A Framework to build Information Systems using a Public Cloud

Diogo André Cardoso Serafim

Thesis to obtain the Master of Science Degree in

Telecommunications and Informatics Engineering

Supervisor: Prof. José Luís Brinquete Borbinha

Examination Committee
Chairperson: Prof. Fernando Manuel Bernardo Pereira
Supervisor: Prof. José Luís Brinquete Borbinha
Member of the Committee: Prof. António Manuel Ferreira Rito da Silva

October 2017
Acknowledgments

First of all, I would like to thank to my professor and thesis advisor, José Borbinha, for all the support and for always being available to help me clear my thoughts, motivate me and also help me review my work.

To Instituto Superior Técnico including all the staff and professors, my gratitude for all the knowledge they have passed on to me.

To my dear friends Pedro Frade, Filipa Carvalho, Pedro Baptista, Sónia Garrido Gil, Fábio Domingos, Giovanna Franchi, Joka Barbosa and Ricardo Wagenmaker, a deeply thank you for being such an inspiring force through my life, for always being present and giving support.

I also want to show my appreciation to all the people that appeared during this journey, that through their presence and encouragement have contributed to the success of this work, these are: Sofia Ribeiro, Rui Lourenço, Laura Barroso, Valdo Lopes, Ivo Pires, Diogo Costa, Jorge Almeida, Rita Curado, and Maria Quaresma.

I also want to show my gratitude for all the support given from my family, specially to my father and mother, who have worked hard their entire life to enable me and my sister to go through university.

Finally I want to dedicate this dissertation to my grandparents, who made me the man I am today and taught me that honesty and humility trumps every challenge.
Abstract

In the last years cloud computing has revolutionized the way organizations use computers and the internet. Most of the services offered by popular cloud providers, allow organizations to migrate their information systems to the cloud based on subscription-plans regarding the resources they need.

However, a few number of small organizations are still unable to support the costs associated. On the other hand, most of systems still in use reside exclusively in Microsoft Excel, with the information spread over multiple documents without any automatization leading to long and exhaustive operations.

The purpose of this thesis was the development of a framework that can be adapted to different small organizations by offering a set of tools to migrate the existing systems, as also to easily create a physical database using public cloud services. Finally, considering all the research made about the capabilities and features provided by some cloud providers, extend the framework to new web applications fully integrated with the system implemented in the cloud and, most important, with no costs associated. The proposed framework was evaluated by users to obtain feedback about the framework’s usability and user experience. After a positive evaluation, some improvements were made which generated the second version of the new DB4G framework.

Keywords: Cloud computing, Software as a Service, Framework, Google Docs, Extend.
Resumo

Nos últimos anos, a computação em nuvem tem revolucionado a forma como as organizações usam os computadores e a internet. A maioria dos serviços providenciados pelos provedores de cloud mais conhecidos, permitem às organizações migrar os seus sistemas de informação para a cloud com base em planos de subscrição alinhados às suas necessidades.

Contudo, as pequenas organizações ainda não conseguem suportar os custos associados. Por outro lado a maioria dos sistemas ainda em uso reside exclusivamente no Microsoft Excel, com a informação dispersa por vários documentos, sem qualquer automatização, levando a operações longas e exaustivas.

O objetivo desta tese consistiu no desenvolvimento de uma framework que possa ser adaptada a pequenas organizações ao fornecer um conjunto de ferramentas para migrar os sistemas existentes, bem como criar uma base de dados utilizando os serviços da cloud pública. Finalmente, considerando toda a pesquisa realizada acerca dos recursos providenciados por alguns provedores de cloud, estender a framework a novas aplicações web totalmente integradas com o sistema implementado na cloud, e mais importante, sem quaisquer custos associados.

A framework foi posteriormente avaliada por potenciais utilizadores para obter feedback sobre a sua usabilidade e experiência de utilizador. Após uma avaliação positiva, foram realizadas melhorias que geraram a segunda versão da framework.

**Palavras-chave:** Computação em Nuvem, Software como um serviço, Framework, Google Docs, Extenção
## Contents

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Resumo</td>
<td>iii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>Acronyms</td>
<td>ix</td>
</tr>
</tbody>
</table>

### 1 Introduction

1.1 Problem Description .................................................. 2
   1.1.1 Animalife .................................................. 2
   1.1.2 APOGEP .................................................. 3
1.2 Motivation and Objectives ........................................... 3
1.3 Contributions ..................................................... 4
1.4 Document Structure .................................................. 4

### 2 Related Work

2.1 Cloud Computing Overview ........................................... 5
   2.1.1 Cloud Computing Core Concepts ............................... 5
   2.1.2 Web Authentication & Authorization Open Technology .......... 7
2.2 Cloud Computing Solutions .......................................... 8
   2.2.1 Software as a Service Solutions ............................... 8
   2.2.2 Application Programming Interfaces ........................... 12
2.3 Software Development Models ....................................... 13
   2.3.1 Main Methods for Software Engineering ....................... 13
   2.3.2 Software Prototyping Models .................................. 14
2.4 Summary .................................................................. 15
3 Problem Analysis and Solution Design 16

3.1 Requirements .......................................................... 16
  3.1.1 Front-end Users .................................................. 18
  3.1.2 Back-end Users .................................................. 19
3.2 Framework Architecture and Technology .......................... 20
3.3 Summary ............................................................... 22

4 Implementation 23

4.1 Home Page Section .................................................... 23
  4.1.1 Login ............................................................... 24
  4.1.2 User Consent .................................................... 24
4.2 Project Management Section ......................................... 25
  4.2.1 Create Project .................................................... 25
  4.2.2 Project Area ...................................................... 26
4.3 System Management .................................................. 27
  4.3.1 Users Management ................................................. 27
  4.3.2 Configuration File ............................................... 28
  4.3.3 Recycle Bin ....................................................... 28
4.4 Database Management ............................................... 29
  4.4.1 Create File ........................................................ 29
  4.4.2 Preview, Edit and Delete File ................................... 30
  4.4.3 Import and Export Files ........................................ 31
  4.4.4 Table Management .............................................. 32
  4.4.5 User Permissions ................................................ 33
4.5 Framework vs Google APIs Limitations ............................ 33
  4.5.1 File Ownership ................................................... 33
  4.5.2 Quota Limits ...................................................... 34
  4.5.3 Performance ....................................................... 35
  4.5.4 Work Offline ...................................................... 36
  4.5.5 Backup project ................................................... 36
4.6 Summary ............................................................... 36

5 Framework Extension 37

5.1 Planning and Requirements ......................................... 37
5.2 Implementation ....................................................... 38
  5.2.1 Adding a new Google API ....................................... 38
  5.2.2 Adding new controllers and views .............................. 39
  5.2.3 Working with data ............................................... 39
5.3 Summary ............................................................... 39
# List of Figures

2.1 Cloud Computing Architecture ....................................................... 6  
2.2 Google Docs Revision Tool .......................................................... 10  
2.3 AppSheet Interface Management .................................................... 12  
3.1 Domain Model .............................................................................. 17  
3.2 Use Case Diagram: Front-end .......................................................... 18  
3.3 Use Case Diagram: Back-end ........................................................... 20  
3.4 Architecture of DB4G Framework .................................................... 21  
4.1 Main Page - Home section ................................................................. 24  
4.2 Google Authentication Server - Consent screen .................................. 25  
4.3 Project Area - page section .............................................................. 26  
4.4 System Management: Users Management section ............................. 27  
4.5 System Management: Recycle Bin section ........................................ 28  
4.6 Database Management area ............................................................. 29  
4.7 Add new table to the database .......................................................... 30  
4.8 Document edition ........................................................................... 31  
4.9 Import File - Process Overview ....................................................... 31  
4.10 Table Management ......................................................................... 32  
6.1 Comparison between the number of errors found in a website and the number of participants that made the test (Nielsen, 1993) ......................................................... 42
List of Tables

4.1 Google Drive API - quota limits .............................................. 35
4.2 Google Sheets API - read quota limits ...................................... 35
4.3 Google Sheets API - write quota limits ..................................... 35

6.1 System Usability Scale ............................................................... 43
6.2 Results from question 1 (first table in Appendix E): medians and modes. ......................................................... 46
6.3 Results from questions 2, 3 and 4 (second table in Appendix E): average, medians and modes ........................................... 46
6.4 Results from question 5 to 11 (third table in Appendix E): average, medians and modes .............................................. 47

E.1 Users evaluation table: results from usability question 1 ......................... 87
E.2 Users evaluation table: results from usability questions 2, 3 and 4. ............... 87
E.3 Users evaluation table: results from user experience questions 5 to 11 ............... 88

F.1 Class Domain Model - attributes description ....................................... 89
List of Acronyms

APOGEP  Associação Portuguesa de Gestão de Projectos
IaaS  Infrastructure as a Service
PaaS  Platform as a Service
SaaS  Software as a Service
API  Application Programming Interface
HTML  HyperText Markup Language
HTTP  HyperText Transfer Protocol
PHP  Hypertext Preprocessor
CSS  Cascading Style Sheet
JSON  JavaScript Object Notation
XML  Extensible Markup Language
MVC  Model View Controller
CRUD  Create, Read, Update, and Delete
SUS  System Usability Scale
UEQ  User Experience Questionnaire
Chapter 1

Introduction

In the last years cloud computing has revolutionized the way organizations use computers and the internet, specially how the information is stored and the way people use the applications.

Most of the existing services offered by the popular cloud providers, allow organizations to migrate their current information systems to the cloud based on subscription-plans regarding the resources they need as well as the number of users.

At the same time, organizations are able to improve their business performance, by speed up the existing processes and reducing infrastructure and computing costs.

However, a few number of small organizations are still unable to support the costs associated and so they are still reluctant to adopt cloud computing resources as a solution for replacing the current systems.

In fact, in the past most of the small organizations could not afford the investment in IT infrastructure and software as also the necessary maintenance, compromising their performance and leaving the organizations striving with maintain the existing system instead of focusing on their main goals. Only the larger organizations had the necessary funds to invest in the infrastructure to meet daily operational requirements.

On the other hand, most of systems currently in use by those organizations resides exclusively in Microsoft Excel, with parts of the information spread over multiple documents without any automatization leading to long and exhaustive operations.

In order to overcome all these problems, it would be interesting to study and implement a solution that allows small organizations to build (or migrate) their information systems to the cloud using public cloud services without any cost or subscription plans.
1.1 Problem Description

Cloud computing is defined as “a nebulous assemblage of computers and servers accessed via the Internet” providing resources and services to store information regarding users and organizations accessible from anywhere [1]. The provision, deployment and maintenance of those servers is made by a cloud computing platform (e.g. Google Cloud Platform).

Since that organizations continue to grow every day, not only in the number of employees, but also in the number of departments and the type of employees, cloud computing is becoming the solution to help organizations to attend their needs and accomplish their goals, as also to increase their business value.

Since that multiple employees work at the same time in a real-time collaboration, cloud software applications also provide tools to configure user controls, that can be used to restrict access to specific areas of the system which helps to prevent security risks associated with human error.

In the past, most of the small organizations used Microsoft Excel to organize their data and share it among the employees. While this system could be convenient in small organizations, it turned out to compromised, corrupted and inaccessible data. Nowadays, instead of accessing spreadsheets separately, cloud applications provide tools to integrate spreadsheets with many aspects of the software, which removes the necessity of importing and exporting data to and from multiple areas as typically occurs in the old systems.

However, despite all the features mentioned above, a few number of small organizations are still unable to support the costs associated to the subscription-plans offered by the cloud services, and for that reason they are still reluctant to adopt the cloud computing solutions.

To be more precise, we describe two real scenarios of two portuguese non-profit organizations: Animalife and APOGEP.

1.1.1 Animalife

Animalife is a non-profit organization of Social and Environmental Support, founded in October (2011). With headquarters set in Lisbon, and more than 50 volunteers, it is responsible for conduct and support initiatives to improve the quality of life of families in need by giving support to the animals that are in their care, thus, preventing the abandonment of animals and over-population in kennels and associations hostels.

Animalife needs to manage the records of the families and animals they support, as also of its own stock of food and medicines. However, the information is maintained in paper records or in Microsoft Excel spreadsheets spread over multiple documents, leading to long and exhaustive operations. This problem, aligned with the fact that the organization collects and records data in different ways according to different geographical areas, leads to an outdated system, compromising the performance and organization of the institution. This problem is part of a thesis currently being done by another colleague (Tiago Neto), who is developing a specific solution for the organization.
1.1.2 **APOGEP**

Associação Portuguesa de Gestão de Projectos (APOGEP) is a non-profit association with headquarters in Lisbon and founded in 1994, whose main goal is to certify project managers, consultants and organizations in project management domain.

Currently linked to International Project Management Association (IPMA), the association counts with more than one thousand certificated professionals and a relevant collective associate portfolio. After meeting some stakeholders of the association, including Certification Body (CB) Director, it was proposed to implement a new system that can replace the current one, which is based exclusively on Microsoft Excel worksheets without any automatization.

This problem is part of a thesis currently being done by another colleague (Jorge Almeida), who started to develop a new web application for the organization using the functionalities and extension capabilities provided by the solution proposed in this document.

1.2 **Motivation and Objectives**

To overcome the problems of each organization, it would be interesting to study and implement a web-based framework that allows different organizations to build (or migrate) their information systems to the cloud using public cloud services, without any costs or subscription plans.

For the case of Animalife, the framework should provide an easy way to migrate the existing spreadsheets files into the cloud. With cloud computing resources, volunteers can step away from their desks and go out to serve the community without losing control over data management processes. On the other hand those resources can be accessed from anywhere via different devices, such as laptops as well as mobile devices and tablets.

For the case of APOGEP, the framework should provide a set of tools to easily create a physical database based on the domain model of the actual system in use. Finally for both systems, the framework should provide a set of generic tools for developers to extend the framework to a new web-application specific to the each organization.

To conclude, the objective of this work was the development of a web-based framework that first, explores all the capabilities and features provided by a specific cloud provider. Then, it should be developed in a way that it can be adapted to any type of organization by offering a set of generic tools to migrate the actual systems, as also tools to easily create a physical database using public cloud services. Finally, it should be possible to extend the framework to new web applications, fully integrated with the information system implemented in the cloud.

The final framework must be quick, efficient, and easy to use. It must offer useful features without overwhelming the users with options.
1.3 Contributions

The contribution resulting from this work is the development of a web-based framework that allows organizations to build (or migrate) their information systems to the cloud using public cloud services, without any costs or subscription plans by offering:

- An intuitive interface for use by non-experts users (assuming they have only a generic knowledge about Google Docs and spreadsheets).
- The possibility to create a system in Google Drive for a specific organization;
- Users management functionalities and real-time collaboration;
- Database and File management functionalities;
- Export and Import functions to migrate the current systems;
- The possibility to extend the framework to other APIs and functionalities by developers using generic templates;
- The possibility to extend the framework to new web applications, fully integrated with the system created in cloud;
- Starting point on the development of a new information system for a non-profit organization - APOGEP.

1.4 Document Structure

The rest of this document is organized according to the following structure. Chapter 2 presents some related work to our problem, which includes an overview about cloud computing, including web authentication and authorization protocols. It also provides a deep analysis of the cloud services and applications most used nowadays, as also a set of application programming interfaces, which allows communication between third-party web applications and cloud services and their integration with other services. Finally it provides an overview of the most know software development methods used in software engineering.

Chapter 3 describes the problem analysis and design made, like requirements, use cases, architecture and technology used. Chapter 4, describes the framework's implementation and functionalities in great detail.

Chapter 5 explains the extension of the framework to new functionalities and web applications, and the necessary requirements to do it correctly. Chapter 6 describes the evaluation method, as well as the results of the user testing and the conclusions made.

The last chapter concludes the dissertation with an overall discussion of the work and proposed future work. Finally, in the appendix we have the administrative and user manuals of the framework, and also the guidelines, survey and results presented in the evaluation chapter.
Chapter 2

Related Work

Since the main goal of this project is the development of a framework using the resources and technologies available across the cloud, it is necessary to understand in a first approach the concept of cloud computing including its architecture and business models.

This chapter provides an overview about the core concepts of cloud computing and some open standards related to the authentication and authorization protocols. Section 2.2 provides a careful analysis of different Software as a Service (SaaS) solutions and Application Programming Interfaces (APIs) offered by some cloud providers. Finally section 2.3 describes the software development models used in software engineering.

2.1 Cloud Computing Overview

Cloud Computing is described by the National Institute of Standards and Technology (NIST) as a "model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"[2].

2.1.1 Cloud Computing Core Concepts

Cloud provides an architecture that can be divided into 4 layers: the hardware layer, the infrastructure layer, the platform and the application layer, as described below [3]:

- The hardware layer - is a layer responsible for the management of the physical cloud resources which include the data centers composed by physical servers, storage systems, networking routers and switches, power and cooling systems, as well as the cabling and physical racks used to organize and interconnect the equipment. Typical issues at hardware layer include hardware configuration, fault tolerance and traffic and power management.
• The *infrastructure layer* - creates a pool of storage and computing resources by partitioning the physical resources using virtualization technologies (e.g., KVM, Xen, VMWare). It is an essential component of cloud computing since some features are only available through virtualization technologies.

• The *platform layer* - is composed of the operating systems and application frameworks. Built on top of the infrastructure layer, the platform tries to minimize the load of deploying applications directly into VM containers. One of the best examples is the Google App Engine that operates at the platform layer to provide API support for implementing storage, database and business logic of typical web applications.

• The *application layer* - located at the highest level of the hierarchy, consists of the actual cloud applications that make use of automatic-scaling feature to achieve better performance, availability and lower operating cost.

Cloud computing also employs a service business model where hardware and platform-level resources are provided as services on-demand, and can be assigned to each layer and grouped into three categories which are [3]:

• *Infrastructure as a Service (IaaS)* - consists in the provisioning of infrastructure layer resources, i.e. users can subscribe to their favorite computing infrastructures with specified requirements in terms of hardware configuration, software installation and data access. Examples of IaaS providers include Amazon EC2 and Google Compute Engine.

• *Platform as a Service (PaaS)* - consists in the provisioning of platform layer resources, including operating system support and software development tools to develop, run and manage applications. Examples of PaaS providers include Google App Engine and Microsoft Azure.

• *Software as a Service (SaaS)* - is hosted as a service and provided to customers over the Internet. This is a very important service since it eliminates the need to install and run the application on the local computer, by providing subscriptions to the software hosted on the Internet [4]. Examples of SaaS include Google Docs, Microsoft Office 365 and AppSheet.
There are three types of different clouds, each one with its own benefits and challenges:

**Public cloud:** is a type of cloud in which service providers share their virtual resources as services to the public in general. Therefore, it provides a highest level of efficiency in shared resources and unlimited scalability. However, it lacks on the control over data, network and security settings, which compromises its effectiveness in many business scenarios.

