Eurostat mobile application.

- This application includes historical information.
- The use of the google maps API allows to ODLAIN offers a more interactive map.
- It uses google charts. This provides a more attractive presentation.
- The interface is implemented for two languages (English and Spanish).

Furthermore, we can list the following statements based on the information collected in section 6.1, and the remaining analyses of chapter 6.

- The Open Data Latin America project is an early project to centralize data in the region which shows a dizzy progress after a year of his creation.
- The content of this report let us affirm that there is a big gap between Europe and Latin America talking about the introduction of the open data initiative. This includes to international organizations stablished in Latin America.
- Our analysis allows to say that Chile is the Latin American country that most has invested to promote the open data initiative, particularly hiring the use of Junar platforms.
- This document stated some suggestions in order to reduce the complexity of the development of mobile applications based on open data projects in Latin America.
- ODLAIN demonstrates the importance of storing data based on standards and regarding the open data principles.
- ODLAIN has evidenced the compatibility between different android tools such as google maps API, google charts, android development tools and android testing framework.
- This project provides software which offers information for people interested in the evolution of Latin America.
- ODLAIN can be used as the basis for the development of a more complex application which can include the features cited in section 6.4.
Android tools evolve quickly because of their popularity. Therefore, developers must implement software able to evolve at the same pace that the tools that they used.

7.2 Future work

This section cites some ideas for future versions of ODLAIN based on our final product, implementation experience and the suggestions of users recorded in section 6.1.1.

- Improve the accuracy of currency conversion. ODLAIN converts from national currency to American dollars using parameters. These parameters are static. In order to improve this accuracy, we suggest that ODLAIN should connect to open data platforms and gathers the current conversion rate.

- Increase the list of indicators and countries. In order to implement this requirement we suggest considering two aspects. First, the number of indicators depends on the data provided by the Junar platforms. Second, if the number of indicators and counties increase is recommended to implement the calculation of indicators in a server. The reasons to suggest this includes. The lack standardization of the data provided for Junar platforms could increase more the complexity of the calculation of the indicators. The new indicators could increase the necessity of updating the parameters that ODLAIN uses.

- Implement the search for indicators by country. It is to list all the indicators available for a country.

- At the moment, ODLAIN offers the option to compare the value of an indicator between the two countries. The next version could offer a comparison for three countries or more.

- Implement comparison of multiple indicators for multiple countries.

- Currently, ODLAIN interface is translated in two languages (English and Spanish). Portuguese and French translations are required because they are the remaining languages spoken in the region.

- ODLAIN data is in English. It is required to translate the elements of the database to other languages.

- Improve support for multiple screen sizes.

- Implement support for iOS and windows phone.

- Implement the forecast of indicator values.
- Calculate indicators per region clustering Latin America in four regions which includes the Caribbean, Central America, Merconorte and Mercosur.

- Add new types of charts, for example, pie, bars, cylinder, pyramid, and surface charts.

- Allow users to choose the type of chart they want to use to interpret the information about indicators.

- Allow users to publish the results of indicator queries (tables and charts) in social networks.

- Allow users to download query results of indicators (tables and charts) in different formats, for example, csv, html, doc, xml, and pdf.

- Add a module to save queries which allows users save queries that they execute frequently.

- Include information on the reliability of the information provided. Display to users the reliability of an indicator value based on the reliability of the datasets used to calculate the indicator.
7. References


Appendix

A. Open Data Latin America datasets

The following table contains datasets stored in the Open Data Latin America project. It shows two datasets for Argentina, two for Chile and one for Peru. The column field of the table specifies the names of the field of a dataset. The data example column gives an example of the data stored in the repository.

