



Escola d'Enginyeria de Telecomunicació i  
Aeroespacial de Castellet

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# MASTER THESIS

**TITLE:** ERP implementation for an administrative agency as a corporative Frontend and an e-commerce Smartphone App

**MASTER DEGREE:** Master in Science in Telecommunication Engineering & Management

**AUTHORS:** Vilma Karina Reynoso Vásquez and Jaume Font Escribano

**DIRECTOR:** Jesús Alcober

**DATE:** January, 17th 2017



**Título:** ERP implementation for an administrative agency as a corporate Frontend and an e-commerce Smartphone App

**Autor:** Vilma Karina Reynoso Vásquez y Jaume Font Escribano

**Director:** Jesús Alcober

**Fecha:** 17 de Enero del 2017

## Resumen

Este documento contiene todas las descripciones, argumentaciones y demostraciones de las búsquedas, análisis, razonamientos, diseños y tareas realizadas para conseguir el requerimiento de evolucionar tecnológicamente una gestoría de modo que pueda, a través de una solución que requiera una inversión reducida, disponer de una herramienta de gestión empresarial con e-commerce y una aplicación móvil que permita acceder y consultar dicha herramienta.

La primera parte del documento describe el escenario para contextualizar el proyecto y se hace una introducción a ERP (Enterprise Resources Planning). En la segunda parte se realiza un trabajo de investigación profundo de productos ERP de mercado, identificando las fortalezas y debilidades de cada uno, para finalizar con la elección del producto más adecuado para el escenario planteado en el proyecto.

En una tercera parte se describe el proceso de instalación del producto seleccionado llevado a cabo en base a la utilización de Dockers, así como las configuraciones y personalizaciones que realizan sobre el ERP seleccionado. También se realiza una descripción de la instalación y configuración de módulos adicionales, necesarios para lograr el alcance acordado del proyecto.

En una cuarta parte de la tesis, se describe el proceso de creación de una App

pasa iOS y Android que conecte con la base de datos del ERP seleccionado. El proceso mencionado empieza con el diseño de la App. Una vez diseñada, se explica el proceso de estudio y documentación de tecnologías para elegir el stack de tecnología que permita realizar una aplicación, robusta y actual sin uso de licenciamiento. Después de elegir las tecnologías a usar se explican las dependencias y la necesidad de instalar sistemas en tiempo de ejecución previo al inicio de la programación. Posteriormente, se describe como se ha planteado y desarrollado el código de la App. A continuación, se indican los mecanismos de compilación y comprobación. Y por último se muestra el resultado del desarrollo de la App una vez distribuido.

Por último, en un capítulo para las conclusiones se analizan las dificultades surgidas durante el proyecto y los logros conseguidos, analizando lo aprendido durante el desarrollo del presente proyecto.

**Title:** ERP implementation for an administrative agency as a corporative Frontend and an e-commerce Smartphone App

**Author:** Karina Vilma Reynoso Vásquez and Jaume Font Escribano

**Director:** Jesús Alcober

**Date:** January 17<sup>th</sup>, 2017

## Overview

This document contains all the descriptions, arguments and demonstrations of the researches, analysis, reasoning, designs and tasks performed to achieve the requirement to technologically evolve an managing agency in a way that, through a solution that requires a reduced investment, makes possible to arrange a business management tool with e-commerce and also a mobile application that allows access and consultation of mentioned tool.

The first part of the document describes the scenario in order to contextualize the project and introduces ERP (Enterprise Resources Planning). In the second part, a deep research of ERP market products is carried out, identifying the strengths and weaknesses of each one of the products in order to finish with the choice of the most suitable product for the scenario proposed in the project.

A third part of the document describes the installation process of the selected product carried out based on the use of Dockers, as well as the configurations and customizations that they make on the selected ERP. A description of the installation and configuration of additional modules is also made, necessary to achieve the agreed scope of the project.

In a fourth part of the thesis, the process of creating an iOS and Android App that connects to the selected ERP database is described. The process begins with the design of the App. Once designed, it is explained the process of study and documentation of technologies to choose the technology stack that allows making an application robust and contemporary without use of licensing. After

choosing the technologies to use there are explained the dependencies and needs to install runtime environments prior to the start of coding. Later, it describes how the code of the App has been raised and developed. The compilation and verification mechanisms are indicated in continuation. And finally, it is showed the result of the development of the App once distributed.

Finally, a chapter for the conclusions analyzes the difficulties encountered during the project and the achievements, analyzing what has been learned during the development of this project.

# INDEX

<b>INTRODUCTION.....</b>	<b>1</b>
<b>CHAPTER 1. INFORMATION, PLANNING AND MANAGEMENT SYSTEM</b>	<b>2</b>
<b>1.1 Introduction.....</b>	<b>2</b>
1.1.1 Study Case .....	3
1.1.2 Main objective of the project.....	3
<b>1.2 ERP Software System .....</b>	<b>4</b>
<b>CHAPTER 2. ERP SELECTION PROCESS.....</b>	<b>6</b>
<b>2.1 Benchmarking: ERP software system solution .....</b>	<b>6</b>
<b>CHAPTER 3. ODOO V9 SOFTWARE SYSTEM SOLUTION IMPLEMENTATION ON A MANAGING AGENCY .....</b>	<b>14</b>
<b>3.1 Managing agency: Business Needs .....</b>	<b>14</b>
3.1.1 Products .....	14
3.1.2 Operations .....	16
3.1.3 Workflow diagrams .....	18
<b>3.2 ODOO V9 Installation and Deployment .....</b>	<b>22</b>
3.2.1 Server setup .....	22
3.2.2 ODOO V9 Installation using Dockers .....	23
<b>3.3 ODOO V9 Modules setup.....</b>	<b>28</b>
3.3.1 Website.....	29
3.3.2 Ecommerce .....	31
3.3.3 External client login .....	33
3.3.4 Knowledge management system .....	34
<b>CHAPTER 4. MOBILE APPLICATION .....</b>	<b>36</b>
<b>4.1 Webapp Design.....</b>	<b>36</b>
<b>4.2 Technology stack .....</b>	<b>38</b>
<b>4.3 System requirements and software installation to develop Ionic 2 mobile application.....</b>	<b>41</b>
4.3.1 Android SDK set up .....	43
<b>4.4 CORS issues using Ionic 2 with Odoo V9.....</b>	<b>44</b>
<b>4.5 Coding the app .....</b>	<b>45</b>
<b>4.6 Connection to Odoo .....</b>	<b>46</b>
4.6.1 XML-RPC .....	47
4.6.2 Services published in the API.....	48
4.6.3 Configuration of the XML-RPC .....	49
4.6.4 Models definition.....	49
4.6.5 Creating providers .....	50
4.6.6 Pages coding.....	52

4.6.7	Compilation, execution and distribution.....	55
4.6.8	Publish the application and add new version with HockeyApp .....	56
4.6.9	Final result .....	59
<b>CHAPTER 5. CONCLUSIONS.....</b>		<b>61</b>
5.1	Global Enviromental impact.....	62



## INTRODUCTION

With diminishing finances, a small managing agency can rarely have enough resources to fulfill the needs for all its customers. Nowadays all the services offered by this kind of business represent an amount of physical documents and waste of resources to collect all the required information of its clients. An evolution of all these methods represents a number of challenges: volume of information resources; nature and quality of information; user needs and expectations; information and communication technology competencies and infrastructure; inflated cost of information resources; and staffing needs. This thesis reports the findings of a thorough study to establish the best ERP solution (Enterprise Resources Planning), to offers an e-commerce website where the clients could contract and have all the information of contracted products that deliver a managing agency, also this solution helps the client to have a file management system to save all the important documents as contracts, invoices, legal documents and so on. An Enterprise Resource Planning (ERP) is a business process management software which helps an organization to use a set of integrated applications for the business management and automatize some functions from the back office related to technology, services and human resources.