**Private cloud:** is a type of cloud in which services and infrastructure are maintained on a private network. It provides a highest degree of control over security, reliability and performance. However, private clouds require a certain company to still purchase and maintain all the software and infrastructure.

**Hybrid cloud:** is a combination of the public and private cloud models where specific resources are used in public and while others are used in private cloud. The intent of this type of cloud is to address the limitations, specifically provide a strait control and security over application data compared to public clouds. However, it is necessary to guarantee that all the platforms are very well integrated and ensure that all aspects of a business can communicate with each other.

### 2.1.2 Web Authentication & Authorization Open Technology

As mentioned before, Cloud computing is changing the way users interact with devices, software, data and processes. However, the organizations are reluctant to store their sensitive data in the cloud due to various privacy and identity tracking threats.

In the past few years, a lot of research and development efforts have been made to define centralized and federated security mechanisms for the protection of identity information in a cloud environment [5]. This section provides an overview about the protocols used for authentication and authorization through the cloud.

- **OpenID** - is an open standard for user authentication that allows the use of an existing account to sign into multiple websites without needing to create new passwords. To do that a user must obtain an OpenID account through an *identity provider* (e.g. Google). Then the user only have to use that account to sign into any website (i.e the *relying party*) that accepts OpenID authentication. With OpenID the user’s password is only given to the identity provider and this one confirms the identity to the websites visited so, no website sees the password. Therefore, it is considered as safe since the user’s identity is not compromised ¹.

- **OAuth2.0** - is an open standard for user authorization that provides a *secured delegated access* for internet users, by delegating user authentication to the service that hosts the user account and authorizing third-party applications to access the user account without sharing their credentials such as passwords or usernames. Designed to work specifically with HTTP, it allows access tokens to be issued to third-party applications by an authorization server, on behalf of a resource owner. Then, the third-party application uses the access token to access the protected resources hosted by the resource server [6].

¹ Information provided by: http://openid.net
2.2 Cloud Computing Solutions

The following SaaS and PaaS solutions were relevant for studying and gaining knowledge about the services offered by some cloud providers. The purpose was to study these services and explore their features and capabilities, advantages, disadvantages and aspects that could be improved. Furthermore, what could be interesting to be reused, and what new functionalities would be interesting to have in the new framework.

2.2.1 Software as a Service Solutions

As mentioned before, Software as a Service (SaaS) describes any cloud service where clients are able to access software applications over the internet.

In SaaS model, users subscribe for the software rather than purchase it, and most important, they subscribe for the exact quantity of the service they require. Applications are subscribed and used online, with files saved in the cloud rather than on individual computers.

Microsoft Office 365

Office 365 is a software as a service (SaaS) suite that allows users to use the familiar applications that have been used for many years in their own computers which include Word, Excel, PowerPoint, and, depending of the subscription chosen, Outlook, OneNote, Access and Publisher to save, edit and modify documents in the cloud.

In addition, it provides a cloud file-hosting service, Microsoft OneDrive, which allows the storage, backup and sharing of documents, photos and videos and their access using a computer or mobile device.

One of the biggest strengths of Office relies on its support for collaboration and sharing of information. With Office 365, teams of an organization can work together, tracking tasks, project timelines, documents and emails all from one centralized place [7].

For collaboration and sharing of information, Office 365 provides three collaborative platforms:

- *Microsoft Exchange Online* - is a platform for email management and administration, calendar and contacts synchronization across all the devices.

- *Skype for Business* - is platform fully integrated with Exchange and Sharepoint providing communication in real time such instant messages, video conferencing and screen sharing across multiple devices.

- *SharePoint* - is a collaborative repository platform designed to host and provide storage, management and access of files between organizations. It provides tools for tracking multiple versions of a file and search features to help users find specific content efficiently independently of physical location. It also offers a set of integrated development tools to help developers in rapid prototyping of web-based applications.
As mentioned above, Office 365 include web-based editing and sharing tools used for editing the files among users in real time.

However, it does not offer a history revision tool (something that is available in other cloud services) which provides a full history of all the changes made to a document, allowing users to restore earlier versions of a file.

Office 365 is license-based, offering various subscription plans to fit the needs of organizations, personal users and also non-profit organizations. Each one determines, based on their needs, which Office 365 features will have to subscribe. It can also be extended with add-on tools from third-party developers, which in this case can be important for the context of this project.

Since that one of the main goals of this work is to develop a framework to build information systems that can be applied in two non-profit organizations, it was also important to explore a package of subscription and donation plans, that Microsoft designed specifically to support non-profit organizations. The Office 365 Nonprofit package was created to empower staff and volunteers, to be succeed with the dynamic set of tools included in Office 365. It provides team members with free email and online document editing, storage and video conferencing.  

Unfortunately, even having a free package donation for non-profit organizations, Microsoft states that it does not guarantee the total security of information in this package and for that reason it cannot be considered as part of the solution to this problem. However, it might be useful as an integration tool for editing documents through the cloud.

**mLab**

MLab is a database as a service platform designed to host MongoDB databases. Typically used for document storage, it runs on cloud providers Amazon, Google, and Microsoft Azure (IaaS partners), providing a set of subscription-plans and 500MB of free storage on a shared virtual machine, typically used for the development and prototyping of web applications. As capabilities and features it offers automated provision and scaling of databases, backup and recovery and web based management tools. It is another type of solution that might be included in the conception of this project.

**G Suite**

G Suite (known until one year ago as Google Apps) is another software as a service suite developed by Google, that provides a set of productivity and collaboration tools and software such Gmail, Google Docs, Google Drive, Hangouts and Calendar. Google Docs is a totally free web-based software office suite that allows users to create, share and edit documents through the cloud. It also includes Google Sheets and Google Slides allowing users to create spreadsheets and presentations. The simplicity of Google Docs resides on the fact that, it is compatible with Microsoft Office file formats, so users can upload word or other text-based documents.

---

3 Information retrieved from mLab: https://mlab.com/company/
into the system, and convert them to the online editing program. From there, multiple users can create or edit the same files online, in a real-time collaboration.

The revision history tool (not available in Office 365) provides a full history of all of the changes made to the document, which is extremely useful for keeping track of changes (see figure 2.2).

However, Google Docs requires internet access while Microsoft Office does not, which sometimes compromise the speed when updating files. For example, if certain spreadsheet files contain a large number of cells and formulas, it may take a significant amount of time for the Google Sheets file to update, when compared to an Excel file of the same size.

Google Docs is fully integrated with Google Drive, and all files created are by default saved in Google Drive.

Google Drive is the cloud storage service created by Google. It allows users to create, store, synchronize and share files in the cloud. For those who have a Google account, it is easy to access Google Drive and enable the service in seconds. However, the 15GB initially provided are shared with other services such as Gmail (it is possible to save attachments from the email directly to drive), and Google+. Like the other cloud providers, Google Drive can be accessed through a web browser or mobile device. The platform also provides tools to import and export Microsoft Office files to and from the cloud. However, the exportation and conversion process to Office formats may not always be the best.

As mentioned above, Google Docs is totally free, and for that reason it might be used as part of the solution of this project in order to store, access and manage all the documents in the cloud. For that reason, a study about the success of G Suite in various types of organizations and companies was made, in order to understand, how they increased their value by using this set of productivity tools.
Google Apps Success Stories

As we know, communication and integration are directly related to business success of a company. Since his creation, G Suite has become one of the most used productivity tools by millions of companies and organizations around the world. It offers a set of tools and options that cover all the centralized sectors of a company, by integrating all the sectors without compromise their integrity and efficiency.

Starting by Iontec Computing and System (Brazil)\(^1\) a company dedicated to the development of information systems. The company faced many challenges in email, in consequence of an outdated server with many overhead problems and lower resources. In to have more stability in email services the company adopted G Suite as their principal tool of work.

Another interesting example is the case of Ascentis Accountants (United Kingdom). The company was facing an expensive upgrade of their old document management system which aligned with incompatibilities in file formats, leaded to complex and long operations, thus increasing the costs of the company and compromising his performance. The company replaced Microsoft Office 365 and Outlook, by G Suite for Business. The migration of the existing documents and metadata from the old document management system to Google Drive made everything easier and accessible everywhere, improving accessibility and collaboration.

To conclude and according to the information available and provided by Google Partners, over 5 million companies are using G Suite to work smarter and faster, increasing their collaboration and integrity of internal and external information, used by many of the world's largest businesses including Jaguar Land Rover (24,000 employees), Rentokil Initial (32,000 employees), Roche (pharmaceuticals 100,000 employees) and BBVA (Banking, 100,000 employees)\(^2\).

AppSheet

AppSheet is another software as service platform, seeking to revolutionize the app market by offering to people with no coding experience, all the tools they need to turn their cloud drives and spreadsheet data, into functional mobile applications.

The platform uses a cloud account (e.g Google Drive), SQL server or any spreadsheet file (it supports Google Sheets and Excel) containing data to create functional application. The service automatically parses all the content inside the spreadsheet and create the initial application in seconds.

Once the application is created, users can customize the interface, security and user permissions of other users enrolled. Since the account is cloud-based, users can save and resume the project's progress from anywhere.

The disadvantage of this service is that it only have been designed to build functional mobile applications.

---

\(^1\)Information provided by: QiNetwork - Google for Work Partner in Brazil

\(^2\)Information provided by Ancoris: http://89-145-92-239.as29017.net/cloud-computing/google-apps-for-business.html
2.2.2 Application Programming Interfaces

SaaS applications can also be accessed by an application programming interface (API) or by applications that utilize the API (known as third-party apps). APIs are defined as a set of routines, functions, protocols and programming standards, used for accessing software application or just for interacting with an external system.

When used in the development of third-party applications and services, they act as a gateway enabling direct or indirect access to hardware, software and platform services hosted on the cloud.

Google APIs

Google APIs is a set of programming libraries and protocols developed by Google, whose main goal is to provide communication with google services and their integration with third-party applications. It also gives developers, capabilities to build their own services and to extend their functionality.

In terms of authentication and authorization protocols, Google APIs makes use of OAuth2.0 (described in section 2.1.2). The APIs provide a developer console where the client app can obtain the necessary credentials and then request an access token from an authorization server, when accessing to a specific service.

There are client libraries in various languages, allowing developers to use Google APIs within their code, such as Java, PHP, JavaScript, .NET, etc.

In the context of this project, only G Suite APIs will be described:

- **Drive API** - allows a third party application to interact with any aspect of a user's Google Drive account and all the functionalities available (e.g. create, update, delete a file, user permissions, etc).

- **Realtime API** - is a JavaScript library that provides collaborative objects, events, and methods for creating collaborative applications.
• *Sheets API* - allows the development of third-party applications, that create, read, update and modify spreadsheet data in Google Sheets.

• *Apps Activity API* - allows to track changes to a user’s Google Drive file, including new comments and other activity.

• *Gmail API* - allows to third-party applications, access to a user’s mailbox under. It supports CRUD operations on Gmail data types such as messages, threads, labels and drafts.

**Microsoft Graph API**

Microsoft Graph API is a unified API that exposes multiple APIs and other Microsoft cloud services through a single endpoint[^4]. His creation came with the necessity of simplifying queries that would otherwise be more complex.

As in the case of Google APIs, Microsoft Graph makes use of OAuth2.0 as authorization and authentication protocol, providing access to a user’s data by third party applications. It can be used to access data from multiple cloud services, including Azure Active Directory, Exchange Online as part of Office 365, SharePoint, OneDrive, OneNote, etc. The core value of Microsoft Graph is to unify Microsoft APIs and to have a much simpler developer experience across all of Microsoft's APIs, by bringing them together on a single endpoint with a single authentication[^5].

### 2.3 Software Development Models

In the past years, organizations have seen changes in the field of software development when it comes to implementation models. Slowness to accommodate uncertainty and rapid changes inevitably lead to the obsolescence of software, turning software artifacts already planned, designed, specified and even already implemented in obsolete and outdated artifacts[^8]. For that reason one of the most important approaches for software engineering in the last years is the use of software development methods, which are described in this section.

#### 2.3.1 Main Methods for Software Engineering

In the field of software engineering, a software development method can be defined as the splitting of a project development life cycle into several stages, where each stage can be executed in different ways producing one or more product artifacts. During the project life cycle, collaboration between developers and clients may occur in order to establish new requirements or detect potentials issues in the designing process. In the end, the final product will be a result of a full collaboration, planning and work management between who wants the product (i.e. the client) and who designs the product (i.e. the developer). Over the years, many software development models have been created and some of them are listed bellow[^9]:

[^4]: Information retrieved from: http://graph.microsoft.com
1. **Waterfall** - is a sequential model where the software development life cycle is divided into 5 stages: (1) Requirement Analysis, (2) Design (3) Implementation (4) Verification (5) Maintenance, with each phase having its own tasks and goals. Each stage only begins if the previous stage is completed, and since it is sequential, there is no place for change or error. Waterfall is useful in small projects where requirement specifications are very well defined and documented before implementation, with each stage having specific deliveries and reviews. However, it is not suitable for large projects, where exists a higher risk of changing in requirements during the development of the project.

2. **Agile** - was developed to overcome the limitations of the Waterfall model. It follows an incremental and iterative approach to development, where requirements and solutions evolve through collaboration between the client and self-organizing teams (i.e. planners, designers, developers and testers). In this case, developers start with a simplified project design, and then begin to work on small modules, typically in weekly or monthly sprints. At the end of each sprint, issues are detected, project priorities are evaluated, new requirements are defined and tests are performed. However, Agile depends deeply of client interaction, so if the client is not clear or sometimes is not available, team members can be driven in a wrong direction.

3. **Software Prototyping** - can be defined as the activity of building application prototypes which display the functionality of the product under development and are used by clients to identify requirements and help developers to understand what is the right path they have to follow in order to meet end-user requirements before the final implementation.

Moreover, an information system prototype can be defined as “...a system that captures the essential features of a later system, is the most appropriate definition of a prototype. A prototype system, intentionally incomplete, is to be modified, supplemented or supplanted.” [10].

### 2.3.2 Software Prototyping Models

In this section, a general outline of the characteristics of the prototyping process models is presented and major features are identified [10].

1. **Rapid Throwaway Prototyping** - is a prototyping model where developers are responsible for developing the prototype as quick as possible, and clients provide feedback by evaluating the prototype which is used to refine software requirements. Only poorly understood requirements are implemented first and quality factors such efficiency, maintainability, portability and documentation are not take into account. Once there is agreement between client and developers on requirements, the prototype is “thrown away” and developers proceed to the design of the system. However, since the prototypes are done as quickly as possible, it is possible that certain aspects can be neglected and developers may not have sufficient time to think of better solutions and conduct a deeper analysis. As a result of this constraint, the final product may not always be the best.
2. **Evolutionary Prototyping** - is a prototyping model where an initial prototype is presented to the client and based on the feedback provided and new requirements, it is constantly refined until it can be turned into a final product. Since that information systems are embedded in an organizational context, as the organization evolves, new requirements appear. In contexts of high uncertainty, the evolutionary prototyping approach dynamically responds to changes in user needs and accommodates changes in requirements as the development of the system evolves (Pape and Thoresen, 1992). As in the case of throwaway prototyping, clients provide feedback to the developers by evaluating the functional prototype, detecting anomalies and opportunities for new features. Therefore, developers can change specifications, update the design and enhance the prototype until accomplish the final system. However, the system may not have all the features that users may have planned, and it can be used as a basis until the final system is done and delivered.

3. **Incremental prototyping** - can be defined as the process of building multiple functional prototypes of a system and then integrating all the prototypes to form a complete system. Typically, development starts with a initial prototype as a base, and then features are added as the prototypes are developed. When all the requirements and features are in place, the final product is released. An advantage of this method is that, it allows the client to provide feedback while other components are still in development, reducing the communication gap between the client and developers, and thus influence the outcome of further development.

4. **Extreme Prototyping** - is a model that was designed especially for developing web applications. It breaks down web development in three phases. The first phase consists in the development of a static prototype composed mainly of HTML pages and some logical data model. The second phase is a coding process where the HTML pages are fully functional using a fully functional prototype service layer. Finally services are implemented and integrated in the final prototype.

### 2.4 Summary

In this chapter, a set of software as a service solutions were carefully analyzed in order to understand which type of cloud service should be used as cloud-based platform in the framework. It was also made a study about two types of APIs used by end-users to access cloud provider services.

Before making a decision about the right service and API platform for this project, we must be aware that there may be needed different API models in order to provide different system functionalities. The compatibility between platform and infrastructure is not always clear so it is important to plan in advance the deployment and ensure that, prior to any development all pieces can interoperate and perform compatibly. This analysis and observations will be considered during the rest of the work and development of a solution in order to achieve a useful and simple framework.
Chapter 3

Problem Analysis and Solution Design

This chapter describes the analysis and discussions made, and justify the decisions taken for the final solution of the new framework (named as DB4G Framework). DB4G Framework is a web-based integrated framework which makes use of the Google Cloud Platform and Google Docs services to communicate, create and maintain an information system in Google Drive, available in the cloud without any costs associated.

The framework should be friendly and easy to use by any user (with at least some knowledge about Google Docs and Google Drive) without having a complex interface and can be accessed anywhere using a computer with internet access. Also when extending the framework to new functions or to a new web application for a specific organization, trying to figure a way to build a generic and simple guide for the new additions. The addition of new functions must follow the same interface and main functionalities. Finally, after considering the different possibilities for the development of a solution taking in consideration the main objectives, it was defined the final solution design.

3.1 Requirements

One of the biggest challenges of the previous research was to find a service were our solution could be deployed with no costs associated. After a careful research it was found that only Google Docs satisfies that condition. Furthermore, the different services were analyzed in order to find disadvantages and some potential capabilities that may be interesting to have in DB4G framework. Most of the services such Office 365 and Google Docs offer a lot of interesting functionalities such the real-time collaboration, a history revision tool and the compatibility with Microsoft Office file formats, which may be interesting to keep in the development of the framework. AppSheet is the most interesting platform and probably the only one that can be used as a starting point for the design of our solution, by offering to people with no coding experience, all the tools to turn their cloud drives and spreadsheet data into functional applications. However, the core business of the service is much more centered in client-side applications not exploring the capabilities offered by the different cloud services.

To conclude, must of the platforms already available do not explore in deep, the capabilities and func-
tionalities offered by cloud services such as Google Docs and Google APIs. The service that Google Docs offers, clearly adapts to the solution currently in place in the two organizations mentioned in the introduction, allowing them to continue to use a database schema based on spreadsheets, but in a more centralized and secure way using cloud services.