<table>
<thead>
<tr>
<th>Country</th>
<th>Dataset Name</th>
<th>Field</th>
<th>Data Example (Spanish)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Argentina</strong></td>
<td>Expenses in White Bay</td>
<td>Date</td>
<td>2013-03-04T00:00:00-03:00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Detail</td>
<td>LIQUIDACION S/RENDICION ADJ.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amount</td>
<td>1772.25</td>
</tr>
<tr>
<td></td>
<td>Salaries in White Bay</td>
<td>Address</td>
<td>DIRECCION OFICINA MUNICIPAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Job title</td>
<td>TECNICO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Working time</td>
<td>48 HS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Salary before discounts</td>
<td>11461</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discounts</td>
<td>5125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Additional salary</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Final Salary</td>
<td>6336</td>
</tr>
<tr>
<td><strong>Chile</strong></td>
<td>Housing ministry budget</td>
<td>Description</td>
<td>APORTE FISCAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Money in</td>
<td>1,575,944,064</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transfers</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total amount</td>
<td>1,575,944,064</td>
</tr>
<tr>
<td></td>
<td>Notifiable diseases 2010</td>
<td>Typhoid and paratyphoid</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meningococcal Disease (Bacterial)</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tuberculosis</td>
<td>2,496</td>
</tr>
<tr>
<td><strong>Peru</strong></td>
<td>Public investment projects 2012</td>
<td>Project name</td>
<td>Mejoramiento de la capacidad resolutiva del servicio de neurocirugía</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Location</td>
<td>Hospital nacional dos de mayo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amount</td>
<td>9,951,775</td>
</tr>
</tbody>
</table>

Table B.1 Open Data Latin America datasets

The above table summarizes the relevant datasets for this project. In some cases, not all fields have been considered. Although the data of table B.1 remains in their original language, field labels and table titles have been translated into English. This is to comprehend the datasets published. Table B.2 is the English version of Table B.1.
<table>
<thead>
<tr>
<th>Country</th>
<th>Dataset Name</th>
<th>Field</th>
<th>Data Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina</td>
<td>Expenses in White Bay</td>
<td>Date</td>
<td>2013-03-04T00:00:00-03:00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Detail</td>
<td>Clearance/Accountability.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amount</td>
<td>1772.25</td>
</tr>
<tr>
<td></td>
<td>Salaries in White Bay</td>
<td>Address</td>
<td>Municipal management office</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Job title</td>
<td>Technician</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Working time</td>
<td>48 hours</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Salary before discounts</td>
<td>11461</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discounts</td>
<td>5125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Additional salary</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Final Salary</td>
<td>6336</td>
</tr>
<tr>
<td>Chile</td>
<td>Housing ministry budget</td>
<td>Description</td>
<td>Government contributions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Money in</td>
<td>1,575,944,064</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transfers</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total amount</td>
<td>1,575,944,064</td>
</tr>
<tr>
<td></td>
<td>Notifiable diseases 2010</td>
<td>Typhoid and paratyphoid</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meningococcal Disease (Bacterial)</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tuberculosis</td>
<td>2.496</td>
</tr>
<tr>
<td>Peru</td>
<td>Public investment projects 2012</td>
<td>Project name</td>
<td>Improvement of the performance of the neurosurgery service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Location</td>
<td>“Dos de mayo” national hospital</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amount</td>
<td>9,951,775</td>
</tr>
</tbody>
</table>

Table B.2 Open Data Latin America datasets (English version)
B. Junar Latin America platforms

The following table contains some examples of datasets from other Junar platforms. The country column refers to the country owner of the resource. The type of resource column indicates if the resource is a data table or a Junar view. The name of the resource column is the name of the table or view. The source of the information column is the name of the organization that provides the resource. The update date column specifies when the last time the data was updated.