Emphasis is placed on the establishment of the most adequate solution of technology system that avoids the use of paper and reduces time and emissions due to transportation of the clients to a physical office to complete procedures. Also central to the research of mobile application technologies in order to develop the most adequate app for iOS and Android to connect to the ERP system implemented.

Furthermore during the developing of this project, there is an intense study of many new technologies along the chapters that help us to apply the desired environment proposed at the beginning of this thesis.

# CHAPTER 1. INFORMATION, PLANNING AND MANAGEMENT SYSTEM

## 1.1 Introduction

Nowadays new technologies play a big roll on the business world. A Managing Agency has a very elemental and basic level of technologic maturity. Although this kind of business has never been characterized for an advanced technological development, it is a fact that some agencies are offering basic services over internet. Some of these enterprises are new and defined as an *On-line Managing Agency*. Some others are the traditional ones which are adapting to the new era, and offer their services or part of them through a web page.

In a world where the clients dominate all the new technologies and appreciate all the advantages of the network (flexibility, availability, fastness...), it is a huge competitive advantage that all the users could manage their services, also hire new ones thought a web interface. Then, for an enterprise that wants to compete and earn share of the market with their competitors, it is mandatory to work with websites and applications that bring closer the business with the clients. Not only with the aim to attract new clients or increase the services acquisition, also for keeping the all the usual clients that every day are looking for other companies which are moving forward with all these technologic improvements.

As it is mentioned before, the Managing Agency business is not characterized for using all the new technologies advantages. They are characterized for using obsolete product, and simple process. The main reasons why this sector do not come forward with the rest of business are:

1. Managing Agency user/client profile.
2. Specialized software usage which limit the changes on the product process.
3. Historic Data Migration difficulties.

### **1.1.1 Study Case**

The study case is a company named GestClick SL, from now on GestClick. GestClick is a managing agency founded on 1981 at Barcelona, it has 12 employees. Since its foundation, the company has always been directed by its owner and main shareholder, Joan Fustè. Since approximately 9 months ago, his son, Xavier Fustè took the company as general director.

Xavier comes with the idea of raise the business, and make it grows by transforming the activities and the positioning in order to adapt the company to the new era.

The strongest points of the company are the services portfolio that always have cover all the clients' needs.

1. Enterprise services
2. Freelancers
3. LOPD
4. Trademark and logo

On the other hand, its biggest weakness is the obsolete technologies usage that represents the threat of losing clients because of a market evolution. Therefore, for Xavier, a project that include the information system improvement, it is considered a strategic project.

### **1.1.2 Main objective of the project**

The objective of the project is developing technologically the business of managing agency to update all the processes and service access to adapt this kind of company to all the new technologies.

It is proposed to build a web interface using an ERP system where all the clients could check all the corporative information about their company, view the products/ services portfolio and contract new products (e-commerce). It is planned also to develop a hybrid mobile application for Android and IOS for the clients to manage and check their features and data on mobility.

Because of the size of the company and the budget available for this project, it is required that the solution proposed does not represent a high cost of implementation and deployment neither recurrent high costs. This requirement affects to all the subjects as licensing, development, and adaptation and infrastructure costs.

It is also required a solution that allows scalability and the possibility to implement new modules as sales, accounting, etc.

## 1.2 ERP Software System

With the aim of achieve the purpose exposed on the introduction, it is proposed to use an ERP software system.

Enterprise Resource Planning (ERP) systems are core software programs used by companies to integrate and coordinate information in every area of the business. ERP (pronounced “E-R-P”) programs help organizations manage company-wide business processes, using a common database and shared management reporting tools. A business process is a collection of activities that takes one or more kinds of input and creates an output, such as a report or forecast that is of value to the customer. ERP software supports the efficient operation of business processes by integrating tasks related to sales, marketing, manufacturing, logistics, accounting, and staffing—throughout a business. In addition to this cross-functional integration, which is at the heart of an ERP system, companies connect their ERP systems, using various methods, to coordinate business processes with their customers and suppliers. In later chapters, you will learn how successful businesspeople use ERP programs to improve how work is done within a company and between companies<sup>1</sup>.

This kind of systems is usually called *back office*, because the clients are not involved with them. Meanwhile, the services offered by the *front office*, create an administrative relation to the consumer or the consumer services (CRM), a system that directly treats with the clients, or with the electronic business systems such as electronic commerce, electronic administrations, electronic telecommunications and electronic finance; but it is a fact that the ERP systems progressively have extended their modules to cover also CRM functions. Actually, the main CRM manufacturing companies have been absorbed by ERP software companies during the past 10 years.

---

<sup>1</sup> Monk, E.; Wagner, B. (2013). *Concepts in Enterprise Resource Planning*. Boston: Cengage Learning.

The most common ERP system modules broadly used in many companies are manufacturing or production, storage, logistics, technologic information, accounting, human resources, marketing and strategic administration.

## CHAPTER 2. ERP SELECTION PROCESS

### 2.1 Benchmarking: ERP software system solution

Before the implementation of the system solution, it is necessary to select the adequate product that covers all the features needed to accomplish the project requirements.

According to the needs explained at CHAPTER 1, the ERP software solutions considered are analyzed at the Table 2.1:

Open source ERP Software	
<i>OpenERP</i>	
Website	<a href="https://www.odoo.com/es_ES/">https://www.odoo.com/es_ES/</a>
Modules and features	<ul style="list-style-type: none"> <li>▪ Customer relation management (CRM)</li> <li>▪ Project management</li> <li>▪ Storage management</li> <li>▪ Accounting and finance management</li> <li>▪ Shopping management</li> <li>▪ Sales management</li> <li>▪ Human resources</li> <li>▪ Marketing</li> <li>▪ Manufacturing</li> <li>▪ Point of sale</li> <li>▪ Knowledge management</li> </ul>
Supported database	PostgreSQL
Operating System	Linux and Windows
Programming language	Python
Architecture	<p>To access to OpenERP V9 only have to use the website with the server IP address where is the web-client.</p> <p>The OpenERP system is composed by three main components:</p> <ul style="list-style-type: none"> <li>▪ The PostgreSQL data base server, which contains all the data bases, each one of them has all the data and most of the configuration elements related with the OpenERP system.</li> </ul>