By considering all those aspects, during the development of the new framework most of the capabilities offered by Google Docs will be explored, while building an interface flexible and auto-explanatory, by following the ten usability heuristics for user interface design [12] and using the evolutionary prototyping model.

In order to meet the solution requirements of the framework, a domain model and use case diagrams were created in order to guide the implementation of the framework.

![Domain Model](image)

Figure 3.1: Domain Model

The domain model illustrated in figure 3.1 contains a set of conceptual classes as also the relationships between all the entities that are part of the problem domain, in order to provide a clear view of the system structure that will be implemented in the cloud.

A detailed description about the entities and respective attributes presented in the domain model can be found in Appendix F.
3.1.1 Front-end Users

The use case diagram presented in figure 3.2 displays the final requirements for the front-end side of the new DB4G framework. The use cases can be defined as the specification of a set of actions performed by one or more actors of the system. For the context of the framework, the use cases define all the actions performed by the actors in the framework, so, each one will be referenced while describing all the interactions made by the different actors.

![Use Case Diagram: Front-end](image)

As displayed in the figure, it is possible to identify three different actors: Super Administrator, Database Analyst and a Developer. In order to use the framework, all the different actors have to authenticate with their Google accounts, however, authenticated actors have different privileges.

Starting by describing the interactions between the Super Administrator and the framework. The Super Administrator corresponds to the owner of the project (i.e. the email account used to create the project). It has full control of the project and framework functionalities and is responsible to add and manage other actors. The term "project" is used in this context to refer the system’s structure that is created in Google Drive.

The use case “Create Project” represents the actions performed by the Super Administrator when creating a project. First, the Super Administrator creates a project and assigns two actors to the project (Database Analyst and Developer). When a project is created, a configuration file is also created in the application server. This configuration file contains all the information regarding the project and all the data created. Every time a file is created, updated or deleted in the database, the configuration file is also updated. All the actors have access to the configuration file in order to keep track of all the changes performed in the project (represented by the three use cases "View Config File").

After a project is created, all the actors have access to the framework. At any time, the Super Admin-
istrator can manage users permissions (represented by the use case "Manage Users") which includes
the functions to add, update and remove new users in the framework. The Super Administration as also
the possibility to backup the configuration file from the local server to the cloud, which is represented by
the use case "Backup Config File".

The Database Analyst can be seen as the person that confirms project requirements by studying
domain model requirements, organize the information in a logical and functional manner in order to build
the database. It has access to the database management area where it can create, read, update and
delete data (i.e. create and edit files according to its type and format) which is represented by the use
cases "Create Data", "Read Data", "Update Data" and "Delete Data" respectively. Meanwhile, when
creating spreadsheets, objects such tables and columns are created by the analyst based on entities
and attributes defined in the domain model. Or simply import files from a local system directly to the
cloud (represented by the use case "Import data"). Furthermore, an option to export data files is also
available (represented by the use case "Export Data"). Finally, the users can logout from the framework
whenever they want while the current status of the project is saved on the cloud based on the real-time
 collaboration capabilities offered by Google Docs.

### 3.1.2 Back-end Users

The use case diagram presented in figure 3.3, presents the current requirements already addressed for
the back-end side of the new framework. The use case diagram only contains one actor, the developer.

As the name says, the developer is the person concerned with all the software development process,
including the research, design, programming, and testing according to software development models.
In this case it is responsible for the maintenance of the whole framework (represented by the use case
"Maintenance of the Framework"), its updates and the extension to new functionalities and new web
applications. The developer can also add new functionalities or features which might include updating
the model and the respective controllers and views (represented by the use case "Add new function-
alities/features to the framework"), but can also remove or simply correct them (represented by the
use case "Remove methods, features or functionalities"). The maintenance of the system includes
not only the front-end and back-end, but also the respective database and the application configuration
and management in Google Cloud Platform (represented by the use case "Application Configuration
and Management in Cloud Platform"). It is important to maintain consistency when adding new
requirements to the system. For that reason, some example files were created, containing the main
aspects of the user interface and the main necessary functions, thus reducing the implementation time
and guiding the developer during the development. When the developer wants to extend the framework
to create a new web application (represented by the use case "Create Web Application") that makes
use of the database created in cloud, the developer has to decide some important aspects, such de-
define the type of authentication (represented by the use case "Define type of authentication") used to
connect the application to the Google Cloud, as also the Google APIs necessary to manipulate data.
Finally the developer has to decide based on the requirements, the best way to define all the application
logic for a specific organization which include the necessary views, controllers and eventually update some methods already provided by the model (represented by the use cases "Create Views", "Create Controllers" and "Update Model" respectively).

3.2 Framework Architecture and Technology

The architecture used in the final solution of DB4G Framework is presented in figure 3.4. The Web browser and the Application Server, represent the main infrastructure communicating with the infrastructure defined by Google Cloud Platform. In order to use cloud services, the framework needs to be registered at Google App Engine environment. Therefore, private information about the registration will be kept in a JSON file in the application server, containing a **clientID** and a **client secret**, used to identify the framework. In this architecture, the user interacts with the Web Browser and performs the login in order to use the framework. Every time a user authenticates in the system, the application server establishes a communication with the Google OAuth Authorization Server that will be responsible for granting access to the user's private data using a Google API. When the user approve the access, an access token is stored in a session variable. One single access token can grant varying degrees of access to multiple APIs. After the user authentication, the respective user interface containing all the information about the project (i.e. the system) is displayed. The Project Area component displays information about the system structure and provides access to the System Management and respective Data Management which include all the data (e.g. file information, users permissions, etc) and respective tasks (e.g.
editing a file, creating a spreadsheet, assign permissions etc). It contains three main components:

- **System Management** - in this component the user manages all the data related to the system environment (e.g. users management, recycle bin management and backup management).

- **Data List Management** - in this component the user defines and manages data presented in the cloud database (e.g. file information, spreadsheet information, users permissions).

- **Task Execution** - in this component the user executes tasks related to information provided in the data list management (e.g. edit file metadata, create file, import file, export file)

---

All functionalities done in the Web Browser by the user will interact with the Application Server through a HTTP/Ajax communication. The Application server is divided in two layers: Application Logic and Business Logic.

Starting by the Business Logic, this layer implements the core functionality of the system and encapsulates the business rules that determine how data can be created, stored, and changed. Conceptually, it can be defined by a set of generic classes or some kind of library and reused across multiple applications or changed in all relevant applications at once (for example, a framework exposing a well-defined API). This layer is extremely useful since allows a developer to build other web applications on the top of it in the future.

The Application Logic is a layer built on top of the business logic and serves to implement all the other elements (e.g. specific use cases) that are specific to an application. So, multiple applications, each
one with different application logic can reuse one business logic which is precisely what is expected in the framework.

Basically, the application logic contains all the sections that are managed by the user which include Users, System and Database Management as also all the tasks performed by the user. Each component depends from the communication with the cloud database as explained by the flow of dependencies. Every time the user performs a task the Application Logic layer send the commands to the layer bellow (i.e. the Business Layer). Then the Business layer communicates with the Google APIs and send the request to the cloud endpoint to update or retrieve data according to the business rule implemented.

It is also important to mentioned that the framework was designed using a MVC (Model-View-Controller) pattern which is divided in three main components: model, view, controller

In this case, the Business Logic is defined in the model, which receive commands from the controller to retrieve and store data in a database. Thus, it directly manages the data contained in the Google Cloud and all the logic and rules of the application. The Application Logic is defined in the Controller, which controls the interaction between the Model and the View. Finally, the View provides the output representation of the information to the user interface.

The technologies used for the concept of this solution are HTML for the front-end (plus Javascript, Jquery, Ajax, Bootstrap and CSS)

For the back-end, it was used an adapted version of MINI PHP framework ¹ that provides a simple and clean structure so future developers don’t have to learn in deep, a sophisticated framework, avoiding lots of time and effort understanding the system. The decision about using PHP and not other languages resides on the fact that, after a careful research about the APIs documentation provided by Google, PHP was the only one that had a powerful documentation specially for Google Sheets API version 3.

For the access and management of the cloud services it is used Google Drive API and Google Sheets API v3, each one directly related to the Google Docs service.

### 3.3 Summary

This chapter describes the problem analysis and the reached final solution design after the study of some related cloud services. Again, the main goal is to build a framework containing a set of tools and features to build information systems for different organizations, allowing them to create their database according to a specific domain model and build their own web application. The next chapter describes in great detail the implementation and final version of the framework.

Chapter 4

Implementation

In order to develop a new framework to build information systems using a public cloud, it was necessary to take full advantage of the technologies chosen in the solution design, always taking into account the consistency and usability of the whole interface as also the extension to other APIs and to a web application for a specific organization.

In Appendix A, the User Manual of the framework provides a brief description and necessary requirements, as also a detailed tutorial how every function of the framework works and how to use them.

This chapter describes in detail the solution implementation and justifies the proposed solution. Note that after analysing the results from the evaluations regarding the first version, some small changes were made to the framework (i.e. relevant improvements and correction of the usability issues found). The new version is described in this chapter.

4.1 Home Page Section

The main pages of the framework provides some essential information about the framework (as shown in figure 4.1). These main pages can be switched through the black navigation bar located on top (by clicking the respective page name). There are in total three main pages: the home page, the about page and the contact page.

The home page shows a simple description of the framework, the about page describes the framework a little more and its functionalities and the contact page provides the contact information. All of these pages follow the same interface presented in figure 4.1 by displaying the main information in the center of the page and follow the same style. Finally, at the top right hand corner of the page a login button is also available.
4.1.1 Login

As previously explained, all the requests made by the framework to the Google APIs must be authorized by an authenticated user using OAuth2.0 protocol. For that reason, in order to access the framework, users must authenticate using their google account credentials.

As mentioned in 2.1.2, before an application can access private data using a Google API, it must obtain an access token that grants the access. In order to that, a variable parameter called scope was set, which controls the set of resources and operations allowed by an access token. During the access token request, the framework sends one or more values in the scope parameter.

So, when the user clicks on the "Login" button, it is redirected to a Google Authentication Server page where his credentials are requested. Some requests require an authentication step where the user logs in with his Google account.

4.1.2 User Consent

After the user logs in, the framework ask the user to grant permissions to the framework to have access to his Google Drive account, management of spreadsheet files and some basic profile information, a process called user consent (see figure 4.2). If the user grants the access, the Google Authorization Server returns an access token that is temporarily stored in the application server using session variables (the access token is destroyed when the user does a logout). Then, the "Login" button is replaced by a "Logout" button. So the user can logout at any time.

Finally, the blue button "Start using DB4G Framework", located at the center of the page allows the user to enter into the project interface.
4.2 Project Management Section

After the login is done successfully, the framework checks if a project is already created (i.e. if there is a configuration file in the application server assigned to a project in a Google Drive account).

If the file exists, the users are redirected to the project area according to a certain role (it will be explained in section 4.2.2). Assuming that there is no configuration file, after the login the user goes directly to the project creation page in order to create a new project and set up the initial requirements for his own system.

4.2.1 Create Project

The project creation page, is where the Super Administrator specify details about the project, including the project name as also the possibility to add more users according to two different type of roles: Database Analyst and Developer.

All the required fields must be filled or warning messages appears and the project is not created. After validate all the required fields, the administrator can create the project by clicking on the "Create Project" button. A "Cancel" button is also available to cancel all the actions performed. When clicking on the "Create Project" button, the framework automatically creates the project structure into the user’s drive account. The structure of the project is composed by a main root folder and three sub-folders:

1. Root: is the main folder of the project created in Google Drive. This folder contains all the system information including users, database and configuration files. Note that, the name assigned to this
folder is the same provided during the creation of the project.

2. Database: is a sub-folder of the project where all the database information is stored and updated.

3. Configuration: is a sub-folder of the project used to backup the configuration file from the application server.

4. Users: is a sub-folder of the project containing a spreadsheet file where all the information regarding the Super Administrator, Database Analysts and Developers are stored.

Note that only the owner of the project has access to the root folder which adds an additional security, since the integrity of the file system structure is not compromised by other users. Once the initial structure is created, the users permissions are added. Like mentioned in the related work, multiple users can work (i.e. can share, edit, view or delete) on the same file in a real-time collaboration. This feature, clearly emphasizes the decision of having three distinct users by the fact that, when a user share a file with multiple users it can control whether they can edit or view a file.

When a new Database Analyst (or Developer) is added, the framework automatically defines permissions to edit or view the sub-folders available in the system. In this case and since they have different purposes, the Database Analyst will have permission to edit the "Database" sub-folder but only have permissions to view the "Users" folder. The developer will have permission to edit the "Database" and "Configuration" sub-folders but only have permission to view the "Users" folder.

4.2.2 Project Area

Once the project is created, the user is redirected to the project area, as showed in 4.3, which displays a specific owner account as example. The project area displays the system structure created by the framework. The three sub-folders are displayed in a table, containing some information about each

![Figure 4.3: Project Area - page section](image)

folder like name, the user name of the file owner, when the folder was created and the last update made in the folder. For each folder, there are two types of functions available: Open and WebLink. It can be said that the functions execute what the name says. The "Open" button opens the folder and goes to the respective folder section. Note that each folder represents a different area in the system. So the
“Database” sub-folder corresponds the Database Management area (explained in section 4.4), while the sub-folders “Users” and “Configuration” corresponds to the Users and Configuration Management areas, respectively. The “WebLink” button redirects the user to the Google Drive website and displays the select folder.

The “Backup Config File” button allows the Super Administrator to backup the configuration file from the application server. This file is automatically stored in the “Configuration” folder which contains a list of backup files. It is also possible to edit the name or delete the project from the cloud.

Like in the main pages, the dark black navigator is presented, containing two buttons (located on the top right hand corner). In this case the users can either go back to the main page (using the “Home Menu” button) or just logout from the system (using the “Logout” button).

On the left side, a vertical bar as also added where it is possible to perform some actions. The first two buttons allow the access to the project and database management areas. The “System Management” button allows the user to switch between the areas that are part of the System Management section. Finally, the “Help” button contains some guides to help the users while they are using the framework.

4.3 System Management

The system management section provides access to the following areas: Users Management, Recycle Bin and Configuration File. Each one can be accessed by the user using the left vertical tab available.

4.3.1 Users Management

The Users Management section displays information about the users of the system. The data is retrieved from a spreadsheet stored in the “Users” folder, and displayed in a table (see figure 4.4). Each user is identified by a unique identifier, name, email address, type of permission and phone.

It is also possible to add or edit the information of each user by clicking on the “Edit” button available. It is also possible to add new users by clicking on the “Add User Button” (the option is only available for the Super Administrator). When clicking on the “Edit” button for a specific user, a new window appears and the necessary fields are displayed.

It is possible to change the name, email address, phone and the type of permission. All the fields must be filled or an alert message appears. After that, the respective data is updated in the database and permissions are updated. Finally, if the permissions are changed the respective user is notified by email.
4.3.2 Configuration File

The configuration file contains information regarding the project structure and each file created by the users. Every time a user logs into the framework, the server automatically uses the information contained in this file to perform calls to the Google API. However, according to the business model implemented by Google, it is possible to create multiple files with the same name which for our system leads to a security problem. Furthermore, according to a Google API limitation, when searching for a specific file's name, the API always return the most recent file (of those with the same name) and don’t make a distinction if the file belongs to a shared folder or if it is a private file.

After research it was found that the only parameter that allows to distinct between files with the same name is the ID (i.e. the file identifier which is unique). To conclude, every time a new file is created, removed or updated, a reference (i.e. a global variable) is created (or updated or removed) with the name and the ID of the specific file (more details can be found in the Administrator Manual available in Appendix B). This file solves not only the security issues described above, but also increases the performance of the framework since it only requests for specific files, using the references presented in this file.

Another advantage of having this file is that, when extending the framework to a web application, developers only have to use those references to performs actions and make calls to the APIs using the functions already provided by the business model implemented.

4.3.3 Recycle Bin

The Recycle Bin area, contains a collection of all the files available in the trash of the user’s drive account. It allows the user to recover files that have been deleted from the project. The inclusion of this functionality is justified by the fact that multiple users can work at the same time in the project and due to specific requirements from Google, only the file’s owner can delete a file. For example, if for some reason the owner accidentally deletes an important file from the project’s folder, instead of permanently delete the file from his google drive account, the framework automatically move the file to the trash. So the user has the ability to recover the file back to the project directory at any time, using the functionalities provided. All the files available in the trash are displayed in a table (see figure 4.5) containing some information regarding each file such the name of the file, the mimeType which is a mechanism used to identify files on the internet according to their extension, the time when it was deleted and options to recover the file or delete the file permanently. It is also possible to recover or delete multiple files.

![Figure 4.5: System Management: Recycle Bin section](image-url)
using the select box option available in the page. When the user wants to recover a file, the framework automatically moves the file back to its previous location. Finally, when a file is moved to the trash, his reference in the configuration file is removed.

4.4 Database Management

The Database Management is the area where all the database information is stored and updated. It displays all the information about files that are stored inside the “Database” folder. Like the pages described above, the left vertical bar is also presented with the rest of the page containing information regarding the files and folders created in the system. The framework supports the following file types: folders, documents, spreadsheets, pdfs, powerpoints and images. At the end of the page is also possible to find the user permissions assigned to the files. The page contains some functions (on the right) to create and upload files, as also to add users and refresh the page. Since that it is possible to have multiple folders in the database, the page also displays the current location where it is possible to navigate between folders. In figure 4.6, the files are displayed in tables containing information of each file which includes the name, the owner of the file, when the file was created, the last time the file was modified as also the email address of the last user that performed changes in the file, small information about the lasts actions and requests performed.

4.4.1 Create File

The framework provides functions to create new folders or tables (i.e. spreadsheets).

The “Add Folder” button allows the user to create a folder. The user must fill the name of the folder, which is validated after the users answers the confirmation message (i.e. the user has the possibility to cancel this action before going further).
The "Create Table" button allows the users to create a new spreadsheet. When the user clicks in the button a new window appears as shown in figure 4.7.

![Create Spreadsheet Dialog]

Figure 4.7: Add new table to the database

In order to create a new table the user must provide a name for the new spreadsheet and for the first worksheet, and define the column headers of the spreadsheet. The columns field available is dynamically incremented as more headers are added to the form. All the required fields must be filled or an alert message appears and the spreadsheet is not created.

When the user clicks on the "Create Spreadsheet" button the framework creates the new spreadsheet. It is also possible to cancel all the actions by clicking in the "Cancel" button.

### 4.4.2 Preview, Edit and Delete File

As mentioned above, there are different functions available for each file represented by the four colored buttons displayed in figure 4.6. The “Open” and “WebLink” buttons were explained section 4.2.2.