<table>
<thead>
<tr>
<th>Country</th>
<th>Type of resource</th>
<th>Name of Resource (Spanish)</th>
<th>Source of the information (Spanish)</th>
<th>Update date (Spanish)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peru</td>
<td>View</td>
<td>Administración Tributaria</td>
<td>Administración Tributaria</td>
<td>4 julio, 2014</td>
</tr>
<tr>
<td></td>
<td>Table</td>
<td>Banco de Proyectos de Inversión 2014</td>
<td>Proyectos de Inversión</td>
<td>26 marzo, 2014</td>
</tr>
<tr>
<td>Argentina</td>
<td>View</td>
<td>Accidentes de Tránsito</td>
<td>Respuesta al Vecino</td>
<td>2 julio, 2014</td>
</tr>
<tr>
<td>Chile</td>
<td>Table</td>
<td>Farmacias de Chile en Línea</td>
<td>Salud</td>
<td>14 abril, 2014</td>
</tr>
</tbody>
</table>

Table C.1 Junar Latin America resources

Although the data of the above table remains in their original language, the table titles and the data of the country and type of resource columns are in English. Table C.2 is the English version of Table C.1.

<table>
<thead>
<tr>
<th>Country</th>
<th>Type of resource</th>
<th>Name of Resource (Spanish)</th>
<th>Source of the information (Spanish)</th>
<th>Update date (Spanish)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peru</td>
<td>View</td>
<td>Tax administration</td>
<td>Tax administration</td>
<td>4th July 2014</td>
</tr>
<tr>
<td></td>
<td>Table</td>
<td>Summary of investment projects on 2014</td>
<td>Investment projects</td>
<td>26th March 2014</td>
</tr>
<tr>
<td>Argentina</td>
<td>View</td>
<td>Road accidents</td>
<td>Replies to the neighbor</td>
<td>2nd July 2014</td>
</tr>
<tr>
<td>Chile</td>
<td>Table</td>
<td>Available pharmacies in Chile</td>
<td>Health service</td>
<td>14th April 2014</td>
</tr>
</tbody>
</table>

Table C.2 Junar Latin America resources (English version)
C. System communication architectures

We defined two communication architectures. The former was the one server communication architecture specified in figure B.1. It is composed for the Junar Latin America Rest server and instances of ODLAIN installed in mobile devices. The latter is the final ODLAIN architecture. It is a multiple server communication architecture, which is specified in figure B.2.

Figure B.1 Communication architecture
One server

Figure B.2 Communication architecture
Multiple servers
D. ODLAIN map implementation

The implementation of the map is composed of two parts. The first is to configure the application to allow users to display the map in their mobile devices. The second is to code the map using the documentation provided by the Google Maps API official site [24].

The configuration of the application is based on the guidelines of the official web site [25]. We suggest paid attention in the following steps of the configuration.

- In the step “Display your app's certificate information”, it is recommended to generate a new certificate for the final version of the application.
- In the step “Specify app settings in the application manifest”, we only specify the following permissions in the manifest file.

  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

In addition to the guidelines in [24], we need to set the latitude and longitude of a country in order to state the location of a country. This is to allow ODLAIN to set the markers in the map. The following figure shows the Country class after upgrade.

![Figure 4.19 Country class – Model layer](image-url)
E. Integration Testing: IT_01

This is to test the charge of years, indicators, and countries in MainActivity fragment which is into MainTabActivity activity. MainActivityTest is the name of the test class created and it extends from ActivityInstrumentationTestCase2.

Inputs
As inputs for this test are cited:

- Request the use of the year, indicator, first country, and second country spinners for this test case.

This section is configured in the setup method of MainActivityTest. Besides this method starts the activity and instantiates the InitialConfiguration class in order to create and charge the spinner widgets with data from the database.

Process
The test case includes five methods that are detailed as follow.

- TestSpinnerYearUI method. - It requests the focus of year spinner, and it states to 2013 as initial selection. As a next step, it selects the year of index 1 from the spinner. At the end, it declares an assert clause to check if the achieved year is 2012.

- TestIndicatorUI method. - It requests the focus of indicator spinner, and it states to average salary as initial selection. As a next step, it selects the indicator of index 3 from the spinner. At the end, it declares an assert clause to check if the achieved indicator is the public investment indicator.

- TestFirstCountryUI method. - It requests the focus of the first country spinner, and it states to Argentina as initial selection. As a next step, it selects the country of index 0 from the spinner. At the end, it declares an assert clause to check if the achieved country is Argentina.