	<ul style="list-style-type: none"> <li>▪ The OpenERP application that contains all the enterprise logic and assure that the ERP runs optimally.</li> </ul> <p>The web server, a separated web application called Open Object, which allows to connect to the OpenERP from any standard web browser.</p>
Small and medium-sized business and Enterprise organizations	Both
Implementation areas	Sanity, hotels, transportation, civil engineering, associations, textile manufacturing, food industry, education centers, restaurants, and many others.
Last update available and support service	Last version released was version 9
Geographic availability	Worldwide
Partners	It has more than 500 partners. <a href="https://www.odoo.com/es_ES/partners">https://www.odoo.com/es_ES/partners</a>
Support documentation available	yes, at their website
<b>OpenBravo</b>	
Website	<a href="http://www.openbravo.com/es/">http://www.openbravo.com/es/</a>
Modules and features	<ul style="list-style-type: none"> <li>▪ Products management</li> <li>▪ Supply chain management</li> <li>▪ Multi-channel management</li> <li>▪ Corporative management</li> <li>▪ Reporting and Analytics</li> <li>▪ Mobile, web and cloud platform</li> <li>▪ Web point of sale</li> <li>▪ Technologic partners</li> </ul> <p><b>Other sectors:</b></p> <ul style="list-style-type: none"> <li>▪ Business processes</li> <li>▪ User interface</li> <li>▪ Security model</li> <li>▪ Mobility</li> <li>▪ Interoperability</li> <li>▪ Process automatization and BPM</li> <li>▪ Model Directed Development (MDD)</li> <li>▪ Development platform</li> <li>▪ Scalability and development</li> </ul>
Supported database	PostgreSQL and Oracle

Operating System	Windows, Linux, Unix, Solaris, FreeBSD
Programming language	Java, HTML, XML and SQL
Architecture	<p>MVC (Model View Controller) an architecture for build applications that separate the data model (model) of the user interface (View) and the processing (controller). With OpenBravo, the MVC implementation is detailed:</p> <ul style="list-style-type: none"> <li>▪ The <b>Model</b> is archived using the OpenBravo SqlC tool. The input is a XML file that contains the standard SQL sentence, and the parameters used with them. These parameters could be operatives or not, and facilitate the Sql sentence generation needed.</li> <li>▪ The <b>View</b> is used the XMLEngine developed by OpenBravo to design the user interface. XMLEngine is a tool used to create XML/HTML documents from a template with XML/HTML format.</li> </ul> <p>The <b>Controller</b> used by the OpenBravo framework is builds by java classes that are extended from HttpBaseServlet. These servlets make a data lecture using the classes generated by SQLC and providing the XMLEngine output.</p>
Small and medium-sized business and Enterprise organizations	Both
Implementation areas	Openbravo has clients in all kind of business areas.
Last update and support service	The last version available is 3.0, and they offer support service
Geographic availability	Worldwide
Partners	<p>All the partners are detailed at the follow link:</p> <p><a href="http://www.openbravo.com/es/partners/">http://www.openbravo.com/es/partners/</a></p>
Support documentation available	All documentation is available at their wiki



Compiere	
Website	<a href="http://www.compiere.com/">http://www.compiere.com/</a>
Modules and features	<p>It includes all the features and modules of a ERP software, but in order to not duplicate information and the need of synchronization, it is organized in a different way. The product modules are:</p> <ul style="list-style-type: none"> <li>▪ Quote to Cash</li> <li>▪ Requisition-to-Pay</li> <li>▪ CRM</li> <li>▪ Partner relations management</li> <li>▪ Supply chain management.</li> <li>▪ Performance analysis</li> <li>▪ Stock management</li> <li>▪ Double-entry Book-keeping</li> <li>▪ Workflow management and online store</li> </ul> <p>It is developing a module of manufacturing as independent CMPCS project.</p>
Supported database	<p>Since the version 2.5.2, Compiere is independent from the database. There is an infrastructure design to connect with multiple data bases. The connectivity with these data bases: PostgresQL, MySQL and Sybase could be available or in process for implementation, but it is not available directly with the Compiere, that only support Oracle data base as official.</p> <p>Compiere also could be run using Firebird database using the Fyrcle extensions, without export. Also it works with the alternative of open code for Oracle data base, EnterpriseDB.</p>
Operating System	Unix-like, Windows
Programming language	Java
Architecture	Compiere is developed with an architecture driven by models (Model Driven Architecture), this is a structure designed with the intention of making changes for the business evolution. In any stage of the process the clients can change the information structure to adapt to the

	<p>new needs on the business.</p> <p>This structure allows the flexibility and integration of the supplementary external information. The information is presented as views (using the architecture MVCCompiere), that could be changed to satisfy the needs of the company. Compiere is entirely based in the concept of Active Data Dictionary (ADD). The Compiere ADD contains the definitions of a data entity (type, validation, etc.), the way that is visualized (labels on the screen and reports, help, showing the sequence and the position with other fields), and the visualization rules. Also it contains security and access rules.</p> <p>Compiere has been developed with Java EE.</p>
Small and medium-sized business and Enterprise organizations	Small and medium-sized business
Implementation areas	Distribution, sales, manufacturing, publishing, health care and pharmaceutical, government, ONG and benefic associations.
Last update available and support service	The last version (R3.3.0) was released at el 01-06-2010. It does not offer support service.
Geographic availability	Worldwide
Partners	<p>All the partners are detailed at the follow link:</p> <p><a href="http://www.compiere.com/partners/partner-directory/index.php">http://www.compiere.com/partners/partner-directory/index.php</a></p>
Support documentation available	Yes, at their website.
<b>NeoGia</b>	
Website	<a href="http://www.neogia.org/">http://www.neogia.org/</a>
Modules and features	<p>Neogia is a ERP solution based on OFBiz framework. It is based on the UML design and the code generation from UML modules as strong points.</p> <p>OFBiz Modules:</p> <ul style="list-style-type: none"> <li>Accounting (component that partially loads at the memory when OFBizNeogia launches)</li> </ul>

	<ul style="list-style-type: none"> <li>▪ Web contents management</li> <li>▪ e-Commerce</li> <li>▪ Stock management (facility), (component that partially loads at the memory when OFBizNeogia launches)</li> <li>▪ Human resources</li> <li>▪ Manufacturing (component that partially loads at the memory when OFBizNeogia launches)</li> <li>▪ Marketing</li> <li>▪ Sales (*)</li> <li>▪ CRM – SRM (*)</li> <li>▪ Point of sale</li> <li>▪ Products (*)</li> <li>▪ Work effort</li> </ul> <p>Modules:</p> <ul style="list-style-type: none"> <li>▪ Accounting (**)</li> <li>▪ Human Resources (***) (alpha version)</li> <li>▪ Manufacturing management (***)</li> <li>▪ Stock management (**)</li> <li>▪ Quality (***)</li> <li>▪ CRM (***)</li> <li>▪ Shipment (*) (based on OFBiz screen)</li> </ul> <p>(*): Use a part of the OFBiz module.  (**): Replace the OZBiz Module.  UML Diagram (data base diagram).  (**): Neogia Module.</p>
Supported database	
Operating System	Windows, Linux and Unix
Programming language	
Architecture	
Small and medium-sized business and Enterprise organizations	
Last update available and support service	The last version of Neogia was release at 2006
Geographic availability	
Partners	
Support documentation available	
<b>Proprietary ERP Software</b>	
<i>Navision</i>	
Website	<a href="http://www.microsoft.com/es-es/dynamics/erp.aspx">http://www.microsoft.com/es-es/dynamics/erp.aspx</a>