The "Edit" button allows to change the file name and the "Remove" button allows the user to remove the file from the project (a Super Administrator authorization is required). Finally, the "WebLink" button redirects the user to the Google Drive Website and displays the selected file.

For the spreadsheet files, there are five functions available: manage content, edit, remove and download. The functions "Edit" and "Remove" have the same functionality used for folders. The "Download" button, allows to download files to a local device (more details about this function are explained in section 4.4.3). The “Management Content” button redirects the user to the Table Management area (explained in detail in section 4.4.4) which allows to perform some actions to the spreadsheet.

For the rest of the files available (i.e. documents, pdfs and images) the four functions mentioned above are also available and have the same purpose. As mentioned in the beginning of this chapter, Google only provides APIs to manipulate the content of spreadsheet files and manage some file metadata. So it is possible for example edit a document’s file metadata but it’s not possible to edit the content using a API that offers full control over the content and appearance of the document.

However, taking into account that limitation, and after exploring some potential solutions for the problem it was decided to integrate the interface available in Google Docs website by embedding the
current document page into the framework like showed in figure 4.8.

![Figure 4.8: Document edition](image)

### 4.4.3 Import and Export Files

As mentioned in 4.4, the framework provides functions to import and export files. The "Import Files" button, allows users to import multiple files according to specific file extensions. At the moment the following extensions are supported: .doc, .docx, .xls, .xlsx, .ppt, .pdf, .jpg and .png. When the user clicks on the "Import Files" a window is opened (see figure 4.9(b)). The files can be imported by choosing a specific location in the computer or using drag-and-drop features. Each file object contains 3 buttons where is also possible to upload only the file, preview the file before submit the request (see figure 4.9(a)) and delete a specific file from the set of files being imported.

![Figure 4.9: Import File - Process Overview](image)

It is also possible to remove all the files from the set or upload them. When the user clicks on the "Upload" button available the framework checks, for every file, if an existing file is already available in the database (returning an error message if a file already exists) as shown in figure 4.9(b). This process is done for each file being imported. The addiction of the import function to the framework is
extremely important since it allows to easily migrate Microsoft Office 365 documents to Google Docs. The framework was designed to automatically convert each Microsoft Office file being imported to a Google Docs format which facilitates the migration of Microsoft Office files from older systems to the cloud.

Finally, the export function which is represented by the "Download" button available for each file, allows the user to download the files. Additionally it also provides the capability to download exported versions of Google Docs (Documents, Spreadsheets, Presentations, etc.) in a format compatible with other office productivity programs such as those offered in Microsoft Office.

However, according to Google some documents might lose some of their original page or text formatting when converted to a Microsoft Office file format.

4.4.4 Table Management

The Table Management is the area where it is possible to manage the content of a specific spreadsheet. When the user clicks in the "Manage Content" button of a specified spreadsheet file it is redirected to the table management area, like showed in figure 4.10 which displays a specific spreadsheet content as example. The page maintains the same aspect and consistency of the pages previously described. It contains four main functions: Back, Refresh, Add User and Add Worksheet. The functions provided by the first three buttons were explained in the previous sections so they will not be explained in this section. The worksheets are displayed in a table (figure 4.10) containing some information about each worksheet regarding the name, last time modified by the user, the headers, the number of lines and the number of columns. For each worksheet there are also three buttons available: Edit, Remove and Weblink. The "Edit" button allows the user to edit the name and the headers of the worksheet. The "Remove" button allows to delete permanently the worksheet (note that it is not possible to send a worksheet to the trash). Finally the "Weblink" button redirects the user to the Google Drive website and displays the current worksheet.

The "Add Worksheet" button allows the user to add a new worksheet to the current project. The user must choose a name for the new worksheet and define the headers, using the same process used in
spreadsheets. After fill the necessary fields, the user answers the confirmation message and creates the new worksheet (recall that the user has the possibility to cancel all the actions before going further). After that, the user is redirected back to the table management area, which is updated with the new information.

4.4.5 User Permissions

The User Permissions section is located in the end of the page of every area of the framework. It displays information about each user assigned to files and folders in the project. In this case, each file (or folder) in the project is assigned to the users of the framework (i.e. the Super Administrator, the Database Analysts and Developers) defined when the project was created, so each file will always have those users.

However, the Super Administrator may decide to include other users only in a particular file or folder inside the project. This problem justifies the decision of having this section in the framework, since it provides methods to manage the user permissions, which in this case consists in functions to edit user permissions (i.e. editor or writer) or just remove user access to a specific file.

4.5 Framework vs Google APIs Limitations

Several challenges were found during the implementation of the framework. Some of them are related to the integration between the framework and the Google APIs.

This section describes the limitations and challenges encountered during the development and the set of solutions adopted to overcome those limitations.

4.5.1 File Ownership

As mentioned above, every file created in Google Drive can only be deleted by his owner. Which means that, if the owner deletes a shared folder, document or even a spreadsheet, it will be completely removed from Drive for all collaborators, and they will no longer have access to the file.

Since that our framework is using a shared project in Drive with other collaborators, it was extremely important to prevent other collaborators from delete permanently the files from the project without the permission of the Super Administrator (i.e the project’s owner).

The initial problem was that, every time a Database Analyst created a file, the API automatically assigned him as the owner of the file, so only the Database Analyst could delete permanently the file. This causes two problems in our project. The first one, was that, if the Database Analyst deleted a file, the file would be completely removed from the project, and the other collaborators (including the Super Administrator) will not be able to access to the the file anymore. The second problem found was related to the user permissions management. Every time a Super Administrator wanted to remove a user from the project, if that user had owner permissions assigned to some files, the API returned an error saying that it was impossible to remove users from a shared folder with files owned by that user.
Fortunately, Google provides a feature that allows a certain user to transfer the ownership of files and folders to other user. Basically when a user transfer the ownership of a file, it cannot delete the file anymore but it still continues to edit the file, unless their permissions are changed by the new owner. However, due to limitations of Drive API, when a user transfers the ownership of a shared folder to other user, the ownership of the files inside the folder remains the same, which will cause the second problem mentioned.

After a careful research about the possible solutions for the problem, it was found that it could be possible to transfer the ownership of the file at the moment of his creation. At the same time it was found that, the other users could request permissions to delete a certain file to the Super Administrator.

The solutions adopted are described bellow:

- Every time a new file is created by a user (e.g. Database Analyst), the framework automatically transfers the ownership of the file to the Super Administrator.

- Every time a Database Analyst wants to delete a file from the database, the framework automatically uses file description tag's to request the permission to the Super Administrator, who decide to approve and remove the file (note that the file will be moved to the trash), or reject the request.

### 4.5.2 Quota Limits

Google Cloud Platform is used by millions of sites around the world. In order to protect the system from receiving more data that it can handle, and to ensure an equitable distribution of the system resources, Google imposes limits and quotas on the API requests. Quotas for example project the community of users by preventing unforeseen spikes in usage. Excessive requests from an API might result from a harmless typo, or might result from an inefficiently designed system that makes needless API calls. Regardless of the cause, blocking traffic from a specific source once it reaches a certain level is necessary for the overall performance of the system. It ensures that one developer’s actions cannot negatively impact the larger community.

Specific policies vary depending on resource availability, user profile, service usage history, and other factors, and are subject to change without notice.

Google Cloud Platform also offers free quotas that provide limited access to projects that consist only of exploring the Google Cloud Platform without having a paid account. However, for paid accounts (i.e. assigned to subscription plans), it is possible to request quota adjustments in Google Platform console, if a notable upcoming increase in usage is expected.

In the unlikely event that if an API request fails, the framework receive an HTTP status code response. A status code of 403 has error information about incorrect input and an HTTP status code of 503 has error information indicating which API quotas have been exceeded. These responses allow to detect these errors and take appropriate actions.

Taking into account all these aspects, the following two tables describe respectively the Google Drive API and Google Sheets API limits and quotas assigned by the Google Cloud Platform to the framework.
<table>
<thead>
<tr>
<th>Quota Name</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries per day</td>
<td>1,000,000,000</td>
</tr>
<tr>
<td>Queries per 100 seconds per user</td>
<td>1,000</td>
</tr>
<tr>
<td>Queries per 100 seconds</td>
<td>10,000</td>
</tr>
</tbody>
</table>

Table 4.1: Google Drive API - quota limits

<table>
<thead>
<tr>
<th>Quota Name</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read requests per 100 seconds</td>
<td>500</td>
</tr>
<tr>
<td>Read requests per 100 seconds per user</td>
<td>100</td>
</tr>
<tr>
<td>Read requests per day</td>
<td>Unlimited</td>
</tr>
</tbody>
</table>

Table 4.2: Google Sheets API - read quota limits

<table>
<thead>
<tr>
<th>Quota Name</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write requests per 100 seconds</td>
<td>500</td>
</tr>
<tr>
<td>Write requests per 100 seconds per user</td>
<td>100</td>
</tr>
<tr>
<td>Write requests per day</td>
<td>Unlimited</td>
</tr>
</tbody>
</table>

Table 4.3: Google Sheets API - write quota limits

The information about the daily usage and quota consumption of App Engine resources allocated to a project can be found Google Cloud Platform Console.

### 4.5.3 Performance

Another challenge found during the development of the framework is related with the API performance and the necessary time to retrieved data from the cloud. The Google Sheets API for example, can be pretty slow depending on many factors including the speed of the internet connection to Google servers, bandwidth, higher latency which causes traffic congestion, etc...

In order to overcome those problems, some techniques were adopted in order to improve the performance of the framework. The first technique adopted to improve the performance of each API calls is by requesting only the portion of the data necessary. This let the framework avoid transferring, parsing and store unneeded data, so it can use resources including network, CPU and memory more efficiently.\(^1\)

On the other hand, each HTTP connection that the framework makes, results in a certain amount of overhead. To overcome this problem, both Google Drive and Sheets APIs supports batching operations, by pipelining several API calls into a single HTTP request.

In short, the implementation of these techniques, resulted in a better utilization of the network and speed up the framework by increasing the network throughput and reducing latency.

The batching operations were applied in the following situations:

- Retrieving metadata for a large number of files.
- Updating file metadata.
- Changing permissions for a large number of files, such as adding a new user or transfer ownership.
- Read and write requests over spreadsheets.

\(^1\)Information available in: https://cloud.google.com/storage/docs/json_api/v1/how-tos/performance
4.5.4 Work Offline

Another interesting feature about Google Drive service, is that it allows users to work with files when they’re are not connected to a Wi-Fi or mobile network. However, the biggest issue to be aware of when using offline synchronization in Google Docs, is the potential for a file to be edited simultaneously by different users (which is exactly what happens in our framework).

For example, if someone (e.g a Super Administrator) edits a document online while other (e.g. a Database Analyst) is editing it offline the two sets of changes will be merged when the new version gets re-uploaded. In fact, the rename option is unavailable in offline mode so it is not possible to save a separate version of the document.

So, while using the framework, it is not recommended to use offline mode for files that are being collaborated on by multiple users (or at least make them aware, or create a duplicate while the user is online, exclusively for his own use). It is possible to see who has edited a file and, if necessary, roll back to an older version by using the history revision tool available in Google Drive website.

4.5.5 Backup project

One of the biggest limitations of Google APIs, is that it does not allow to download folders and for that reason it is not possible to include a functionality to backup the entire project from the framework to a local device.

However, Google provides a tool that allows to export and download data from the Google Drive, by creating an archive to keep all the records or to be used in another service. Google Takeout is a tool that offers the ability to create an archive containing all the data contained in Google Drive, which in this case will be useful to save a copy of the project. Moreover, when the archive is created, it is possible to send the archive by email or even add the archive to other software as a service providers such Dropbox or Microsoft One Drive, so the owner of the project can always save a copy of the entire project to one of those locations.

4.6 Summary

In this chapter details regarding the specific features and functionalities were described, as also, the description of every section of the framework. Each section display many features, but the main ones are: describing and showing information about the project; displaying and managing users; presenting and managing all the files in the database. All of these sections make the actual framework and its respective interface. Furthermore, the User Manual of the framework is available in Appendix A containing a brief description and necessary requirements, but also presents a detailed tutorial explaining how every function provided by the framework works. The next chapter describes the framework extension to other functionalities and web applications, from planning all the requirements to the steps required to add it successfully to the framework, while respecting the framework’s interface and main functionalities.
Chapter 5

Framework Extension

In the last chapter, the implementation of the framework, as also the respective interface and main functions were presented in detail.

However, it was not mentioned how the framework can be extended to include new functions and how it can be extended to build a new web application for a specific organization, and the benefits of doing so without affecting the functionalities already implemented.

This chapter describes how the extension of DB4G framework is done by the developers, while maintaining the same interface and main functions.

The Administrative Manual is available in Appendix B which describes all the necessary steps to install the framework in a server, as also the necessary steps to configure the framework in Google Cloud Platform, and finally the necessary steps to build the new web application (i.e. add new controllers and methods successfully). It also contains some example files to guide the user when adding new functions to the framework.

5.1 Planning and Requirements

First of all, the developer should define a list of requirements before start to do any coding or design. The list should clarify all the necessary functions, capabilities and characteristics related to the application logic and the plans for creating it. The list may include business requirements, which define the objectives and what problems that the stakeholder intends to solve with the application as also the functional requirements which provide details about the application’s behavior.

After the requirements are defined, the developer needs to decide important details: since the web application will be connected to a cloud database already created using the framework interface, he needs to decide what type of authentication should be used to connect the application to the Google Cloud. Then the developer must decide what are the Google APIs that will be included to manipulate data and if new APIs or new functions must be added to the framework. Finally the developer has to decide based on the requirements, how many controllers and views will be needed to define all the application logic layer.
If the developer decides to use a different API that is not yet available in the framework, a new class containing new functions should be added to the business logic layer (explained in the Administrator Manual). To help and guide developers regarding the front and back-end aspects, it was created 8 example files (described in detail in the Administrator Manual), that shows how to add new functions and new APIs to the framework, as also how to add the new functions to the main interface. In order to make the extension of the framework much easier, the files contain the same interface’s style sheet used in the framework interface. The developer is responsible for adapt the interface according to all the changes and additions in the back-end and made use of the configuration file available, which gives access to the information contained in the database. Finally the developer should plan all the software development cycle using an appropriate software development method.

5.2 Implementation

After plan and establish the software requirements, the implementation can now go forward. First the back-end, the developer needs to decide the type of authentication used to authenticate users into the web application. Currently, the framework uses an authentication process based on OAuth2.0 client credentials (as explained in section 2.1.2).

However, another type of authentication is allowed in Google Cloud Platform. After a careful re-search, it was found that Google OAuth2.0 system supports server-to-server interactions using a service account, which is an account that belongs to the web application instead of to an individual user. This type of account is typically used, when a web application uses Google APIs to work with its own data rather than a user’s data, so user’s aren’t directly involved. Moreover, the application call the APIs on behalf of the service account instead of the user. After creating a service account in the Google API Console, the developer only have to run a script already available (explained in more detail in the Administrator Manual) in order to create the access token that will grant access to the application data.

However, it is important to mention that an additional authorization, authentication and encryption protocol must be defined by the developer in order to control the access to the web application by users.

5.2.1 Adding a new Google API

If the developer decides to use a different API library that is not yet available in the framework it must add a new class to the business logic layer (i.e. to the model).

Every new class that is created must have a required function to get the client authentication access token. It is also the class where the developer implement the necessary functions to connect and use a certain API library.

1 Information provided by Google Identity Platform
5.2.2 Adding new controllers and views

Since the framework have some controllers already implemented, in order to avoid mistakes (e.g. deleting a file that should not be deleted) and make the structure much more clean and organized, the framework provides a folder named as "system". So, in order to add new controllers to create the new web application, the developer must define them inside that folder.

When the back-end is completed, the developer should create the views that represents the presentation layer, inside a folder also named as "system", in order to avoiding compromising the views that are part of the framework interface.

5.2.3 Working with data

After the example files have been added to the framework, the developer now adapts and makes the necessary changes according to the list of requirements, and implements the functions needed to call the application server functions (like get data, update data, delete, etc...). Note that, the files only contain a particular example just to guide the developer while adapting to the framework. Regarding the interface of each section, the developer has to follow the file's structure and change the parts that are need to be modified. For example, the left navigator bar can be used to add new sections that the new web application will have. Depending on the functionality, the application can have one or more sections (all of them must follow the same structure) and the developer is responsible for adapt each section by following the available examples. For the manipulation of the information stored in the database the files also contain examples of functions to create, read, update and delete data. The style sheet provides the same pattern to every section of the web application, so the consistency between the framework's interface and the web application interface (the developers should be careful not to stay out of the main interface).

5.3 Summary

This chapter described the necessary requirements and planning that the developers need to do for the extension of new functions to the framework, as also the extension to a web application. After the planning is done, the implementations can go forward and the developers start in the back-end by adding the necessary functions and APIs to interact with the data. Then, the example files are used to help and guide the developers regarding the front-end. These files make the developers follow the main interface and patterns. Furthermore, they demonstrate examples of how the functions are done and APIs are added, plus the available presentations of the data which the developers only need to adapt according to the application logic. The available Administrative Manual describes all the necessary steps in detail and the respective structure (using actual examples) in order to deploy a web application successfully which can be read in Appendix B. The next chapter describes the user evaluation stage, from planning to the result analysis and respective conclusions.
Chapter 6

Evaluation

In the previous chapter, the extension of the framework to new web applications and functions was presented. In order to evaluate the framework solution and see if the main objectives were accomplished, a series of evaluations were conducted with two types of users: the expert and future users. The evaluations were made in order to evaluate the framework regarding the usability and user experience using two evaluation methods: System Usability Scale (SUS) and User Experience Questionnaire (UEQ).

The main goal of the evaluation was to obtain the first impressions of the users regarding the framework functionality as also possible issues detected while interacting with the framework, and finally responding to a survey regarding their experience and new ideas to improve the framework.

6.1 Application Scenarios

As mentioned in the introduction, DB4G framework is currently being applied in other two master thesis, more precisely, in the development of a solution for two Portuguese non-profit organizations: Animalife and APOGEP.

Both organizations share the same problem by using outdated systems based on Microsoft Excel with several problems compromising the automatization, management and performance of each organization.

In order to evaluate all the framework, from the point of view of his extension to other web-applications but also regarding the usability and user experience, two different approaches were applied. The first one was the migration of the Animalife current system into the framework. Unfortunately, the Animalife project was delayed so only a part of the system was migrated. The idea was to show the potential of the framework from the point of view of his extension as also in the migration of older systems.