- TestSecondCountryUI method. - It requests the focus of the second country spinner, and it states to Argentina as initial selection. As a next step, it selects the country of index 3 from the spinner. At the end, it declares an assert clause to check if the achieved country is Chile.

- TestMainActivityUI method. - It is in charge of managing the sequence to call to the other methods described in this process.

Outputs
The test case was executed successfully which implies that the following features work properly.

- Aggregation of indicators in the indicator table, and countries in the country table.
- Query elements of the indicator and country tables.
- Loading data into year, indicator, first country, and second country spinners.
- Selection and manipulation of year, indicator, first country, and second country spinners.
F. Integration Testing: IT_02

This test case is to verify the connection of ODLAIN to a Junar API using its Control class. ControllerFromMainTest is the name of the test class created and it extends from ActivityInstrumentationTestCase2.

Inputs
As inputs for this test are cited:

- Instantiate a Controller class which has to MainTabActivity as activity base.
- Instantiate a GenericController class which has to MainTabActivity as activity base.
- Instantiate an activity monitor which has to IndicatorComparisonTabActivity as target activity.

This section is configured in the setup method of ControllerFromMainTest.

Process
The test case includes three methods that are detailed as follow.

- TestRetrieveQueryConnectionOneYear. - It requests the query connections that Argentina has for the indicator average salary on 2013. It checks that the result is not null.
- TestCallOneServer. - It calls to TestRetrieveQueryConnectionOneYear and executes an asynchronous call to test the connection to Junar server and retrieve its data. It uses the activity monitor to catch the result since IndicatorComparisonTabActivity. It checks that the result is not null.
- TestCalculateIndicatorOneServer. – It calls to TestCallOneServer and calculates the indicator using the Control class. It checks that the ConnectionResult list is not null. It checks that the ConnectionResult list size is equal to one. It checks that the ConnectionResult is not null. It checks that the QueryConnection list is not null and has a size of one.

Outputs
The test case was executed successfully which implies that the following features work properly.

- Query of elements in QueryConnection table.
- Use of the retrieveQueryConnections method of the Control class.
- Use of GenericController class and test of connection to a Junar server.
- Retrieve of data from a Junar server and manipulation of this data.
- Communication between activities.
- Calculation of the average salary indicator for Argentina on 2013.
G. Integration Testing: IT_03

This test case is to verify the connection of ODLAIN to Junar APIs using its Control class. ControllerFromComparisonTest is the name of the test class created and it extends from ActivityInstrumentationTestCase2.

Inputs

As inputs for this test are cited:

- Instantiate a Controller class which has to MainTabActivity as activity base.
- Instantiate a GenericController class which has to MainTabActivity as activity base.
- Instantiate an activity monitor which has to IndicatorComparisonTabActivity as target activity.
- Instantiate an activity monitor which has to IndicatorDetailTabActivity as target activity.

This section is configured in the setup method of ControllerFromComparisonTest.

Process

The test case includes five methods described as follow.

- CallComparisonTabActivity. – It instantiate the IndicatorComparisonTabActivity in order to call to their methods.
- ConfigureComparisonTabActivity. – It configures test elements including a controller class which has to ComparisonTabActivity as activity base, and a genericController class which has to ComparisonTabActivity as activity base.
- TestRetrieveQueryConnectionsOneCountryYears. - It requests the query connections that Costa Rica has for the indicator trade balance rate. It checks that the result is not null.
- TestCallServers. - It calls to testRetrieveQueryConnectionsOneCountryYears and executes an asynchronous call to test the connection to Junar servers and retrieve their data. It uses an activity monitor to catch the result since IndicatorDetailTabActivity. It checks that the result is not null.
- TestCalculateIndicatorServers. – It calls to testCallServers and calculates the indicator using the Control class. It checks that the ConnectionResult list is not null. It checks that the ConnectionResult list size over to one. It checks that the ConnectionResult is not null. It checks that the QueryConnection list is not null and has a size over to one. It checks that the result contains result values for years from 2009 to 2013.