Modules and features	<p>Microsoft Dynamics NAV covers the following areas:</p> <ul style="list-style-type: none"> <li>▪ Finance management.</li> <li>▪ Sales and Marketing.</li> <li>▪ Sales.</li> <li>▪ Stock.</li> <li>▪ Manufacturing.</li> <li>▪ Projects.</li> <li>▪ Resource Planning.</li> <li>▪ Services.</li> <li>▪ Human Resources.</li> </ul> <p>The Microsoft Dynamics NAV code is accessible with the proper license, this is why the product is very manageable. This feature makes possible create new functional area and add them to the system, to complement the standards functionalities. Also is possible to add new functionalities to existing areas. In fact, there are more than 2000 developed and registered solutions.</p>
Supported database	Windows Server 2008/2012, Windows 7 and forward
Operating System	SQL Server
Programming language	C/AL
Architecture	
Small and medium-sized business and Enterprise organizations	Both
Implementation areas	Sales, marketing, social, services, customer service, manufacturing, finance enterprises.
Last update available and support service	They offer different support service plans available their website.
Geographic availability	Worldwide
Partners	<p>They have partners worldwide; all partners available at this link:</p> <p><a href="https://pinpoint.microsoft.com/en-us/search?type=companies&amp;keyword=Microsoft+Dynamics+ERP&amp;market=&amp;competency=100007&amp;page=0&amp;geoRadius=5">https://pinpoint.microsoft.com/en-us/search?type=companies&amp;keyword=Microsoft+Dynamics+ERP&amp;market=&amp;competency=100007&amp;page=0&amp;geoRadius=5</a></p>
Support documentation available	<p>All the support documentation is at:</p> <p><a href="https://www.microsoft.com/en-us/dynamics/support.aspx">https://www.microsoft.com/en-us/dynamics/support.aspx</a></p>

SAP AG	
Website	<a href="http://www.sap.com/spain/pc/bp/erp.html">http://www.sap.com/spain/pc/bp/erp.html</a>
Modules and features	
Supported database	
Operating System	
Programming language	
Architecture	
Small and medium-sized business and Enterprise organizations	Both
Implementation areas	Sales, marketing, social, services, customer service, manufacturing, finance enterprises.
Last update available and support service	They offer support service for developers.
Geographic availability	Worldwide
Partners	They have partners worldwide; all partners available at: <a href="http://go.sap.com/partner.html">http://go.sap.com/partner.html</a>
Support documentation available	All the support documentation is available at their website

**Table 2.1** Benchmark of the most important ERP products in the market

Within the selection of solutions evaluated, and after analyze the pros and cons of each product; the most adequate software solution is OpenERP (Odoo).

The reasons why this product is the proper one for the projects are:

1. It is an open source product: There is no budget available for this project and it is not considered license expenses for a property product.
2. Versatility: considering the open source products, it is the solution that has more modules. It is true that most of them are not going to be used during this project, but it is important to consider the possibility of extend the project in the future.
3. Support: Odoo has the broadness worldwide support. And with partners all over the world.
4. Easiness: It is true that this is a solution with high potential and capacity, but is a product with a very simple architecture, because it has BBDD, application engine and a web server to have web Access.

## **CHAPTER 3. ODOO V9 SOFTWARE SYSTEM SOLUTION IMPLEMENTATION ON A MANAGING AGENCY**

### **3.1 Managing agency: Business Needs**

As mentioned in Chapter 1, the reasons for requiring the implementation of an ERP with those characteristics are diverse. Mainly, apply an update of the operations and processes of the company to face a new business paradigm much more oriented to new market standards that could be defined as "anything, anywhere". The aim of this achieving is to expand the customers target and increase customers share in a market that slowly pivots to innovation.

Thus, it is important to set properly the business needs in order to guarantee that the ERP system contains the products, operations and workflows necessary to achieve these objectives previously mentioned.

Below, business requirements relating to products, operations and workflows are detailed.

#### **3.1.1 Products**

The products that the business development department has decided to offer, both through e-commerce site and the web application are the following:

##### *3.1.1.1 Enterprise Organization services*

This product includes all the advising services related to accounting, taxation and labor and legal matters. It is intended for enterprise organizations with more than 50 employees.

The product price is an on-going amount of 65€ per month, which includes all the services listed in the Annex. This product does not have conditions of permanence and it can be unsubscribed at any time by the client. In addition, the first month is for free if the service is subscribed online.

### *3.1.1.2 Freelance*

This product has two variants, the basic pack including accounting and tax advising and registration/removal on the freelance log file. The full pack in addition to the basic services package includes billing control, and accounting advising. This product is aimed to self-employed professionals.

The product price is an on-going amount of 25€ per month for the basic package and 45€ per month for the full package. The details of the services included in both versions can be found in the Annex. This product does not have conditions of permanence and it can be unsubscribed at any time by the client. In addition, the first month is for free if the service is subscribed online.

### *3.1.1.3 Trademark registration and logo*

This product includes all the necessary procedures for the registration of a trademark in the Spanish Office of Patents and Trademarks.

The product price is 195€ for each mark registration or renewal thereof. The registration lasts 10 years. All kind of clients, both freelancer and corporate organizations can hire this product.

### *3.1.1.4 LOPD Compliance*

This product consists in performing the following services related to compliance with the Organic Law on Data Protection (LOPD):

- Registration of files in the Spanish Data Protection Agency
- Development of mandatory Security Document within your business
- Clauses in contracts providing access to data by third parties
- If that has video surveillance, preparation of the necessary documentation and information and deliver of mandatory badges.

This product costs 80€ for each time advice or review of LOPD compliance are needed.

### 3.1.1.5 Products summary

Through the ERP system a total of 5 products are offered to our clients. These products will be one-time or on-going as can be seen in the Table 3.1 **Comparative of products per duration of the service:**

Product	One-time	On-going
[Enterprise] Full package		X
[Freelance] Full package		X
[Freelance] Basic package		X
LOPD	X	
Trademark	X	

**Table 3.1** Comparative of products per duration of the service

The one-time products are billed only once and the client can hire the product as many times as needed. On-going products consist in a service that is billed monthly if the customer has contracted product.

## 3.1.2 Operations

Once the products to offer through the website and mobile application are defined, the operations that users can perform must be defined.

The operations defined for project implementation are as follows:

### 3.1.2.1 New user

It consists in creating a new user to access to the customers area of the website and to access to the application. In this operation is requested basic information about the user as Name, Last Name, e-mail, phone number and password access.

### 3.1.2.2 User removal

It consists in the removal in the system of all user data. From this point the user cannot access the system and cannot recover the user. If the user needs access to the system again they must request a new user.



### 3.1.2.3 New organization

It consists in creating a business account for services that require it. When creating the new organization, the following information is requested: company name, VAT number, telephone number, administrator name, name of CEO, CFO's name and account number to bill for services will be requested.

In addition, the necessary number of users may be associated to the organization so that they can create new service requests and upload documents to the application.

### 3.1.2.4 Organization removal

It consists in the elimination of all the business account data in the system. To be able to unregister an organization all the products have to be terminated previously. It is not the same case with the users associated to the organization, is not necessary to remove them, although they will be detached from that organization.

### 3.1.2.5 Product/service hiring

It consists in the acquisition by a user or an organization of any of the products described in the Section 3.1.

The type of contract is defined by the type of product as can be seen in the Table 3.2.

Product	Organization	User
[Enterprise] Full package	X	
[Freelance] Full package		X
[Freelance] Basic package		X
LOPD	X	X
Trademark	X	X

**Table 3.2** Comparative of products by type is client

Within the products hiring must be validated the billing information in cases of on-going services and payment must be validated in one-time services.