However, the colleague (Tiago Neto) that is currently developing a solution for Animalife also participated in the evaluation of the framework main functionalities as one of the users. The migration of Animalife system consisted in the migration of all the current database files (i.e. all the spreadsheets) into the project’s database using the import functions provided by the framework. Then it was only necessary to develop all the application logic related (i.e. all the controllers and views related to the user
For the case of APOGEP, since that it is a solution that will be covered by another colleague (Jorge Almeida) using DB4G framework, he also participated in the evaluation of the framework.

### 6.2 Planning

Before proceeding with the evaluation process, a careful planning was made in order to define all the tasks that will be performed by the participants, as also the amount of time that will be spend on each participant. So, in order to avoid mistakes and have an effective time-management, a timeline containing the sequence of events of the evaluation was prepared and was divided as follows:

- **Prepare the evaluation and provide all the necessary documents to the participant**: 3 minutes;
- **Explain the evaluation and the main goals to the participant**: 5 to 10 minutes;
- **Start executing the tasks**: 25 to 35 minutes in total;
- **Short interview with the participant about the initial impressions**: 5 minutes;
- **Ask the participant to fill a survey regarding the usability, user experience and person opinion about the framework**: 5 minutes;

The estimated time to complete the tasks was between 25 and 35 minutes, and the total estimated time for the evaluation was approximately 1 hour.

### 6.2.1 Setup

Regarding the setup for the evaluation, the participants only needed a computer with internet access and one of the following recommended browsers: Chrome or Safari.

A flash drive was also provided containing a folder with some files to be used on specific tasks, a Guidelines document and a User Manual to be read before starting the tasks.

The User Manual (presented in Appendix A) provides a detailed explanation about the framework and the respective functions. The Guidelines document (presented in Appendix C) contains a brief description regarding the framework, the required configuration and all the tasks to be performed.

Before starting the tasks the participants explored the User Manual and the "Import Files" folder available in the flash drive. The folder contains 3 files with different formats and will be used to perform specific actions during the evaluation. A few number of participants opened the User Manual just to had a quick look for preparation, although it was not mandatory to do so. After that, the users opened the Guidelines document and the first version of the framework and started performing the tasks.
6.2.2 Task Definition

The proposed tasks consisted in exploring the first version of the framework and using all its available functions and receive the respective feedback in the end. The participants followed a specific order to execute the tasks which started with simple tasks like creating the project, with the level of difficulty increasing when going to the next task. The idea is to obtain a feedback about the functionalities of the framework as also to show the power of Google App engine and Google Docs. The last task consisted in answering a survey regarding the satisfaction and user experience of the participants when using the framework for the first time, as also the usability of the framework and new ideas and last comments to improve the framework. The survey is presented in Appendix D.

6.2.3 Participants

The type of participants chosen to carry out the usability tests was made always taking into account the specificity and characteristics of the framework. The choice of the number of users was made following the recommendations of Jakob Nielsen which indicates that a number ranging from 3 to 5 users is sufficient to detect about 80% of usability problems [13]. It also states that with more than 5 users most of the errors found are redundant (see figure 6.1). However, in some cases it could be necessary to test two or more user groups according to their own specificities. Nielsen recommends using 3 to 4 users for each group if the test is performed by two groups and only 3 users for each group if there are 3 or more groups. In this case, it should be noted that this method is expensive and time-consuming.

Figure 6.1: Comparison between the number of errors found in a website and the number of participants that made the test (Nielsen, 1993)

So, taking into account the application scenarios described and the main goal of the framework, it seems consensual the use of two types of participants in the evaluation of the usability: the experts and the future users.

For the context of the framework, the experts are only two users (Tiago and Jorge) who have deep knowledge about all the aspects of the project (which includes all the requirements, as also all the aspects related to the back-end and technologies used).

The future users can be defined as the target audience of a particular site or product. In this case the evaluation is performed with an observer who collects and annotate all the difficulties of the users during the usability test.
To conclude, in total 7 volunteers participated in the evaluation with all of them having some knowledge about Google Drive and database management (some more expert than others). The evaluations were made in person in IST and INESC-ID (Taguspark), which become extremely useful to obtain a closer observation of the participant’s actions, note the eventual mistakes, record the time taken to complete the tasks and finally obtain a trustful and valid feedback.

6.2.4 Evaluation Methods

As mentioned in the beginning of this chapter, in order to evaluate the usability and user experience two different methods were used, which are described bellow:

System Usability Scale (SUS)

To conduct the usability evaluation it was used the System Usability Scale (SUS) [16], which is a questionnaire provided to the users for measuring the perceived ease of use of software, hardware, cell phones and websites. It provides 10 questions that are shown in table 6.1.

<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I think that I would like to use this website frequently.</td>
</tr>
<tr>
<td>2</td>
<td>I found this website unnecessarily complex.</td>
</tr>
<tr>
<td>3</td>
<td>I thought this website was easy to use.</td>
</tr>
<tr>
<td>4</td>
<td>I think I would need assistance to be able to use this website.</td>
</tr>
<tr>
<td>5</td>
<td>I thought the various functions in this website were well integrated.</td>
</tr>
<tr>
<td>6</td>
<td>I thought there was too inconsistency in this website.</td>
</tr>
<tr>
<td>7</td>
<td>I would imagine that most people would learn to use this website very quickly.</td>
</tr>
<tr>
<td>8</td>
<td>I found this website very cumbersome/awkward to use.</td>
</tr>
<tr>
<td>9</td>
<td>I felt very confident using this website.</td>
</tr>
<tr>
<td>10</td>
<td>I needed to learn a lot of things before I could get going with this website.</td>
</tr>
</tbody>
</table>

Table 6.1: System Usability Scale

It also provides to participants, response options that fall between “Strongly disagree” to “Strongly agree”, in a scale numbered from 1 to 5;

User Experience Questionnaire (UEQ)

Regarding the user experience, it was used the User Experience Questionnaire (UEQ), which is a data analytical approach that evaluates the interaction with a product. As they are different from the SUS surveys, every UEQ question belongs to some evaluation measurements, defined in 6 scales which are the following [17]:

- **Attractiveness**: how attractive is the product in an overall impression. Do users like or dislike the product?
- **Perspicuity**: how easy the product is to use. Is it easy to get familiar with the product?
- **Efficiency**: how fast the user solves the tasks. Does the user interface looks organized?
- **Dependability**: Does the user feel in control of the interaction? Is the interaction with the product secure and predictable?

- **Stimulation**: how well the product motivates the user. Does the user feel motivated to further use the product?

- **Novelty**: how innovative and creative the product is. Does the product grab users attention?

### 6.2.5 Survey

As mentioned above, after completing all the tasks, each user answered the respective survey questions which were divided in three parts: **Usability**, **User Experience** and **Personal Opinion**.

Usability, in general, is influenced by the user, the destination and the situation at the time of use. Different users can generate different usability assessment. Moreover, the ISO 9241-11 states that the usability is defined as *"the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"*[15].

To evaluate the usability it was followed five dimensions of usability (5Es) which are the following:

- **Effectiveness**: the completeness and accuracy with which users achieve specified goals;

- **Efficiency**: analyzes the users ability to quickly perform task once they have learned the design;

- **Engaging**: how well the interface draws the user into the interaction and how pleasant and satisfying it is to use;

- **Error tolerance**: tracking how many errors do users make, how severe are these errors, and how easily the product can help them recovering from the errors;

- **Easy to learn**: how easy it is for a user to accomplish basic tasks the first time they interact with the product and how well the product supports both initial orientation and deeper learning;

User experience is defined as a person's perception and response resulting from the use or anticipate the use of a product, system or service. In fact, according to the ISO 9241-210 definition, user experience *"includes all the emotions of users, namely beliefs, preferences, perceptions, physical and psychological responses, behaviors and achievements that occurred before, during, and after use"*[14].

Taking into account that usability and user experience can be similar when measured, it was necessary to plan carefully the type of questions used in the survey in order to have positive answers and results. Starting by the usability, the dimensions were evaluated by the participants and registered while observing them performing the tasks. The efficiency dimension was measured by calculating the average time that participants took to complete all the tasks. The effectiveness and error tolerance dimensions were determined by registering the number of mistakes made by the participants and see if they can recover from the errors and complete all the tasks. For the engaging and easy to learn dimensions, the participants answered to the four questions presented in the survey, by ranking the level of
difficult using a scale from "D) Very Difficult" to "A) Very Easy" of each function of the framework as also the usefulness of specific functionalities using a scale from "1) Should not be available" to "5) Extremely Useful" and finally the last two questions were used to determine the user's expectations and satisfaction regarding the whole experience.

To evaluate the user experience, instead of using the questions provided in the UEQ scales, four questions provided in the SUS table were selected and used to answer to the 4 scales provided in the UEQ which are: Stimulation, Perspicuity, Efficiency and Dependability. For the Novelty and Attractiveness scales, it was created specific questions using the same format presented in the UEQ for these scales. Finally, it was created four questions to obtain the user's personal opinion regarding positive and negative aspects about the whole experience, as also new ideas and last comments to improve the framework. To conclude, the survey had a total of 15 questions divided in: 4 questions regarding the usability, 7 questions regarding user experience and the last 4 regarding the personal opinion) and can be found in Appendix D.

6.3 Results and Discussion

After performing the 7 evaluations, the respective analysis and discussion were made with the obtained results regarding the first version of the framework. However, some users were not comfortable with the framework, since they did not know much about Google Docs service, and in consequence some results were influenced because of it. The results from the survey can be found in Appendix E.

6.3.1 Usability

As previously mentioned, in order to measure the efficiency it was registered the time that the participants took to accomplish some tasks, as also the errors and mistakes during specific tasks and their ability to recover from those mistake. In addiction, measures of central tendency were measured for statistical purposes (i.e. the medians and modes of each question). The average time that participants took to execute all the tasks was 28 minutes which indicates that most users performed the tasks inside the expected duration (users with experience in Google Docs or users who have previously read the manual, finished the tasks faster than other).

Overall, the participants classified most of the available functions very easy to accomplish. However, a few number with less knowledge about Google Drive, had some difficult to understand the collaboration in real time between the different accounts and their relationship with the users management area. They also have some difficult the first time they created a table and his relationship with the Google Sheets. The results are presented in tables 6.2 and E.1.

Regarding the behavior and the utility of the framework, the participants revealed a good understanding of what was happening in the framework, as shown in tables 6.3 and E.2.

However, functions regarding the existence of a configuration file and having a recycle area and the functionality of recovering files were pointed by the users as the least useful when compared to other
Table 6.2: Results from question 1 (first table in Appendix E): medians and modes.

<table>
<thead>
<tr>
<th>Quest.</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
<th>1.4</th>
<th>1.5</th>
<th>1.6</th>
<th>1.7</th>
<th>1.8</th>
<th>1.9</th>
<th>1.10</th>
<th>1.11</th>
<th>1.12</th>
<th>1.13</th>
<th>1.14</th>
<th>1.15</th>
<th>1.16</th>
<th>1.17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Med.</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Mod.</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

* Quest. = Question, Med. = Median, Mod. = Mode.

* From question 1.1 to 1.17: D) Very Difficult, C) Difficult, B) Easy, A) Very Easy.

Functionalities. The rest of the functionalities, which include the import and export functions and the document edition tool were ranked as extremely useful.

Table 6.3: Results from questions 2, 3 and 4 (second table in Appendix E): average, medians and modes

<table>
<thead>
<tr>
<th>Quest.</th>
<th>2.1</th>
<th>2.2</th>
<th>2.3</th>
<th>2.4</th>
<th>2.5</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aver.</td>
<td>4.28</td>
<td>4.71</td>
<td>4.71</td>
<td>4.71</td>
<td>4.42</td>
<td>4.2</td>
<td>4.4</td>
</tr>
<tr>
<td>Med.</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Mod.</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

* Aver. = Average, Med. = Median, Mod. = Mode

* Questions 2.1 to 2.4: 1) Should not be available and 5) Extremely useful.

* Question 3: On a scale of 1 to 5, 1) Strongly disagree and 5) Strongly agree.

* Question 4: On a scale of 1 to 5, 1) Very unsatisfied and 5) Very satisfied.

Overall, the majority of users completed the tasks successfully without any mistakes or eventual errors, but the ones who committed, in the end managed to recover from the mistakes. In general, it can be concluded that the framework’s usability and utility is good taking into account the presented results.

### 6.3.2 Usability Issues

However, despite the positive results some usability issues were registered as a result of the mistakes made by the users during the tasks. The first one spotted was when the users needed to navigate between the project area and the database management. They had difficult to understand that each folder displayed in the project area, represented a different area in the system. The tendency was to click on the “System Management” button, which is justified by the fact that, normally people tend to visualize from left to right, in consequence they saw the button first and had the tendency to click right away before seeing the rest. A way to improve this was to have two buttons that gives access to the project area and database management. The final issue observed was when users tried to import the files, some users did not see or understood that each file had 3 buttons available to remove, upload or preview. In consequence, the respective users lost time in order to find it and some users required assistance. These issues were considered important and relevant to improve the framework’s usability. So the framework was updated and the second version was released.

### 6.3.3 User Experience

Considering the tables 6.4 and E.3, it is possible to see that the user experience was good based on the average results (with values between 4 and 4.7). In general, users considered that the framework was easy to use and the interface was user-friendly and efficient. Moreover, they considered that the majority of the functions were well integrated and also felt very confident when using the framework (even the
<table>
<thead>
<tr>
<th>Quest.</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>4</td>
<td>4.1</td>
<td>4.6</td>
<td>4.3</td>
<td>4.6</td>
<td>4.3</td>
<td>4.7</td>
</tr>
<tr>
<td>Med.</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Mod.</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4.5</td>
<td>5</td>
</tr>
</tbody>
</table>

*From questions 5 to 11: on a scale of 1 to 5, 1) Strongly disagree and 5) Strongly agree.*

Table 6.4: Results from question 5 to 11 (third table in Appendix E): average, medians and modes

users who did not understand the Google Docs service). The participants also stated that the framework is innovative and creative, with the majority answering very positively.

However, the following two questions regarding if the users would like to use the framework if they have a organization in the future and if they thought it was easy to use and get familiar with the framework were the least positive. This is explained by the fact, that the majority of the participants would like to use the framework in the future, however, some of them were not interested in using Google Docs or having organization which leaded to mixed results and thus the presented results.

In general, the 4 scales provided by the UEQ (i.e Efficiency, Dependability, Stimulation and Novelty) were archived with a positive note, but the Attractiveness and Perspicuity were the least positive. These two depend mostly of the participants objectives and knowledge about Google Docs. However, it should be considered in the future to improve these aspects in order to attract more users regardless of their experiences and knowledge.

### 6.3.4 Participants Opinions and Observations

The participants also provided their feedback and opinions. Like mentioned previously, users with good knowledge of Google Docs services stated the positivity of having a framework flexible, simple and easy to use. Furthermore, users revealed interest in having more available functions and integration with other APIs in the future.

Many suggestions for new improvements were received which include: the possibility to create multiple tables from a domain model, incorporate other existing APIs, the possibility to create graphs and assign data types to the attributes, and finally the possibility to import or export other types of formats such Zip. Those suggestions can be considered for future work in order to improve the framework and increase his potential and future growth.

### 6.4 Summary

In this chapter, the evaluation of the developed solution was presented. The participants completed the tasks given and answered a survey regarding their experience plus personal opinions and suggestions to improve the framework. Despite having noticed some usability issues in general the usability and user experience were satisfactory which indicated that the developed solution achieved its main objectives.

So the DB4G framework was updated to a second version according to the suggestions stated in section 6.3.4. The next chapter will finalize this dissertation by summing up what was described and state what can be done in the future.
Chapter 7

Conclusions and Future Work

In the last years cloud computing has revolutionized the way organizations use the computers and the internet. Most of the software as services offered by popular cloud providers, allow organizations to migrate their information systems to the cloud based on subscription-plans regarding the resources they need. However, a few number of organizations are still unable to support the costs associated.

On the other hand, most of systems still in use reside exclusively in Microsoft Excel, with the information spread over multiple documents without any automatization leading to long and exhaustive operations.

7.1 Contributions

In order to overcome all these problems, a set of software as a service solutions were carefully analyzed in order to understand which type of cloud service should be used as cloud-based platform in our solution. One of the biggest challenges of the research was to find a service where our solution could be deployed with no costs associated. After a careful research it was found that only Google Docs satisfies that condition. The service that Google Docs offers, clearly adapts to the solution currently in place in the two non-profit organizations mentioned in the introduction, allowing them to continue to use a database schema based on spreadsheets, in a more centralized and secure way using cloud services. Furthermore, the different services were analyzed in order to find disadvantages and some potential capabilities that may be interesting to have in the new solution.

By considering all those aspects, the designed and developed solution was a new web-based framework that explores most of the capabilities and functions offered by Google Docs, developed in a way that it can be adapted to any type of organization by offering a set of generic tools to migrate the actual systems, as also tools to easily create a physical database based on the domain model regarding a specific system. Furthermore, it provides diverse and useful functionalities (like importing and exporting files, create tables, user management, etc...). Finally it is possible to extend the framework to a new web application fully integrated with the information system implemented in the cloud. In order to do that, the developers only need to follow the steps written in the Administrative Manual.
Analysing the results of the evaluations made for the first version of the framework with users revealed that the framework was efficient and all functions were well integrated. Some results regarding attractiveness and perspicuity were the least positive, as well as, the occurrence of some usability issues during the tasks. But in general, the participants were satisfied with the whole experience and framework. The evaluation of the framework also revealed that participants were able to use all the available functions with positive results regarding the usability features and user experience. Moreover, it was considered important to correct the usability issues found in the evaluation, in order to improve the framework. So the required changes and updates were made and the second version of the DB4G Framework was released, plus the User Manual that was also updated with the new version.

In conclusion, it can be argued that Cloud Computing can help organizations to increase their business value by developing robust and reliable applications using resources that can be dynamically adjusted. Moreover, Cloud Computing provides a lot of benefits which include:

- Significant Cost Reduction
- Quality of Service
- Scalability
- Rapid Deployment
- File Hosting and Accessibility
- Security of Data and Information

### 7.2 Future Work

After measuring the results of the evaluations and analyzing all the received suggestions, it was concluded that some suggestions are considered to be interesting and are good ideas to implement into the framework. These ideas might improve the framework in general and increase the variety of available functions. Like the possibility to create multiple tables from a domain model, importing zip files or export the project folder (currently those actions are not possible to perform due to API limitations). Finally, incorporate other existing frameworks that allows to create a domain model and then convert to the database automatically.