Outputs

The test case was executed successfully which implies that the following features work properly.

- Query of elements in QueryConnection table.
- Use of the retrieveQueryConnections method of the Control class.
- Use of GenericController class and test of connection to Junar servers using multiple query connections.
- Retrieve of data from a Junar server and manipulation of this data.
- Communication between activities.
- Calculation of historical data for trade balance rate indicator for Costa Rica.

This case uses four query connections with a response time of 7 seconds. Besides, this test case was used to retrieve the historical information of notifiable diseases for Chile. This second case used three query connections where 2000 rows were retrieved in every connection. The response time for this second case was 8 seconds. In order to reduce the response time, reduceResultString method was implemented in the GenericController class. More information about this implementation is in section 4.3.2.1. After execute the test case for second time, the response time was reduced from 7 to 5.4 seconds on the first case and from 8 to 5.8 in the second case.
H. System Testing: ST_01

This is to test the MainActivity fragment which is into MainTabActivity activity as well as the IndicatorComparisonActivity fragment of IndicatorComparisonTabActivity activity. ComparisonActivityTest is the name of the test class created and it extends from ActivityInstrumentationTestCase2.

**Inputs**
As inputs for this test are cited:

- Request the use of the year, indicator, first country, second country spinners, and buttonMA button from MainTabActivity.
- Instantiate an activity monitor which has to IndicatorComparisonTabActivity as target activity.

This section is configured in the setup method of ComparisonActivityTest. Besides this method starts MainTabActivity to charge the spinner widgets with data from the database.

**Process**
This test case includes a method called testComparisonUI, and its implementation can be outlined as follow.

- Initialize the spinners. Set to one the select index of the year spinner. Declare an assert clause to check if the achieved year is 2012. Set to one the select index of the indicator spinner. Declare an assert clause to check if the achieved indicator is average salary. Set to one the select index of the first country spinner. Declare an assert clause to check if the achieved country is Bolivia. Set to four the select index of the second country spinner. Declare an assert clause to check if the achieved country is Peru.
- Perform the click event of buttonMA and use the activity monitor to catch the IndicatorComparisonTabActivity activity.
- Catch the intent sent from MainTabActivity, and declare an assert clause that the intent extra message is not null.
- Recover the list of connection result objects, and declare an assert clause to verify the list size is equal to two.
- Declare assert clauses to check if the indicator detail values of the connection result objects are not null.
- Recover the MainTabActivity action bar and select the tab of index 0 which corresponds to MainActivity fragment.
- Recover the elements textFirstCountryNameIC, textFirstCountryValueIC, textSecondCountryNameIC, textSecondCountryValueIC of IndicatorComparison fragment.
▪ Declare assert clauses to verify that the values of the elements recovered in the previous step are the expected values.

**Outputs**
The test case was executed successfully which implies that the following features work properly.

▪ Querying elements of indicator and country tables.
▪ Charge of data into year, indicator, first country, and second country spinners in MainTabActivity.
▪ Execution of events of buttons in MainActivity.
▪ Navigability between MainTabActivity and IndicatorComparisonTabActivity.
▪ Connection to Junar servers.
▪ Calculation of indicators.
▪ Validation of information displayed in the interfaces.
I. System Testing: ST_02

This test is to verify the IndicatorComparisonActivity fragment which is into IndicatorComparisonTabActivity activity as well as the IndicatorDetailActivity fragment of IndicatorDetailTabActivity activity. ComparisonActivityTest is the test class used for this case because the code that we can reuse.

Inputs
As inputs for this test are cited:

- Request the use of the firstCountryMoreImageIC imageview element from MainActivity.
- Instantiate an activity monitor which has to IndicatorComparisonTabActivity as target activity.
- Instantiate an activity monitor which has to IndicatorDetailTabActivity as target activity.