#### *3.1.2.6 Product/service termination*

It consists in the removal of one of the contracted products. Only on-going services can be terminated.

For one-time services, the product will be terminated automatically in the system a month after the hiring of the product.

#### *3.1.2.7 Service Request*

It consists in the creation of a request for advising or management within the types of requests that include contracted services. Service Requests only apply to customers who have contracted any of the on-going services.

#### *3.1.2.8 Documentation uploading*

It consists in the functionality that allows users to provide documentation relating to a request. The documents uploaded to the system will always be in PDF format and cannot exceed 2 MB.

#### *3.1.2.9 Invoicing*

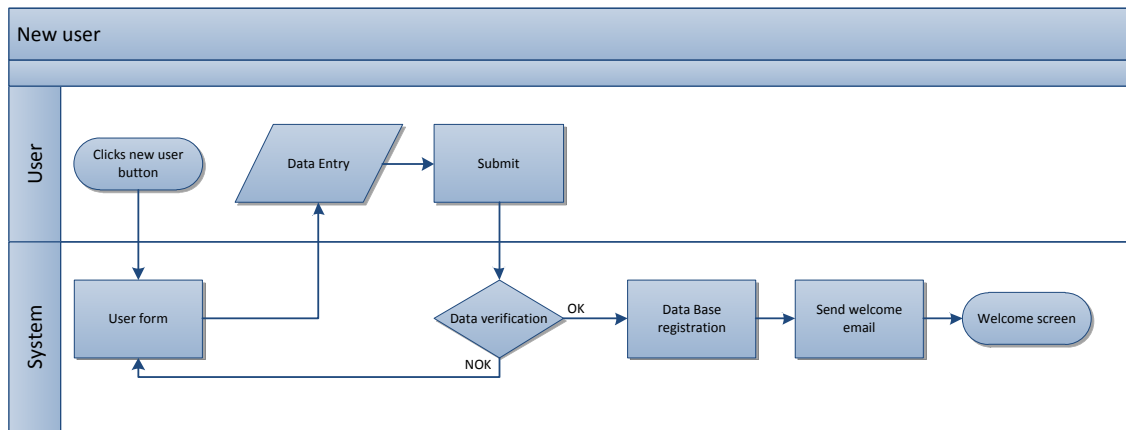
It consists in the operation that automatically generates and sends an invoice to the customer. It calculates the total amount that one client have to pay monthly and generates an invoice and a charge to the account of the customers.

### **3.1.3 Workflow diagrams**

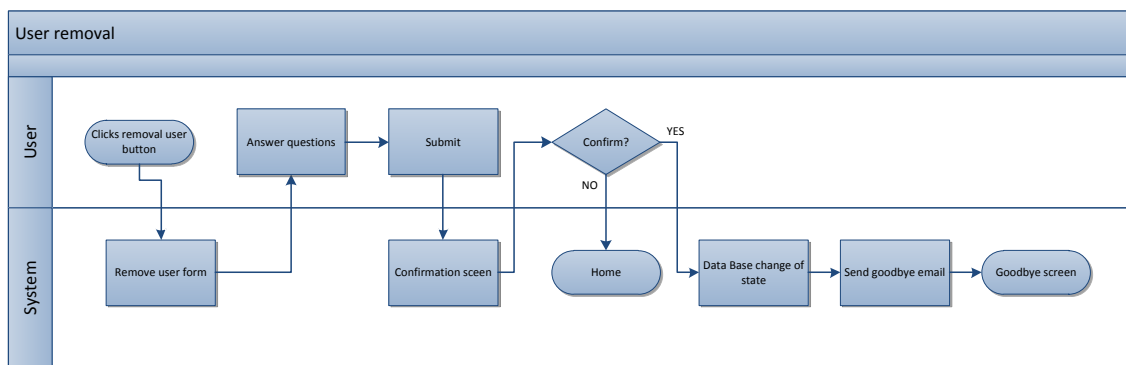
Once defined the requirements at the level of products and operations by business, it is necessary to define the workflows to be implemented for each of the operations.

Then the flows will need to implement in the application for each of the operations are defined:

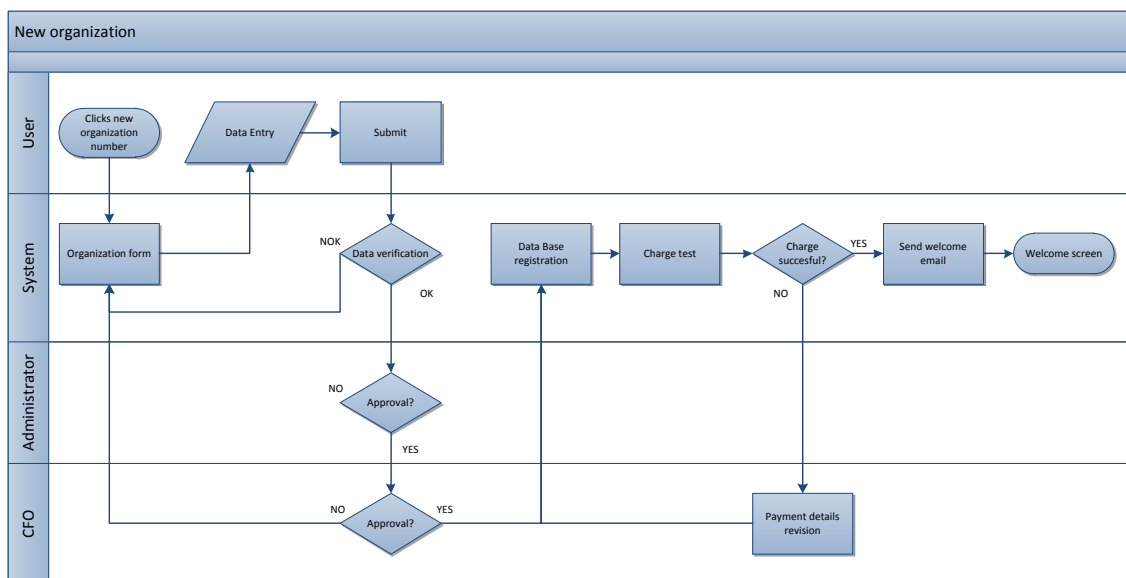
- New user



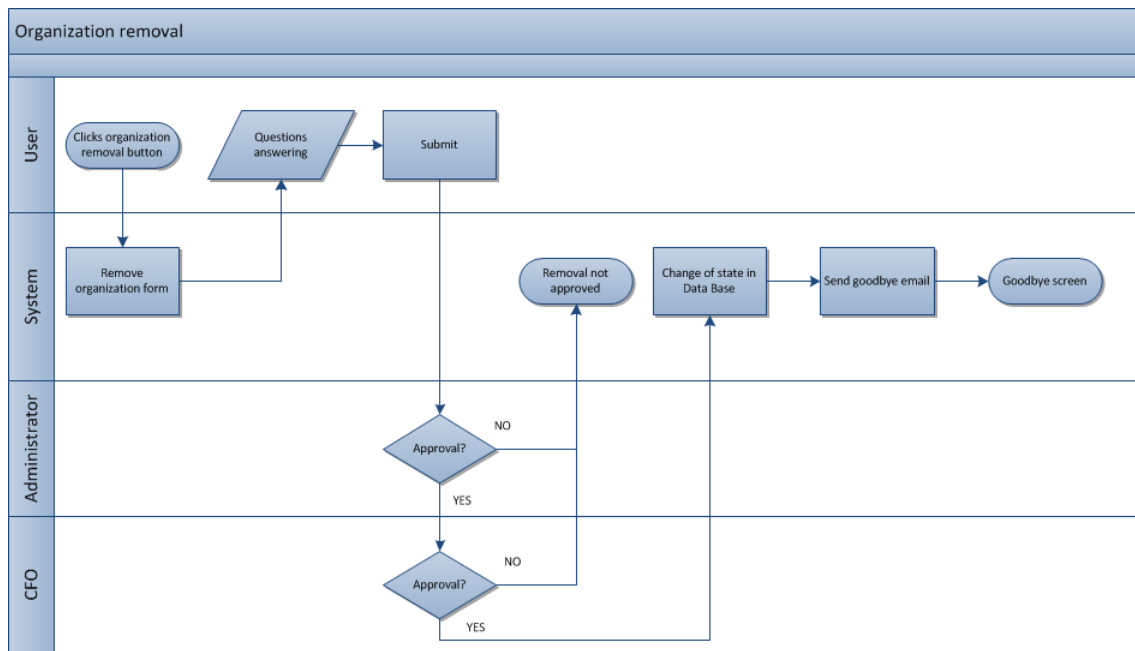
- User removal



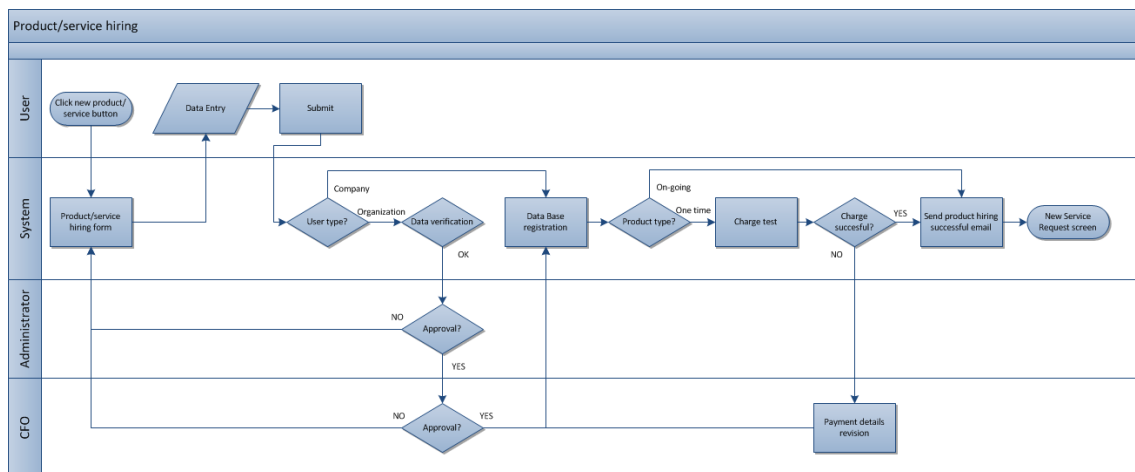
- New organization



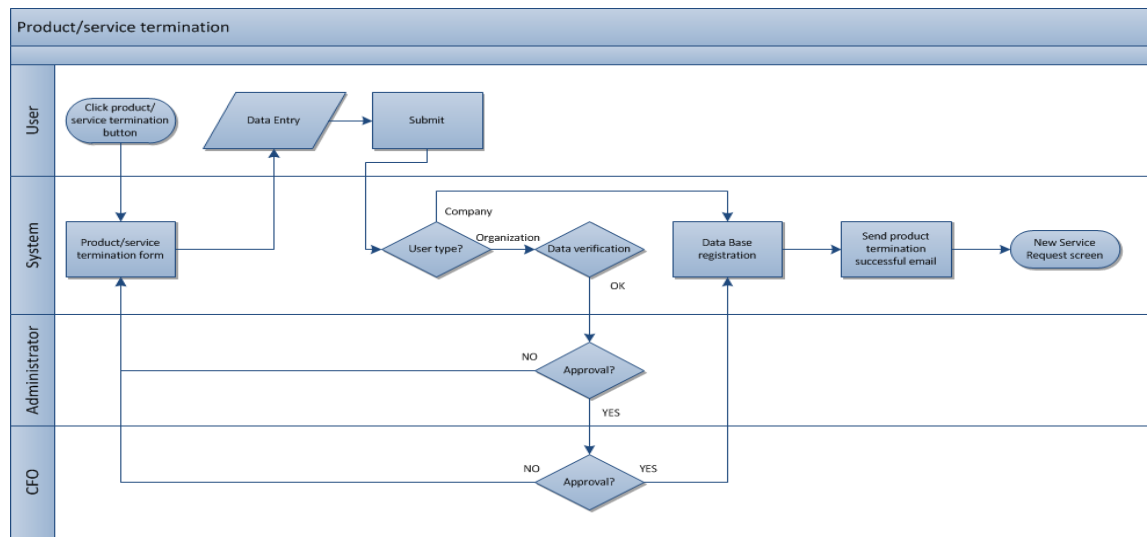
- Organization removal



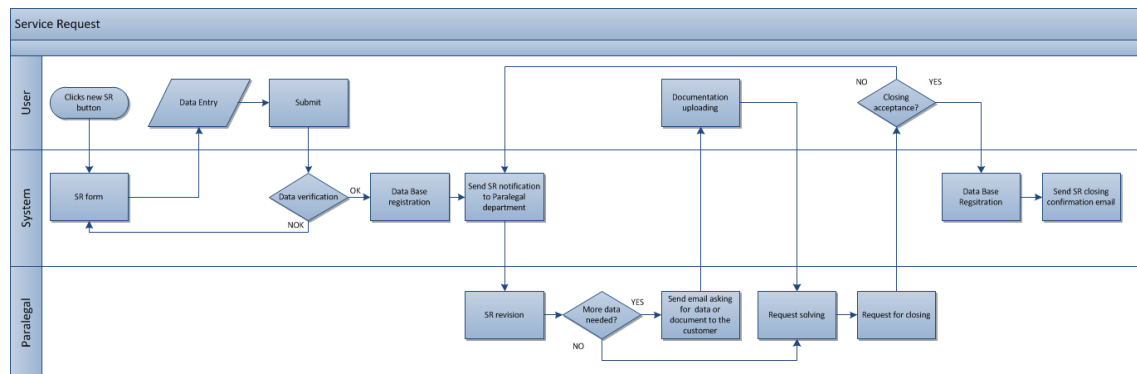
- Product/service hiring



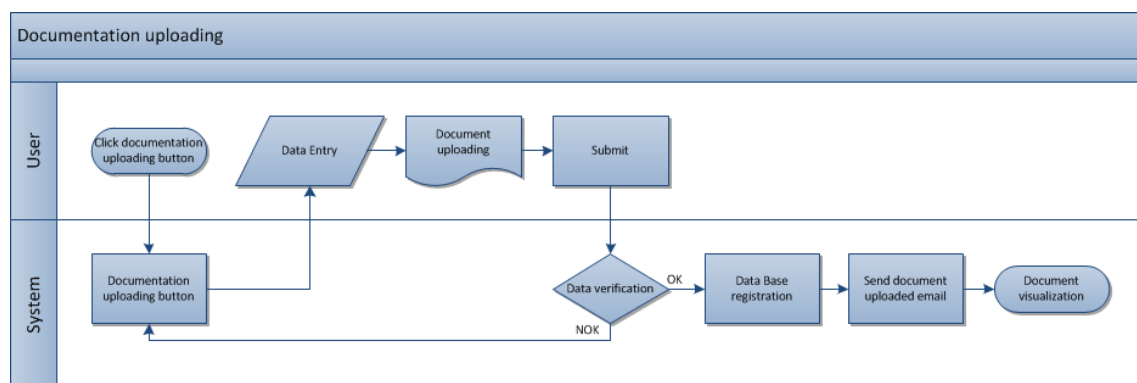
- Product/service termination



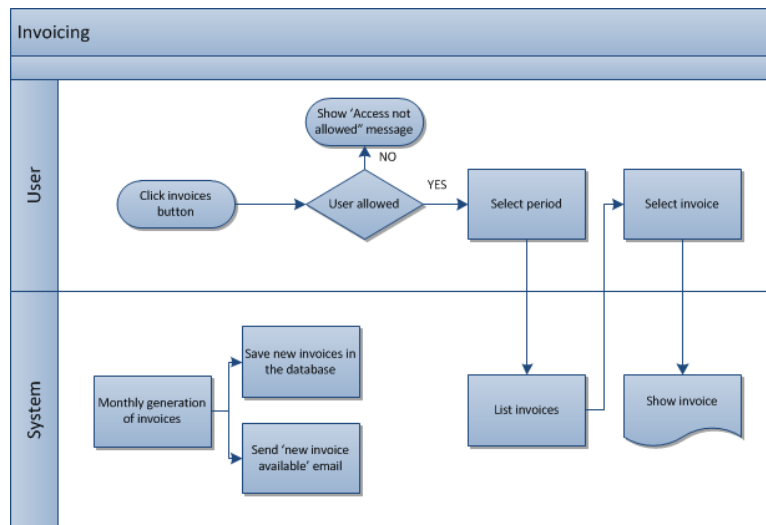
- Service request



- Documentation uploading



- Invoicing



## 3.2 ODOO V9 Installation and Deployment

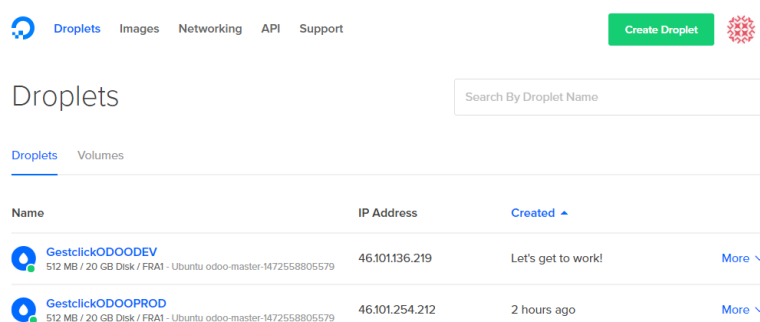
The server was contracted using Digital Ocean, it is a cloud server. Digital Ocean offers a service of Droplet, that are virtual server images already predetermined.

In order to create the new image, it is necessary to know the requirements that should have the system in order to have Odoo Version 9 perfectly running.

- Operating System: Linux, Ubuntu 14.04
- Disk Space:  $\geq 20\text{GB}$
- Memory:  $\geq 512\text{ MB}$
- Transmission data:  $\geq 1\text{GB}$
- Weekly backup

### 3.2.1 Server setup

Once it is established all the minimum requirements of the system, we create two droplets with an Ubuntu server with all these requirements, one for production named GestclickODOOPROD, and other one for development named GestclickODOODEV as we can observe in the Fig. 3.1.



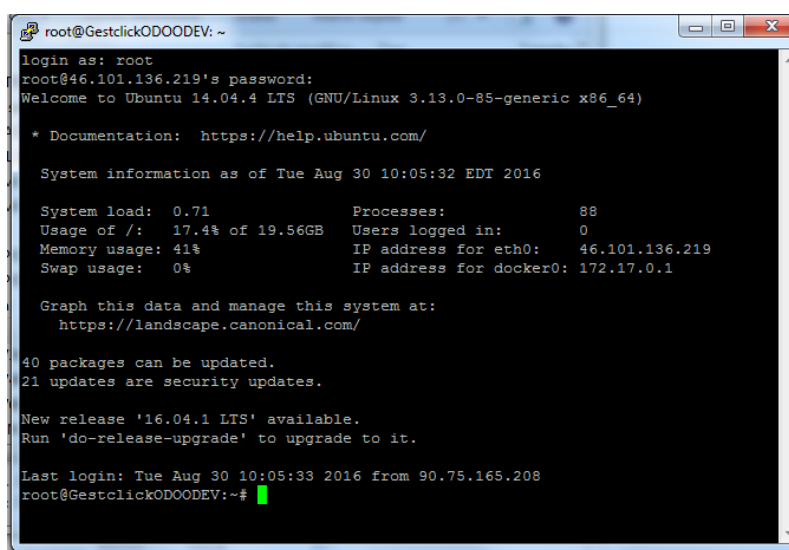