Another aspect that is important to analyze for future work is related with the spreadsheets manipulation. Currently the framework manipulates the spreadsheet data using Google Sheets API v3 (at the moment of the development it was the most stable version). However, Google states that the Google Sheets API v4 is now stable. So, as a personal note it is recommended to migrate the framework to Google Sheets API v4. However, the new version is JSON based so the structures are not backward-compatible with the XML structures used in v3. A careful migration should be done using the migration guide provided by Google. To conclude, the new version of Sheets API will increase the framework functionality by adding a substantial amount of functionality (such direct cell read and write, conditional formatting, updating sheet properties, etc...) that is not possible in the current version (i.e. v3).
Bibliography


Appendix A

User Manual

INSTITUTO SUPERIOR TÉCNICO

VERSION 2.0

DB4G Framework - User Manual

Author
DIOGO SERAFIM

Supervisor
PROF. JOSÉ BORBINHA
Contents

1 Introduction 3
   1.1 Scope and Purpose ............................................. 3
   1.2 Process Overview ............................................ 3

2 Getting Started 4
   2.1 Interface Overview ......................................... 4
   2.2 Login/Logout .................................................. 4

3 Create Project 5
   3.1 Project Area .................................................. 5

4 Database Management 6
   4.1 Create Folder ................................................ 6
   4.2 Create Table .................................................. 7
   4.3 File Management ............................................. 7
      4.3.1 Open ..................................................... 7
      4.3.2 Preview .................................................. 8
      4.3.3 Edit ..................................................... 8
      4.3.4 Delete .................................................. 8
   4.4 Table Management ........................................... 8
      4.4.1 Add worksheet ......................................... 9
      4.4.2 Edit worksheet ....................................... 9
      4.4.3 Delete worksheet .................................... 10
   4.5 Import Files ................................................ 10
   4.6 Export Files ................................................ 11

5 Users Management 11
   5.1 Interface Overview ........................................ 11
   5.2 Add new user ............................................... 11
   5.3 Edit user .................................................... 12

6 Configuration File 12

7 Recycle Bin 12
   7.1 Restore or Delete Files .................................... 13
1 Introduction

1.1 Scope and Purpose

DB4G Framework is a web-based integrated framework that allows users and developers to build and maintain information systems using the public cloud (more specifically Google Cloud Platform) without any costs associated. The framework provides usability and flexibility for a better user experience, as also the extension of the framework to new functionalities at anytime. In addition, it provides management functionalities for users to create and manage their own systems as also functionalities for a faster and easy usage of the framework.

The main goal of this manual is to guide and help users regarding the whole framework functionalities, plus the respective steps to execute each available function successfully.

It is important to mention that, this framework is recommended for users who have at least knowledge about spreadsheets and Google Docs services in order to understand the available functions of the framework. The framework operates on Chrome (recommended), Firefox, Safari and Internet Explorer browsers.

1.2 Process Overview

In order to use the DB4G Framework, it is required a computer with internet access using a recommended browser. The main pages of the framework provide some basic information and a button to login into the framework. To get access to the framework’s main interface, the authentication must be done using a google account. After that, if no configuration file is created, the user is redirected to the project creation page where it is possible to create a new project in user’s Google Drive account and assign Database Analysts and Developers.

In the context of the framework, the term “project” represents all the system structure that is automatically created. The structure is divided in three main areas. A user management where the user permissions enrolled in the system are added and updated, a configuration area where containing a backup list of the configuration file associated to the project and a database, where all the database management occurs (i.e where the data files are created, deleted, imported and exported).

The framework allows also to export versions of Google Docs (Documents, Spreadsheet, Presentations, etc.) in formats compatible with other office productivity programs such as those offered in Microsoft Office. Furthermore, alert messages may appear in some cases like creating, importing data or adding a new user,’ when they are not done correctly or something is missing.
2 Getting Started

This section will give you a starting overview of the framework's interface and, afterwards, a more detailed description of the main functions of the framework.

2.1 Interface Overview

The main interface of the framework is presented in the figure below. However, the main pages don't display the left vertical bar (i.e. sections navigation, system management and help) increasing the content area.

![Framework interface](image)

Figure 1: Framework interface

2.2 Login/Logout

In order to login into the framework and have access to all main functionalities the user just:

1. Click on "Login" button to login into the framework.

   ![Login button](image)

   Figure 2: "Login" button

2. Click on "Start DB4G Framework" to enter into the framework interface.

   ![Start using DB4G Framework button](image)

   Figure 3: "Start using DB4G Framework" button

The logout function is available everywhere, so at anytime it is possible to logout from the framework which redirects the user to the main page. In order to logout just:

3. Click on "Logout" button (the same place used for the login button).
3  Create Project

After the login, if no project exists, the user goes directly to the project creation area. The following functions from this page are described below. Note that the majority of other functions are only available after a project is created.

1. Click on "Create Project" button.

2. Complete the form

3. Click on "Create Project" button.

After that the project will be created in Google Drive, but in case the “Cancel” button is clicked the project is not created and the form disappears. An alert message is displayed if the fields are filled incorrectly.

3.1  Project Area

The project area (see figure 4) displays the structure of the project created. Note that this page can only be accessed by users with Super Administrator privileges. The following functions from this page are described below:

1. **Backup Configuration file**:

   To backup the configuration file from the application server to the cloud you just have to:

   (a) Click on "Backup Config File" button;

   (b) Click "Backup" to submit or “Cancel” to return to the current page;

2. **Edit Project**

   To edit the project name:

   (a) Click on "Edit Project" button;

   (b) Fill the new name for the project;
3. **Delete Project**

To delete the project from google and the configuration file from the application server:

(a) Click on "Delete Project" button;

(b) Click "Delete" to submit or "Cancel" to return to the current page;

4. **Open Folder**

To open a specific folder you can use one of the two options:

(a) Click on "Open" button to open the folder.

(b) Click on "WebLink" button to open the folder in Google Drive Website (a second tab in the browser will be open).

4 **Database Management**

The database management interface consists in several tables and functions in order to manipulate each file available in the database which include folders, spreadsheets, documents, pdfs, powerpoint and images. Depending on the file chosen, the number of functions can differ. This interface can only be accessed if an existing project is created. The following main functions provided by the project interface are explained below.

4.1 **Create Folder**

To create a new folder:

1. Click on "Add Folder" button.

2. Fill the form. An error message will appear if the field is incorrect.

3. Answer to the confirmation message that appears (it is possible to cancel the action). An error message may appear if a file with the same name already exists in the system.
4.2 Create Table

To create a table the following actions must be performed (note that the term “Table” refers to a spreadsheet file):

1. Click on "Create Table" button.

The following form contain some fields that must be filled in the following order:

(a) Choose a name for the table (Note that this name will be the name of the spreadsheet that will be created).
(b) Choose a name for the first worksheet (At least one worksheet is required).
(c) Define headers.
(d) Click on "Create" button to create the new file (it is possible to cancel the action). Again, an error message may appear if a file with the same name already exists in the system.

4.3 File Management

Like stated previously, the database management interface functions can differ based on the file format. However, the following functions are provided in most of the cases.

Note that some functions can only be performed by users with owner permissions (e.g. remove a file to the trash).

4.3.1 Open

The open method is only available for folders. It allows to access to the content of a specific folder. The interface and functions are the same for each folder available in the database. To open a folder just:

1. Click on "Open" button of the desired folder.
4.3.2 Preview

The preview method, as the name says, allows to preview a specific file. In this case, the function is available for files with the following format: Documents, PDFs, Images and Powerpoints. To preview a file just:

1. Click on the “Preview” button on the desired file.

4.3.3 Edit

The edit function allows the user to change the name of a specific file. The function is available for all the files in the database. To edit an existing file (i.e. change the respective name field):

1. Click on “Edit” button on the desired file.
2. Change the name field.
3. Click on “Save” button to save the changes or click on “Cancel” to cancel the changes.

4.3.4 Delete

The trash function is available for each file presented in the database. Due to some limitations a file can only be removed by his owner. To move a specific file to the trash:

1. Click on “Remove” button on the desired file.
2. Answer the confirmation message by clicking on “OK” or click on “Cancel” to cancel the changes.

4.4 Table Management

The Table management section allows to perform some actions to a specific spreadsheet file. In order to access:

1. Click on “Manage Content” button of the desired spreadsheet file.

The user is redirected to the Table Management section, as showed in figure 14 which displays a spreadsheet file as example.

The page contains some functions to return to the folder where the spreadsheet is located, an option to return to the root folder (only for users with Super Administrator privileges), and options to add users and new worksheets to the current spreadsheet.
4.4.1 Add worksheet

To add a new worksheet to the spreadsheet:

1. Click on "Add Worksheet" button.

2. Choose a name for the new worksheet.

3. Add the headers for the new worksheet.

4. Answer the confirmation message by clicking on "OK" or click on "Cancel" to cancel the changes.

4.4.2 Edit worksheet

The Edit option allows to change the name and the headers of a specific worksheet. To edit a worksheet:

1. Click on the "Edit" button on the desired file.

2. Change the fields available for edition.

3. Click on "Update" button to save the changes or click "Cancel" to cancel all the changes.
4.4.3 Delete worksheet

To delete a worksheet:

1. Click on "Remove" button on the desired file.
2. Answer the confirmation message by clicking on "OK" or click on "Cancel" to cancel the changes.

4.5 Import Files

It is possible to import files to the project.

1. Click on "Import Files" button. The files can be uploaded in two different ways.
2. You can drag and drop the files. Just open a folder and drag and drop the files into the content area.
3. Click on "Browse" button to navigate to a specified file in your local system. Before upload the files, you can perform the following actions to certain files.
4. To preview the file click on the preview icon on the desired file.
5. To remove just one file click on the remove icon on the desired file.
6. To upload one single file: Click on the upload icon on the desired file. You will see a loading bar while the files is being update.
7. To remove all the files from the list: Click on the "Remove" button.
8. To upload all the files to the database: Click on the “Upload” button.

Note that each file will be verified by the framework taking into account his name and extension. A JSON message will appear if a file with the same name already exists in the database or if the extension is not supported by the framework.

4.6 Export Files

To export a file:

1. Click on the “Download” button on the desired file (it may take some seconds when exporting files with a large size). Note that, it is not possible to export/download folders. The framework also provides the capability to download exported versions of Google Docs (Documents, Spreadsheet, Presentations, etc.) in a format compatible with other office productivity programs such as those offered in Microsoft Office.

5 Users Management

To access to the Users Management area just:

1. On the left bar click on “System Management”.
2. Click on “Users Management”.

5.1 Interface Overview

The page displays a table containing a list containing some information about the user, which include the identifier, email address, the permission (Super Administrator, Database Analyst or Developer), a role (editor or writer), name and phone. Due to some limitations, and for safety reasons only a user with Super Administrator privileges can add and update user permissions.

5.2 Add new user

To add a new user to the framework (Super Administrator privileges are required):

1. Click on “Add User” button.
2. Fill the required fields.

3. Click on "Add" button to submit or click on "Cancel" button to cancel all the actions.

### 5.3 Edit user

It is possible to edit a specific user permission. In order to that:

1. Click on "Edit" button of the desired user.
2. Change the necessary fields.
3. Click on "Update" button to save the changes into the database or click on "Cancel" button to cancel all the actions.

### 6 Configuration File

The configuration file can be previewed by all the users enrolled into the database. However, only the Super Administrator have access to the list of the backup configuration files.

To preview the configuration file:

1. On the left bar click on "System Management".
2. Click on "Users Management".

To go to the backup configuration section.

1. Navigate to the root folder folder (you can use the "Root Folder" button available in the current page).
2. Open the folder "config" from the list of available folders.

The page contains a collection of all the backup files. It provides functions to preview, download, open the file in Google Drive website or move a specific file to the trash (the functionality of these methods is the same that those provided in the Database Management section 4.3).

### 7 Recycle Bin

In order to access to the Recycle Bin:

1. On the left bar click on "System Management".
2. Click on "Recycle Bin".

The user is redirected to the Recycle Bin section, which contains a collection of all the files available in the trash of a specific user's drive account. It allows users to recover files that have been deleted from the project.

It is possible to delete permanently delete a file, or restore a file back to the project's database. The following methods are described bellow:
7.1 Restore or Delete Files

It is possible to restore or delete a specific file or multiple files at the same time (the process is the same for both functions).

a) To restore a specific file just:

1. Click on the "Restore" button on the desired file.

2. Answer the confirmation message by clicking on "OK" button or click on "Cancel" button to cancel.

b) To restore multiple files:

3. Use the selection box available for each file.

4. Click on the "Restore Select Files" button.

5. Answer the confirmation message by clicking on "OK" button or click on "Cancel" button to cancel.

c) To delete a specific file:

6. Click on the "Delete" button on the desired file.

7. Answer the confirmation message by clicking on "OK" button or click on "Cancel" button to cancel.

d) To delete multiple files:

8. Use the selection box available for each file.

9. Click on the "Delete Select Files" button.

10. Answer the confirmation message by clicking on "OK" button or click on "Cancel" button to cancel.
Appendix B

Administrative Manual

INSTITUTO SUPERIOR TÉCNICO

VERSION 2.0

DB4G Framework - Administrative Manual

Author
DIOGO SERAFIM

Supervisor
PROF. JOSÉ BORBINHA
1 Introduction

1.1 Scope and Purpose

DB4G Framework is a web-based integrated framework that allows users and developers to build and maintain information systems using the public cloud (more specifically Google Cloud Platform) without any costs associated. The framework provides usability and flexibility for a better user experience, as also the extension of the framework to new functionalities at anytime. In addition, it provides management functionalities for users to create and manage their own systems as also functionalities for a faster and easy usage of the framework.

The main goal of this manual is to guide and help developers regarding the framework configuration and installation process, as also the whole framework features, architecture and technologies, plus the respective steps to add new functionalities to the framework. It also contains the procedure steps for developers to create their own client web based applications.

It is important to mention that, this framework is recommended for developers who have experience in web technologies and MVC (Model-View-Controller) software architecture patterns.

1.2 Process Overview

The DB4G Framework is a web-based integrated framework developed in PHP which makes use of the Google APIs library to communicate, create and maintain an information system in Google Drive. In order to use the framework it is required a computer with internet access using a recommended browser. In order to authenticate into the framework the user must provide his google authentication credentials, and give permission to the framework to access to his drive account. After the authentication is done, there are two possibilities for the user. If none system (i.e. project) is created, and assuming that this user will be the Super Administrator of the project, a project creation area is opened. In this area it is possible to create and delete the project, assign database analysts, as also the developers that will participate in the system development. When the project is created, a configuration file is created in the server. It contains a reference (an identifier assigned to a constant) to each file that is created in the project. This configuration file is extremely useful for developers who only have to use these references to access and manipulate the data and build the necessary controllers and views for their web client applications. After creating the project, the user has access to the project management area where all the information is managed (i.e. where all the data, including folders, tables, documents are created, deleted, edited, imported and exported). It also provides user management functionalities and a recycle bin. While the data is being managed, the configuration file is automatically written and updated. Due to security reasons, the additional controllers and views should be implemented in a folder separated from the controllers specific to the framework. Furthermore, new functionalities can be added to the framework at anytime by the developer following the required steps that will be described later on.
2 Framework Overview

This section presents an overview of the framework's architecture and structure, as also the technologies used in the framework.

2.1 Architecture

The architecture used in the final solution of DB4G Framework is presented in figure 1.

![Architecture of DB4G Framework](image)

The Web browser and the Application Server, represent the main infrastructure communicating with the infrastructure defined by Google Cloud Platform. In order to use cloud services, the framework needs to be registered at Google App Engine environment. Therefore, private information about the registration will be kept in a JSON file in our local server. This file contains the client ID and Client Secret, used to identify the application in Google Cloud Platform. After that the application have access to all the services offered by Google. In this architecture, the user interacts with the Web Browser and performs the login in order to use the framework. In order to perform user authentication, the application server starts the OAuth2.0 process, establishing a communication with the Google Authorization Server that will be responsible for granting access to the user's info. Every time a client authenticates in the framework, a session variable (with information such access token and expire session time) will be stored. After the user authentication component approves the access, the respective user interface containing all the information about a project is displayed. The Project Area component displays information about the system structure and it is divided in three main components:

- System Management - in this component the user manages all the data related to the system environment (e.g. users management, trash management and backup management).
• Data List Management - in this component the user defines and manages data presented in the cloud database (e.g. file information, spreadsheet information, users permissions).

• Task Execution - in this component the user executes tasks related to information provided in the data list management (e.g. edit file metadata, create file, import file, export file)

All functionalities performed by the user in the Web Browser, will interact with the Application Server through a HTTP/Ajax communication. The Application server is divided in two layers: Application Logic and Business Logic.

The Business Logic implements the core functionality of the system and encapsulates the business rules that determine how data can be created, stored and changed. The application logic contains all the sections that are managed by the user which include Users, System and Database Management as also all the tasks performed by the user. Each component depends from the communication with the cloud database as explained by the flow of dependencies. Every time the user performs a task the Application Logic layer send the commands to the layer bellow (i.e. the Business Layer). Then the Business layer communicates with the Google APIs and send the request to the cloud endpoint to update or retrieve data according to the business rule implemented. By accessing through the App Engine, the application server will have access to all the services and features available, each one fully integrated with cloud database where all the organization files are stored.

2.2 Technology

The technologies used for the concept of this solution are, HTML for the front-end (plus Javascript, Jquery and CSS). In addition, it is also used Bootstrap for styling purposes. For the back-end (i.e for the application server) it was created a PHP framework using an MVC (Model-View-Controller) pattern:.

By using an MVC pattern, the framework is divided in three main components: model, view, controller. In this case, the Business Logic is defined in the Model, which defines how the data will be stored and retrieved, while the Application Logic is defined in the Controller, which controls the interaction between the Model and the View. Finally, the view provides the output representation of information to the user interface, based on the information that is retrieved/updated in the model.

Regarding the libraries used, they are the following:

• Web Browser
  – jQuery/Javascript.
  – jQuery File Upload (to import data to the cloud).

• Application Server
  – JSON Access token (for OAuth2.0 authentication).
  – PHP File System (to manage/upload local files).
  – Google Drive API (to manipulate files and user permissions).
  – Google Sheets API (to create, insert, update and delete spreadsheet files).
2.3 Structure

The framework file structure is displayed in figure 2. The structure can be divided into two main folders, application (blue square) containing all the application logic and public (red square) containing all the files that are accessible for the user.

This division is justified by the fact that when the user enters the server, the .htaccess file in the framework's root folder will route him to /public/index.php. This action prevents any access to the application folder, making the framework much more secure.

The index.php loads the application configuration files stored in the config folder (green square) containing the recognized path/URL settings.

The public folder also contains css and js folders (containing javascript and css files) used in the front-end of the framework.