This section is configured in the setup method of ComparisonActivityTest. Besides this method starts MainActivity to charge the spinner widgets with data from the database.

Process
This test case includes two methods, and its implementation can be outlined as follow.

CallComparisonActivity

This method emulates the test executed in appendix F, but for the following values.

- The year spinner selection is set to 2013.
- The indicator spinner selection is set to public investment.
- The first country spinner selection is set to Argentina
- The second country spinner selection is set to Argentina

TestDetailUI

- Execute the method CallComparisonActivity
- Perform the click event over firstCountryMoreImageIC.
- Use the activity monitor to catch the IndicatorDetailTabActivity activity.
- Catch the intent sent from IndicatorComparisonTabActivity, and declare an assert clause to control that the extra message is not null.
- Recover the list of connection result objects, and declare an assert clause to verify the list size is one.
- Recover the indicator detail value of the connection result.
- Declare an assert clause to verify if the value is not null.
- Declare an assert clause to verify if the value contains the results of the years from 2009 to 2013.
Recover the IndicatorDetailTabActivity action bar and select the tab of index 0 which refers to IndicatorDetailActivity fragment.

Recover the elements textFirstYearValueID, textSecondYearValueID, textThirdYearValueID, textFourthYearValueID, and textFifthYearValueID of the IndicatorDetailActivity fragment.

Declare an assert clauses to verify that the values of the elements recovered in the previous step are the expected values.

Outputs

The test case was executed successfully which implies that the following features work properly.

- Execution of events of image views in IndicatorComparisonActivity.
- Navigability between IndicatorComparisonTabActivity and IndicatorDetailTabActivity.
- The connection to Junar servers.
- The calculation of indicators for historical values.
- Validation of information displayed in the interfaces.
- The functionality of IndicatorDetailActivity.
J. System Testing: ST_03

This test is to verify the MapActivity fragment which is into MainTabActivity activity. MapActivityTest is the name of the test class created and it extends from ActivityInstrumentationTestCase2.

Inputs

As inputs for this test are cited:

- Request the use of year spinner from MainTabActivity.
- Request the use of indicator spinner from MapActivity.
- Request the use of buttonMap button from MapActivity.
- Request the use of mapViewM map view from MapActivity.
- Instantiate an activity monitor which has to MainTabActivity as target activity.

This section is configured in the setup method of MapActivityTest. This method starts MainTabActivity to charge the spinner widgets with data from the database.

Process

This test case includes a method called testCountriesUI, and its implementation can be outlined as follow.

- Set to one the select index of the year spinner. Declare an assert clause to check if the achieved year is 2012.
- Set to 0 the select index of the indicator spinner. Declare an assert clause to check if the achieved indicator is public investment.
- Perform the click event of buttonMap.
- Use the activity monitor to catch the MainTabActivity activity.
- Recover the list of connection result objects, and declare an assert clause to verify that the list size is equal to five.
- Declare assert clauses to check if the indicator detail values of the connection result objects are not null.
- Recover the MainTabActivity action bar and select the tab of index 1 which corresponds to MapActivity fragment.
- Recover the mapViewM element of IndicatorComparison fragment and recover its markers positions.
- Declare assert clauses to verify that the values of the markers are the expected values.

Outputs
The test case was executed successfully which implies that the following features work properly.

- Querying elements of indicator and country tables.
- Charge of data into year, indicator, first country, and second country spinners in MainTabActivity.
- Execution of events of buttons in MainActivity.
- Navigability between MainTabActivity and IndicatorComparisonTabActivity.
- Connection to Junar servers.
- Calculation of indicators.
- Validation of information displayed in the interfaces.
K. Tasks execution

Task 1. - Find the public investment amount for Argentina and Chile in 2011.

![Figure A.1 Results Task 1](image)

Task 2. - Find the average salary in Bolivia on 2010.

![Figure A.2 Results Task 2](image)

Task 4 - Find all countries which have information about notifiable diseases indicator on 2012.