**Fig. 3.1** Servers at One Ocean

The IP address of GestclickODOODEV is **46.101.136.219**, once the development server is set up and properly working can be created another production server with a snapshot from GestclickODOODEV, and set the GestclickODOOPROD.

*SSH credentials:*

*User: root*

*Password: gestclick123*



**Fig. 3.2** Screenshot of the access to the development server

### 3.2.2 ODOO V9 Installation using Dockers

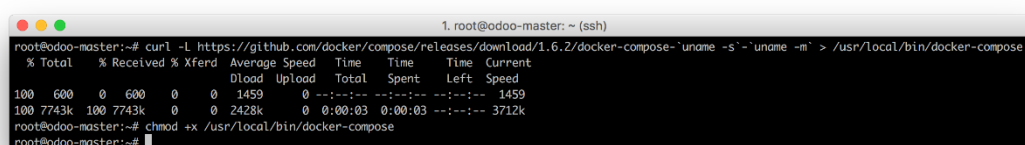
For this project are used pre-designed Docker containers to setup the Odoo structure required for each server. Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code,

runtime, system tools, and system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in. Containers have similar resource isolation and allocation benefits as virtual machines but a different architectural approach allows them to be much more portable and efficient.

Both servers, as it is mentioned before, are set up with Ubuntu as a virtual machine. To setup correctly all the Docker containers needed, it is used a tool of Dockers named Docker Compose. Docker compose is used for development, testing, and staging environments, as well as CI workflows.

In the Fig. 3.3 it is shown how we download the Docker compose tool from GIT (curl -L <https://github.com/docker/compose/releases/download/1.6.2/docker-compose-`uname -s`-`uname -m`> > /usr/local/bin/docker-compose).