The application folder (blue square) contains 5 folders. The config folder explained above. The controller (orange square) contains all the necessary controllers which accept the inputs and converts it to commands for the model or view. The core folder (yellow square) contains two files: the application file used to start the application and all the calls made by the user. The controller file is responsible for load all the model classes.

The model folder (brown square) contains business model classes containing functions responsible for the interaction between the server and the public cloud, being directly connected to the Google API library's (presented in vendor folder). It receives the information provided by the controllers and makes the requests to the Google APIs (in this case each class is responsible for managing data for a certain API). For example the file "spreadsheetModel.php" is responsible for communicate with the Google Sheets API and manage all the spreadsheets information contained in Google Drive.

The client-secret folder contains a JSON file where the OAuth 2.0 credentials are stored. This file is extremely important since it provides the authentication information necessary to identify and connect the framework into the google cloud endpoint. The credentials folder contains all the OAuth 2.0 credentials specifically for the service account (explained later on).

As mentioned before, the configuration file containing all the references to the files created in the project, hosted in the google drive. Finally, the view folder (black square) contains all the files responsible for the user interface of the framework and representation of the results provided by the controller. Finally the README.txt file that displays a brief explanation of the framework file structure.

Figure 2: DB4G Framework Structure
3 Framework Installation

This section describes all the necessary steps to install the framework in a server plus the necessary steps to configure the web application in Google Cloud Platform.

3.1 Prerequisites

First of all, in order to install the framework the following requirements are required.

- PHP 5.4 or greater with the command-line interface (CLI) and JSON extension installed.
- Apache2 Server with mod_rewrite activated.
- Composer (a tool for dependency management in PHP).

3.2 Step 1: Configuration of mod_rewrite in Apache Web Server

After installing the Apache Web Server in your system you have to edit the "httpd.conf" file in order to activate the mod_rewrite.

1. Open your prompt terminal
2. Go to the apache root folder (default location is: /etc/apache2).
3. Open the "httpd.conf" file (You’ll need to provide administrator permission).
4. Uncomment the following line "LoadModule rewrite_module libexec/apache2/mod_rewrite.so"
5. Edit the following lines as presented in (change AllowOverride from None to All)

The mod_rewrite is a module that uses a rule-based rewriting engine to map a URL to a file system path and redirect one URL to another URL. It provides a flexible and powerful way to manipulate URLs using an unlimited number of rules. Each rule can have an unlimited number of attached rule conditions, to allow you to rewrite URL based on server variables, environment variables, HTTP headers, or timestamps. It operates on the full URL path and can be invoked in httpd.conf or .htaccess. The path generated by a rewrite rule can include a query string or can lead to internal sub-processing or external request redirection.

3.3 Step 2: Download DB4G Framework and Load Composer Dependencies

After downloading the framework to your web server it is necessary to install the Google APIs Client library, using composer.

In this case, each library required was added to a composer.json file and for that reason the developer only have to perform the following steps.

1. Go to the framework root folder in your web server.
2. Locate the file "composer.json".

3. Using administrator privileges run the following command in your terminal: composer install

3.4 Step 3: Google Cloud Developers Console

In order to make use of the framework it is necessary to create a project in the Google Developers Console and automatically turn on the APIs.

First the developers and the administrators of the project have to define an email account that will be used to register the project in the Google Cloud Platform. Note that this email account will be used to create the project root folder in Google Drive and to connect the framework to the Google Cloud Endpoint.

The necessary steps are described bellow:

1. Using a recommended browser, open the following URL:

2. After sign in, you must choose a name for the project.

   Before adding the OAuth credentials to the project page it is necessary to turn on others APIs that are used by the framework.

3. Go to the library panel, look for the G Suite APIs and turn on the Google Drive, Google+ and Google Sheets API (If you want to add other API library’s to the project for framework extension you must turn on other APIs in the library panel).

4. After turn on the APIs, go to the credentials page.

5. At the top of the page, select the OAuth consent screen tab. Select the email address, enter a product name (if not already set), and click the Save button.

6. Select the Credentials tab, click the Create credentials button and select OAuth client ID.

7. Select the application type Other, enter a name (e.g. "DB4G Framework API"), and click the Create button. Click OK to dismiss the resulting dialog.

8. Click the (Download JSON) button to the right of the client ID.

9. Move this file to the directory .client-secret (in the model folder) and rename it with "client_secret.json".

3.5 Step 4 (Optional): Generate Service Account Credentials

It is also possible to generate service account credentials, which is an account that belongs to your application instead of to an individual user and allows your application to make calls to Google APIs on behalf of the service account, so users aren’t directly involved. Moreover, if your application uses Google APIs to work with its own data rather than a specified user’s data you should use a service account to authenticate its calls to each API.
For that reason a script was created in order to make the process much easier to the developer. The steps to create generate the credentials are described above.

- Open the prompt terminal in your system.
- Go to your web server directory folder where the framework root folder is located.
- Locate the file "generateServiceCredentialsScript.php".
- Run the following command "php generateServiceCredentialsScript.php".
- Following the information displayed in the terminal.
- Check that the file "serviceCredentials.json" was added to /model/.credentials folder.

4 Framework Extension

This section gives you an overview of some aspects of the framework, plus it describes how the new controllers are added to the framework and the necessary steps to do it successfully.

4.1 Model and Controllers Overview

Since the main goal of this thesis was the development framework that could be adapted to various scenarios a generic business model was created. This model contains classes that encapsulate operations and rules that allows to perform actions in Google Drive using the Google API libraries, regardless of the type of data being stored.

So, in order to save time for the developer all the model was build and tested and for that reason, the developer only have to create the necessary controllers and views with respect to the application domain model that will be implemented.

At this moment the framework provides the following generic classes:

- **File** - contains functions (e.g. create, read, edit, delete, update) to manage files in Google Drive.
- **Folder** - contains functions (e.g. create, read, edit, delete, update) to manage folders in Google Drive.
- **UserPermissions** - contains functions (e.g. add, update, delete) to manage user permissions in Google Drive.
- **UserProfile** - contains functions to retrieve user account information.
- **Authentication** - contains functions responsible for making the OAuth2.0 authentication and Service Account Authentication.
- **Spreadsheet** - contains functions to manage (e.g. create, delete, read, update) spreadsheet files in Google Drive.
4.2 Planning your own web application

Before starting the implementation of your own web application, it is recommended to plan it first in order to follow all steps and to avoid revisiting previous steps, because something is missing. It might become more difficult to know where the mistakes are (if the problem is in the controller, in the model or just in the view) or where the missing data should be, which takes much more time to finish the implementation. Before starting the implementation you should answered to the following questions:

- What are the main objectives and requirements of the application?
- What are the Google APIs that I will use in order to manipulate data. Do I need to add a new class to the model?
- What type of authentication should I use to connect the application to the Google Cloud?
- How many views and controllers do I need to create my application?

The following quick start tutorial will help you to understand how to extend the framework to other functionalities and new APIs, but also how to start creating your web application.

4.3 Files Examples and Basic Methods Overview

The framework provides 8 files as example of how the back-end and front-end should be done, but also represent the mandatory methods that every new functionality should have at least. These files will be necessary to start building a new web application and their names and description are the following:

- **example_header.php**: how the header should look like, which respects the same interface already provided by the framework;
- **example_data_entry_form.php**: contains an example of a data form used to insert data in a table;
- **example_view_results.php**: displays all the results provided by the controller. In this case displays all the content of a specified spreadsheet;
- **example_controller.php**: contains the main methods used to manipulate data using the business model, which methods to retrieve, create, update and delete data.
- **example.css**: style sheet containing the necessary main styling of every section.
- **example_model_GmailAPI.php**: contains an example methods to retrieve a specified number of messages from a user’s mail inbox, using Gmail API library;
- **example_email_controller.php**: contains methods to retrieve a list of specified number of messages from the Gmail model;
- **example_email_results.php**: displays all the results provided by the controller. In this case displays the first twenty messages from a user’s mailbox;
4.4 Quick Start Tutorial

Before start creating your application, you must perform the installation steps that are described in section 3.

Recall that in order to create your project and build your initial database, you must use the framework interface. If necessary please make use of the User Manual available in Appendix A.

4.4.1 Adding a new class to the model (Optional)

If you want to use a different API library that is not yet available in the framework you must perform the following steps (note that the figures and files are only examples but the steps are the same for every class being implemented):

1. Create a new file in the existing “model” folder with the name “gmailModel.php”.

   • Every new class that is created must have a required function to get the client authentication access token. It is also the file where you would implement the necessary functions to connect and use a certain API library.

   • Copy all the code available in the file “example_model_GmailAPI.php” to the file “gmailModel.php”.

   • Open the file “apiScopesModel.php”;

   • Add the following line to the Scope model: Google_Service_Gmail::GMAIL_READONLY

2. In order to load the new class model to your application, you have to create a new object. To do that open the file “controller.php” available in /application/core and add the following lines.

   • require APP . ’model/className.php’
     (e.g. require APP . ’model/gmailModel.php’);

   • $this->objectName = new ClassName()
     (e.g. $this->email = new Gmail());

4.4.2 Extending the framework to a new method

The following steps shows how to add a new controller to the framework main functionalities that makes use of the model class implemented above.

1. In /application/controller create a new file with the name “email.php”;

2. Copy the content inside the file “example_email_controller.php” to the file created in (1).

   Take a look to some methods inside the file.

   The login method, is used for authenticate the user into Google Cloud. When the user logs into the framework it will be redirected to a Google Authentication Server to provide his email and password.
Take a look to the index() method in the controller. Similar to the login() method, but in this case it requests data to the model.

Basically the $this->email->listMessages() calls the listMessages() method that was created in the application/model/gmailAPIModel.php.

4.4.3 Update Interface

The results (provided in the $emailList variable), can then easily be rendered to a view inside /views. The file example_listMessages.php contains the code necessary to list in a table all the email messages returned by the controller.

1. Create a folder in /view and rename it with "email";
2. Create a file in /view/email with the name "email.php";
3. Copy the content inside the file "example_email_results.php" and past it inside the file created above;
4. In /view/_templates/ open the file "initialiseHeader.php";
5. Add a new section to the left navigation bar;

After that you should have the new functionality available in the framework interface as shown in figure:

Figure 3: Email Management section after the extension

4.4.4 Extend the framework to a new web application

Since the framework have some controllers already implemented, in order to avoid mistakes (e.g. deleting a file that should not be deleted) and make the structure much more clean and organized, the framework provides a folder named as "system", available in the "view" and "controller" folders. So, in order to add new controllers and views to create your own web application, you have to define them inside that folder.
**Working with data from cloud database**

Assuming that your web application will make use of the database created using the framework interface, it is important to explain the necessary steps that will help the developer to start creating the web application.

The following steps describe an example of how to obtain data from a spreadsheet:

- First and if you don’t have a project already created, login into the framework and create the project following the steps described in the User Manual;
- Create or import your spreadsheet data (if necessary, import the spreadsheet file named "Data_Example.xlsx" available in the examples folder);
- Locate the configuration file named as “dbinfo.php”, containing all the information regarding the files created in the database. By looking to the file you should find the name of the spreadsheet file and the correspondent ID as shown in figure 4;

![Figure 4: Configuration file](image)

After that you only have to build the necessary controllers and views. The following steps describe how the implementation can be done and how the new web application can be easily integrated. Some example files presented can be used to start the implementation of the new application, by containing some generic methods to create, update and delete data from the database.

**Implementation: Back-end**

1. Before start creating the necessary files, check that a folder named as "system" is presented inside the "controller" and "view" folders.
2. In controller/system, create a new controller (e.g. example_controller.php).
3. In view/system, create a new view (e.g. example_view.php).
4. Copy the content inside the example file "example_controller.php", and paste in the controller file created. Note that the name of the class should be equal to the name of the file.

Note that the first lines of the controller contains a function to include the configuration file available in the model (see figure 5). Every time you create a controller, this lines must be included in order to use the configuration file and have access to the files available in the cloud database.
Each method available in the file, is responsible for making calls to the business model created in the model. In this particular case, it is responsible for making calls to the class responsible for manipulate all the spreadsheet data (i.e. class Spreadsheet).

The index() method first defines the type of authentication (in this example we are using service account authentication) being used to authenticate the application as seen in figure.

Figure 5: Require function to include configuration file

```
class ExampleController extends Controller
{
    public function index()
    {
        $client = $this->authentication->getAccountServiceAuthentication();
        $logged = $this->authenticate($this->logged);
        $client = $this->authentication->getAuthentication();
        $SESSION['serviceAccountSet'] = 'set'; /# This variable is used in the Spreadsheet model to check the type of authentication.
        $currentSpreadsheet = $this->spreadsheet->getSpreadsheetById(Data_Example);
        $worksheetContent = $this->spreadsheet->getWorksheetByTitle($currentSpreadsheet,
        $voluntiries); $voluntiriesContent = $this->spreadsheet->getWorksheetListFeed($worksheetContent);
        $voluntiriesHeader = $this->spreadsheet->getWorksheetTableHeader($worksheetContent);
        require APP . '/view/system/templates/example_header.php';
        require APP . '/view/system/dataExample/example_view_results.php';
    }
}
```

Figure 6: Retrieve all the content from a specific spreadsheet

As you can see, the variable "Data_Example" presented in the configuration file is used to get the spreadsheet by ID. The controllers also contains functions to manipulate data, i.e to add, update and delete entries from the database, as described in figures 7, 9 and 8.

Figure 7: Add new entry to the spreadsheet

```
public function addNewEntry(){
    if(isset($_POST['submit_new_volunteer'])){ $currentSpreadsheet = $this->spreadsheet->getSpreadsheetByTitle(Data_Example);
        $voluntiriesWorksheet = $this->spreadsheet->getWorksheetByTitle($currentSpreadsheet,

        $voluntiriesContent = $this->spreadsheet->getWorksheetListFeed($voluntiries);
        $row = array();
        for($i = 0; $i < sizeof($_POST['key_values']); $i++){
            if($_POST['key_values'] ['$i'] == 'id'){
                $row[$_POST['key_values'] ['$i'] = uniqid();
            }else{
                $row[$_POST['key_values'] ['$i'] = $_POST['row_values'] ['$i'];
            }
        }
        $this->spreadsheet->addNewRowContent($voluntiriesWorksheet,$row);
        $voluntiriesWorksheet = $this->spreadsheet->getWorksheetListFeed($voluntiriesWorksheet);
        require APP . '/view/system/templates/example_header.php';
        require APP . '/view/system/dataExample/example_view_results.php';
    }else{
        require APP . '/view/system/templates/example_header.php';
        require APP . '/view/system/dataExample/example_data_entry_form.php';
    }
}
```

Figure 8: Update entry based on a specified primary key

```
public function updateEntry(){
    $row = array();
    for($i = 0; $i < sizeof($_POST['key_values']); $i++){
        $row[$_POST['key_values'] ['$i'] = $_POST['row_values'] ['$i'];
    }
    $spreadsheet = $this->spreadsheet->getSpreadsheetByTitle(Data_Example);
    $voluntiriesWorksheet = $this->spreadsheet->getWorksheetByTitle($spreadsheet, 'Users');
    $this->spreadsheet->updateRowByKey($voluntiriesWorksheet, $_POST['key_values'] ['$0'], $_POST['row_values'] ['$0'], $row);
}
```
Finally the output is rendered to the view available in view/system/dataExample folder.

**Implementation: Front-end**

1. In view/system create a new folder named as "dataExample";

2. Copy the following example files to the folder:
   - example_header.php;
   - example_data_entry_form.php;
   - example_view_results.php;

4.5 Testing

You can test your application and the whole framework in your laptop before updating to the actual server:

- Open the respective link.
- The webpage should be the same as shown in figure 10.

![Figure 10: Template used to build a new web application](image-url)
Appendix C

User Evaluation - Guidelines

DB4G Framework Evaluation

DB4G Framework is a web-based integrated framework that allows users and developers to build and maintain information systems for organizations using the Google Cloud Platform with no costs associated. The framework provides usability and flexibility for a better user experience, as also the extension of the framework to new functionalities at anytime. In addition, it provides management functionalities for users to create and manage their own systems as also functionalities for a faster and easy usage of the framework.

The evaluation main goal is to evaluate the framework’s usability and interface, but also evaluate the user experience. It consists on several tasks to be completed after the main setup is done and finally there is a questionnaire to be answered. The evaluation should take between 25-35 minutes.

Note: it is recommended that you have at least knowledge about spreadsheets and Google Docs services in order to understand the available functions of the framework.

Thank you for your collaboration and time.

Setup

Before starting the tasks, the following requirements should be satisfied:

- Have a computer with internet access;
- Explore the “Import files” folder that was given (contains some files that will be needed for some tasks);
- Open a browser: Chrome or Safari;
- Go to: db4gframework.dlinkddns.com

Tasks

After doing all the setup requirements, you can start executing the tasks below on the framework following the order displayed. The User Manual provided, is just for consulting, in case you don’t know how to do a step from the task. These tasks should be done individually and in the end the respective questionnaire should be answered.

Scenario: Login using a Google account

Task A

1. Login into the framework (Google credentials of your choice);
2. Go to DB4G Framework interface;
Scenario: Create a new project in your Google Drive account, assign a developer and a database analyst and check that the structure was created in your Google Drive account;

Task B
1. Create a new project;
   (a) Add the name “Project DB4G” to the project;
   (b) Fill the database analyst information with the email: databaseAnalystDB4G@gmail.com.
   (c) Fill the developer information with the email: developerDB4G@gmail.com (the name and phone you can choose whenever you want);
   (d) Confirm and Execute all the process;
   (e) Open a new tab in your browser;
   (f) Go to Google Drive: https://www.google.com/drive
   (g) Login with the same email account that you are logged in DB4G Framework;
   (h) Check that a folder with the name “Project DB4G” was created (do not close the page);
   (i) Return back to the framework page;

Scenario: Take a look to the project area and explore some functions;

Task C
1. Go to the System Management;
2. Take a look to the Configuration File;
3. Close the configuration file window;
4. Backup the Configuration File;
5. Navigate to the “Configuration” folder and check that the file was added to the project;
6. Return back to the project area;

Scenario: Take a look to the Users Management section and explore the functions available;

Task D
1. Go to System Management → Users Management;
2. Explore the functions available;
3. Change the Developer’s name;
4. Check that the information presented in the table was changed;
5. Change the role type of the Developer to Database Analyst;
6. Do not close the page;

7. Open a new page and enter into Developer’s Gmail account:
   • Email: developerDB4G@gmail.com
   • Password: db4gEvaluationTest2017;
8. Check that you have received an email in Developer’s account;
9. Return back to the project area;

Scenario: Check that the user permissions and project structure were created successfully;

Task E
1. Go again to your Google Drive account (the second tab that you opened in your browser);
2. Open the folder “Project DB4G”
3. Check that the folder contains 3 folders: Users, Configuration and Database;
4. Return to the project area;
5. Close the second tab (i.e. Google Drive webpage);
6. Logout;

Scenario: Explore Database Management area.

Task F
1. Login again but now using the following credentials:
   (a) Email: databaseAnalystDB4G@gmail.com;
   (b) Password: db4gEvaluationTest2017;
2. Enter into the framework interface;
3. Check that you have been redirected to the Database Management area;
4. Take a look to the Database Management area (the current folder will be “Database”);
5. Have a quick look to the functions available (just click in some buttons to understand what they are supposed to do);

Scenario: Add sub-folders to the database;

Task G
1. Create a folder with the name: Folder_1
2. Confirm that the folder is created in the configuration file;
3. Check that the folder was created successfully in Google Drive using the “WebLink” function;
4. Change the name of the folder to: NewVersion;
Scenario: Create tables;

Task H

1. Create a table (i.e. a spreadsheet file) with the following information;
   (a) Name: First_Spreadsheet;
   (b) Entity name: First_Worksheet;
   (c) Attributes: id, email, name, phone;
2. Verify that the table was created successfully;
3. Verify that the table was created successfully in the Google Drive using the "WebLink" function;
4. Verify that the table information (name and id) is presented in the configuration file;

Scenario: Add more entities (i.e. worksheets) and attributes (i.e. headers) to a specific table; Change attributes;

Task I

1. Open the table management section of the previous table (use the function "Manage Content");
2. Have a quick look to the section and the available functions;
3. Edit the following information:
   (a) Change the name of the first worksheet to: Worksheet_NewVersion
   (b) Change the attributes "id" and "email" to "identification" and "emailAddress" respectively;
   (c) Confirm all the changes;
   (d) Check that the information was updated successfully;
4. Create a new worksheet (you can choose the name and attributes) following the same process made in the creation of the first table;
5. Return back to the database main folder;

Scenario: Import functions; Import the files provided in the folder "Import Files"; Trying to add a file with the same name to the database;

Task J

1. Import all the files provided in the folder "Import Files" to the import box by drag-and-drop the files;
2. Preview the file "Test.pdf";
3. Upload only the file "Test.pdf";
4. Confirm that the file was updated;
5. Remove the file "Document.docx";
6. Confirm that the file was removed from the import box;
7. Using the "Browse" option upload again the file "Document.docx";
8. Upload all the files;
9. Try to upload again the file "Test.pdf";
10. Return to the database;

Scenario: Explore the files imported; Check that the migration of Office 365 files into Google Sheets was done correctly;

Task K

1. Open the table management of the spreadsheet "Familias";
2. Open the spreadsheet your local computer;
3. Confirm that the information displayed is the same following the steps described below:
   (a) Worksheets information: Name, Number of Files and Number of Columns;
   (b) Worksheets headers (i.e. column headers);
4. Open the spreadsheet in Google Drive and check that the conversion from Office 365 format to Google Sheets was done successfully (while comparing with the file opened in your local computer);

Scenario: Documents edition; Explore additional functionality provided by the framework to edit documents;

Task L

1. Import the file "Document.docx" to the database;
2. Preview the file;
3. Try to edit the file using the tools available by Google Docs interface;
4. Open the file in your Google Drive;
5. Confirm that the changes are updated automatically while you are editing the file;
6. Close the edition box;
7. Return to the database;
Scenario: Export functions; Export files to a local device;

Task M
1. Export the following files;
   (a) Spreadsheet file: "Familias";
   (b) PDF file: "Test";
   (c) Document: "Document";
2. Confirm that the files were exported successfully, that the document and spreadsheet files were converted to Office 365 format successfully;
3. Logout;

Scenario: Remove Files; Delete files and request permission to Super Administrator;

Task N
1. Delete the following files:
   (a) Spreadsheet file "Familias";
   (b) PDF file "Test.pdf";
   (c) Folder "Pasta1NovaVersao";

Scenario: Explore Recycle Bin; Move files to the trash and then try to recover them;

Task O
1. Login using your Super Administrator credentials;
2. Go the database management;
3. Check that you have requests from the Database Analyst;
4. Move the following files to the trash;
   (a) Spreadsheet file "Familias";
   (b) PDF file "Test.pdf";
   (c) Folder "Pasta1NovaVersao";
5. Confirm that the files were removed from the database;
6. Confirm that the information regarding those files was removed from the configuration file;
7. Recover all the files;
8. Confirm that the configuration file is updated;
9. Return back to the database;
10. Confirm that the files are again in the database;

Scenario: Final stage; Exploring the real-time collaboration and revision history tool; Delete the project;

Task P
1. Login using your Super Administrator credentials;
2. Go the database management;
3. Check that you have access to all the files created by the Database Analyst in the previous steps;
4. Change the name of the file "Familias" to "FamiliasEditado";
5. Confirm that the attributes "updated time" and the "last modifying user" were updated;
6. Go to the project area root folder;
7. Delete the project;
Appendix D

User Evaluation - Survey

Survey - DB4G Framework

DB4G Framework is a web-based integrated framework that allows users and developers to build and maintain information systems for organizations using the Google Cloud Platform with no costs associated. The framework provides usability and flexibility for a better user experience, as also the extension of the framework to new functionalities at anytime.

The survey should only take between 5 and 10 minutes, and your responses are completely anonymous.

Thank you for your participation and time! :D

* Required

Usability
The following questions regarding the framework usability

1. 1. Rank the degree of difficulty for: *
   Mark only one oval per row.

<table>
<thead>
<tr>
<th></th>
<th>Very Difficult</th>
<th>Difficult</th>
<th>Easy</th>
<th>Very Easy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assign Users</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explore Project Area</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add sub-folders</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create tables</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add more tables</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edit table information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Import data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document Edition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove files from the project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recover files back to the project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export files</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logout</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. 2. Rank the usefulness of: *
   Mark only one oval per row.

<table>
<thead>
<tr>
<th></th>
<th>Should not be available</th>
<th>Useless</th>
<th>Useful</th>
<th>Extremely Useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration File</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Import Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document Edition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recycle Bin and Recovering files</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. At any time, I knew what was happening on the framework. *
Mark only one oval.

1 2 3 4 5
Strongly Disagree Strongly Agree

4. 3.1 Comments

________________________________________
________________________________________
________________________________________
________________________________________

4. How satisfied are you regarding the whole experience performed on the framework?
Mark only one oval.

1 2 3 4 5
Very Unsatisfied Totally Satisfied

User Experience
The following questions regarding the user experience.

5. I think that I would like to use this framework frequently if I have an organization. *
Mark only one oval.

1 2 3 4 5
Strongly Disagree Strongly Agree

7. 5.1 Comments

________________________________________
________________________________________
________________________________________
________________________________________

6. I thought the framework was easy to use. *
Mark only one oval.

1 2 3 4 5
Strongly Disagree Strongly Agree

8. 6.1 Comments

________________________________________
________________________________________
________________________________________
________________________________________

9. I thought that the framework is very efficient. *
Mark only one oval.

1 2 3 4 5
Strongly Disagree Strongly Agree

10. 7.1 Comments

________________________________________
________________________________________
________________________________________
________________________________________

11. I felt very confident using the framework *
Mark only one oval.

1 2 3 4 5
Strongly Disagree Strongly Agree

12. 8.1 Comments

________________________________________
________________________________________
________________________________________
________________________________________

13. I found the various functions in this framework were well integrated. *
Mark only one oval.

1 2 3 4 5
Strongly Disagree Strongly Agree

14. 9.1 Comments

________________________________________
________________________________________
________________________________________
________________________________________
15. 9.1 Comments

16. 10. I found the framework and its interface pleasing and user-friendly. *
Mark only one oval.

1  2  3  4  5

Strongly Disagree  Strongly Agree

17. 10.1 Comments

18. 11. I think that the framework interface is innovative and creative. *
Mark only one oval.

1  2  3  4  5

Strongly Disagree  Strongly Agree

19. 11.1 Comments

Personal Opinion
The following questions regard your personal opinion about the framework. These questions are not mandatory, but it would appreciate an opinion regarding the negative and positive aspects of the framework.

20. 12. Negative aspects of the framework?

21. 13. Positive aspects of the framework?

22. 14. Do you have any thoughts or ways to improve the framework?

23. 15. Any last observation or comment regarding the whole framework?
## Appendix E

### User Evaluation - Results

| Users | Time | Usability | Usability - Question 1. Rank the degree of difficulty for: | Usability - Question 2. Rank the usefulness of: | Usability - Question 3. Rank the satisfaction: | Usability - Question 4. Rank the agreement with: |
|-------|------|-----------|----------------------------------------------------------|--------------------------------------------------|--------------------------------------------------|
|       |      | 1.1       | 1.2                                                      | 1.3                                              | 1.4                                              | 1.5                                              | 1.6                                              | 1.7                                              | 1.8                                              | 1.9                                              | 1.10                                             | 1.11                                             | 1.12                                             | 1.13                                             | 1.14                                             | 1.15                                             | 1.16                                             | 1.17                                             |
| 1     | 29 m | A         | A                                                        | B                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               | B                                               | A                                               | A                                               | A                                               | A                                               | A                                               |
| 2     | 37 m | A         | A                                                        | B                                               | B                                               | A                                               | A                                               | A                                               | B                                               | A                                               | A                                               | A                                               | A                                               | B                                               | A                                               | A                                               | A                                               |
| 3     | 32 m | B         | A                                                        | A                                               | B                                               | A                                               | B                                               | A                                               | A                                               | B                                               | A                                               | A                                               | B                                               | B                                               | B                                               | B                                               | B                                               |
| 4     | 24 m | A         | A                                                        | A                                               | A                                               | B                                               | A                                               | A                                               | A                                               | A                                               | B                                               | A                                               | A                                               | A                                               | B                                               | A                                               | A                                               |
| 5     | 27 m | A         | A                                                        | A                                               | C                                               | B                                               | C                                               | B                                               | B                                               | B                                               | A                                               | A                                               | A                                               | A                                               | B                                               | A                                               | A                                               |
| 6     | 23 m | A         | A                                                        | B                                               | B                                               | A                                               | C                                               | A                                               | A                                               | B                                               | A                                               | B                                               | A                                               | A                                               | A                                               | A                                               | A                                               |
| 7     | 28 m | A         | A                                                        | A                                               | A                                               | B                                               | A                                               | B                                               | A                                               | B                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               |
| Med.  | 28 m | A         | A                                                        | A                                               | A                                               | B                                               | A                                               | B                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               |
| Mod.  | -    | A         | A                                                        | A                                               | A                                               | B                                               | A                                               | B                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               | A                                               |


Table E.1: Users evaluation table: results from usability question 1.

<table>
<thead>
<tr>
<th>Users</th>
<th>Question 2. Rank the usefulness of:</th>
<th>Question 3. Rank the satisfaction:</th>
<th>Question 4. Rank the agreement with:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.1</td>
<td>2.2</td>
<td>2.3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Avg.</td>
<td>4.28</td>
<td>4.71</td>
<td>4.71</td>
</tr>
<tr>
<td>Med.</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Mod.</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>


Question 2. On a scale of 1 to 5: 1) Should not be available and 5) Extremely useful.

Question 3. On a scale of 1 to 5: 1) Strongly disagree and 5) Strongly agree.

Question 4. On a scale of 1 to 5: 1) Very unsatisfied and 5) Very satisfied.

Table E.2: Users evaluation table: results from usability questions 2, 3 and 4.
Questions 5. to 11. On a scale of 1 to 5, 1 being strongly disagree and 5 strongly agree.

Table E.3: Users evaluation table: results from user experience questions 5 to 11

<table>
<thead>
<tr>
<th>Users</th>
<th>User Experience - Questions 5 to 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4 5 5 5 5 4 5</td>
</tr>
<tr>
<td>2</td>
<td>3 4 5 5 4 5 5</td>
</tr>
<tr>
<td>3</td>
<td>5 4 5 4 5 5 4</td>
</tr>
<tr>
<td>4</td>
<td>5 4 5 5 5 5 5</td>
</tr>
<tr>
<td>5</td>
<td>4 4 3 3 5 3 5</td>
</tr>
<tr>
<td>6</td>
<td>3 4 4 3 4 4 4</td>
</tr>
<tr>
<td>7</td>
<td>4 4 5 5 4 5 5</td>
</tr>
<tr>
<td>Avg.</td>
<td>4 4.1 4.6 4.3 4.6 4.3 4.7</td>
</tr>
<tr>
<td>Med.</td>
<td>4 4 5 5 4 5 5</td>
</tr>
<tr>
<td>Mod.</td>
<td>4 4 5 5 4.5 5 5</td>
</tr>
</tbody>
</table>


Table E.3: Users evaluation table: results from user experience questions 5 to 11
Appendix F

Domain Model - Attributes Description

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>firstName</td>
<td>Variable used to store the first name of the authenticated user.</td>
</tr>
<tr>
<td>Users</td>
<td>familyName</td>
<td>Variable used to store the last name of the authenticated user.</td>
</tr>
<tr>
<td>Users</td>
<td>id</td>
<td>Variable used to store the user’s ID.</td>
</tr>
<tr>
<td>Users</td>
<td>email</td>
<td>Variable used to store the email account used to login into the framework.</td>
</tr>
<tr>
<td>Users</td>
<td>phone</td>
<td>Variable used to store the phone number of the user.</td>
</tr>
<tr>
<td>Users</td>
<td>role</td>
<td>Variable used to store the role type of the user. It can assume the values: Super Administrator, Database Analyst, Developer.</td>
</tr>
<tr>
<td>User Permissions</td>
<td>permissionID</td>
<td>Variable used to store the user’s permission list.</td>
</tr>
<tr>
<td>User Permissions</td>
<td>role</td>
<td>Variable used to store the role type of the user. It can assume the values: Owner, Editor, Writer.</td>
</tr>
<tr>
<td>Project</td>
<td>name</td>
<td>Variable used to store the name of project created in the cloud.</td>
</tr>
<tr>
<td>Project</td>
<td>owner</td>
<td>Variable used to store the email of the owner (i.e. the account used to create the project).</td>
</tr>
<tr>
<td>Project</td>
<td>creationTime</td>
<td>Variable used to store the time the project was created.</td>
</tr>
<tr>
<td>Configuration</td>
<td>timestamp</td>
<td>Variable used to store the timestamp of the configuration file.</td>
</tr>
<tr>
<td>Configuration</td>
<td>owner</td>
<td>Variable used to store the owner of the project in the configuration file.</td>
</tr>
<tr>
<td>Configuration</td>
<td>creationTime</td>
<td>Variable used to store the time the configuration was backup from the application server.</td>
</tr>
<tr>
<td>Folder</td>
<td>name</td>
<td>Variable used to store the folder name.</td>
</tr>
<tr>
<td>Folder</td>
<td>id</td>
<td>Variable used to store the folder ID.</td>
</tr>
<tr>
<td>Folder</td>
<td>mime type</td>
<td>Variable used to store the format/extension of the folder.</td>
</tr>
<tr>
<td>Folder</td>
<td>owner</td>
<td>Variable used to store the folder’s owner.</td>
</tr>
<tr>
<td>Folder</td>
<td>creationTime</td>
<td>Variable used to store the time the folder was created.</td>
</tr>
<tr>
<td>Folder</td>
<td>lastUpdate</td>
<td>Variable used to store the time of last update made in the folder.</td>
</tr>
<tr>
<td>File</td>
<td>name</td>
<td>Variable used to store the file name.</td>
</tr>
<tr>
<td>File</td>
<td>id</td>
<td>Variable used to store the file ID.</td>
</tr>
<tr>
<td>File</td>
<td>mime type</td>
<td>Variable used to store the format/extension of the file.</td>
</tr>
<tr>
<td>File</td>
<td>owner</td>
<td>Variable used to store the file’s owner.</td>
</tr>
<tr>
<td>File</td>
<td>description</td>
<td>Variable used to store the description of the file.</td>
</tr>
<tr>
<td>File</td>
<td>creationTime</td>
<td>Variable used to store the time the file was created.</td>
</tr>
<tr>
<td>File</td>
<td>lastUpdate</td>
<td>Variable used to store the time of last update made to the file.</td>
</tr>
<tr>
<td>File</td>
<td>lastModifyingUser</td>
<td>Variable used to store the email of the user that made the last changes to the file.</td>
</tr>
<tr>
<td>File</td>
<td>webLink</td>
<td>Variable used to store the url that provides access to the file via Google Drive website.</td>
</tr>
<tr>
<td>Spreadsheet</td>
<td>name</td>
<td>Variable used to store the spreadsheet name.</td>
</tr>
<tr>
<td>Spreadsheet</td>
<td>listOfWorksheets</td>
<td>Variable used to store a list of worksheets contained in the spreadsheet.</td>
</tr>
<tr>
<td>Spreadsheet</td>
<td>lines</td>
<td>Variable used to store number of lines of the first worksheet.</td>
</tr>
<tr>
<td>Spreadsheet</td>
<td>columns</td>
<td>Variable used to store the number of columns of the first worksheet.</td>
</tr>
<tr>
<td>Spreadsheet</td>
<td>headers</td>
<td>Variable used to store the list of attributes (i.e. headers) of the first worksheet.</td>
</tr>
<tr>
<td>Documents</td>
<td></td>
<td>The attributes are the same used in the file entity.</td>
</tr>
<tr>
<td>Presentations</td>
<td></td>
<td>The attributes are the same used in the file entity.</td>
</tr>
<tr>
<td>Images</td>
<td></td>
<td>The attributes are the same used in the file entity.</td>
</tr>
<tr>
<td>Worksheet</td>
<td>name</td>
<td>Variable used to store the worksheet name.</td>
</tr>
<tr>
<td>Worksheet</td>
<td>lines</td>
<td>Variable used to store the number of lines of the first worksheet.</td>
</tr>
<tr>
<td>Worksheet</td>
<td>columns</td>
<td>Variable used to store the number of columns of the first worksheet.</td>
</tr>
<tr>
<td>Worksheet</td>
<td>headers</td>
<td>Variable used to store the list of attributes (i.e. headers) of the worksheet.</td>
</tr>
<tr>
<td>Recycle Bin</td>
<td>listOfFiles</td>
<td>Variable used to store the list of files available in the Google Drive trash.</td>
</tr>
<tr>
<td>Recycle Bin</td>
<td>deletedTime</td>
<td>Variable used to store the time the file was deleted from the project.</td>
</tr>
<tr>
<td>Recycle Bin</td>
<td>userPermission</td>
<td>Variable used to store the user permission of each file available in Google Drive trash.</td>
</tr>
</tbody>
</table>

Table F.1: Class Domain Model - attributes description