After download the docker compose to the bin folder, it is necessary to set the executable permissions to this tool (chmod +x). All the information about the installation and the proper utilization of the docker compose can be find at <https://docs.docker.com/compose/>.



```

1. root@odoo-master: ~ (ssh)
root@odoo-master:~# curl -L https://github.com/docker/compose/releases/download/1.6.2/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 600      0 600    0     0  1459      0 --:--:-- --:--:-- --:--:--  1459
100 7743k 100 7743k    0     0 2428k      0  0:00:03  0:00:03 --:--:-- 3712k
root@odoo-master:~# chmod +x /usr/local/bin/docker-compose
root@odoo-master:~#

```

**Fig. 3.3** Downloading of Docker Compose tool to the server

For each one of our servers, production and development, we will need to setup three different containers. One with the Odoo server, version 9, another one with a Postgres database, default database for Odoo, and an NginX proxy used to solve some problem that came out during the mobile application development.

Using Compose is basically a three-step process:

1. Define the app environment with a **Dockerfile**, so it can be reproduced anywhere. Because server structure, the docker-compose will execute the Dockerfile to specify which image it will be used to setup the nginx proxy container, previously defined at the *default.conf* file. The nginx proxy image available online, do not solve the problem detailed at Section 5.3.



### Dockerfile content:

```
FROM nginx
ADD default.conf /etc/nginx/conf.d/default.conf
```

The default.conf file has a custom designed NginX proxy settings that satisfy all our needs. The following file, add headers “\$cors\_header” to all the package receive at the server 46.101.136.219:80 (internet), and create a translation to 46.101.136.219:8069 (Odoo default hearing port) which inabilities CORS, 'Access-Control-Allow-Origin'.

### Default.conf content:

```
map $http_host $cors_header {
    hostnames;
    default "$http_origin";
}
server{
    listen 80;
    server_name localhost;
    location / {
        proxy_pass http://46.101.136.219:8069;
        add_header 'Access-Control-Allow-Origin' $cors_header;
        add_header 'Access-Control-Allow-Methods'
'POST,GET,OPTIONS,DELETE,PUT,HEAD';
        add_header 'Access-Control-Allow-Credentials' 'true';
        add_header 'Access-Control-Allow-Headers' 'content-type';

        if ($request_method = 'OPTIONS') {
            add_header 'Access-Control-Max-Age' '1728000';
            add_header 'Content-Type' 'text/plain charset=UTF-8';
            add_header 'Content-Length' '0';

            add_header 'Access-Control-Allow-Origin' $cors_header ;
            add_header 'Access-Control-Allow-Methods'
'POST,GET,OPTIONS';
            add_header 'Access-Control-Allow-Credentials' 'true';
            add_header 'Access-Control-Allow-Headers' 'content-type';

            return 204;
        }
    }
}
```

2. Define the services that make up the app in **docker-compose.yml** so they can run together in an isolated environment. At this file it is defined the settings for the three containers used in the project.

The docker-compose.yml defines a container with an nginx proxy, named **nginx** (container\_name). As it is mentioned at point 1, the available image for the nginx proxy does not solve the problem that we detailed at section 5.3, then it was necessary to run the custom nginx setting defined at the default.conf. It is only specified that everything that is accessible from the port 80 (internet) will be translated to the same port, (port: "80:80").

Also at the same file it is set another container named **odoo** (container\_name), with an Odoo server version 9 image (image), available through port 8069, Odoo default port, (ports: "8069:8069"). Also for this docker it is necessary to create a link to the Postgres database (link: "db:db").

And for the last it is essential to create another docker for the database, named **db** (container\_name). It is used a Postgres database image (image: "Postgres 9.5.1"), that is the database supported by Odoo, with 2 volumes available inside.

In order to stablish the connection for the Odoo container to the db, it is included in both containers settings the same database credentials (link: "db:db").

Docker-compose.yml content:

```
nginx:
  build: .
  container_name: nginx
  restart: always
  ports:
    - "80:80"
odoo:
  image: "odoo:9.0"
  container_name: odoo
  restart: always
  ports:
    - "8069:8069"
  links:
    - db:db
  environment:
    - PGHOST=db
    - PGUSER=odoo
    - PGPASSWORD=5scWK52p72t2f2Q
db:
  image: "postgres:9.5.1"
  container_name: db
  restart: always
  environment:
    - POSTGRES_USER=odoo
    - POSTGRES_PASSWORD=5scWK52p72t2f2Q
  volumes:
    - "/var/local/postgresql/data:/var/lib/postgresql/data"
    - "/var/log/postgresql:/var/log/postgresql"
```

3. Lastly, run **docker-compose up** and Compose will start and run your entire app as it is shown in the Fig. 3.4.

```

root@odoo-master:~/odoo# docker-compose up -d
Pulling db (postgres:9.5.1)...
9.5.1: Pulling from library/postgres
fdd5d7827f33: Pull complete
a3ed95cae02: Pull complete
beb59dc2ad34: Pull complete
f42a5322ef13: Pull complete
f6719ae287c6: Pull complete
0dc08677d778: Pull complete
5f3b03c1dd66: Pull complete
4d4c6707d860: Pull complete
11f8efebbb9b: Pull complete
edefda034373: Pull complete
5fe4bba523f0: Pull complete
a2d55ee03342: Pull complete
Digest: sha256:f6a2b81d981ace74aeafb2ed2982d52984d82958bfe836b82cbe4bf1ba440999
Status: Downloaded newer image for postgres:9.5.1
Creating db
Pulling odoo (odoo:9.0)...
9.0: Pulling from library/odoo
51f5c6a04d83: Pull complete
a3ed95cae02: Pull complete
f36ca6a6d06f: Pull complete
0549b14593ad: Pull complete
3207a3c8a001: Pull complete
95b272482e12: Pull complete
b354067cc8c2: Pull complete
51a3c5033afa: Pull complete
Digest: sha256:17c8d35234b2a301cd71ff3dad8713e608e90e790f4361be3566cdfb5abb3087
Status: Downloaded newer image for odoo:9.0
Creating odoo
root@odoo-master:~/odoo#

```

**Fig. 3.4** Execution of the Docker-compose

All the dockers running in a machine can be check with the command: `docker-compose ps`. The Fig. 3.5 shows the three dockers running in the development server.

```

root@GestclickODOODEV: ~/odoo
login as: root
root@46.101.136.219's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-85-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Wed Nov 23 12:25:45 EST 2016

System load: 0.04          Processes:           93
Usage of /:  19.5% of 19.56GB   Users logged in:       0
Memory usage: 74%           IP address for eth0:    46.101.136.219
Swap usage:  0%              IP address for docker0: 172.17.0.1

Graph this data and manage this system at:
https://landscape.canonical.com/

77 packages can be updated.
44 updates are security updates.

Last login: Wed Nov 23 12:25:58 2016 from 62.81.73.50.static.user.ono.com
root@GestclickODOODEV:~# cd odoo
root@GestclickODOODEV:~/odoo# docker-compose ps

```

Name	Command	State	Ports
db	/docker-entrypoint.sh postgres	Up	5432/tcp
nginx	nginx -g daemon off;	Up	443/tcp, 0.0.0.0:80->80/tcp
odoo	/entrypoint.sh openerp-server	Up	0.0.0.0:8069->8069/tcp, 8071/tcp

```

root@GestclickODOODEV:~/odoo#

```

**Fig. 3.5** Docker information

Now it is possible to access via web through a browser to our already setup Odoo server as is shown at Fig. 3.6 and Fig. 3.7:

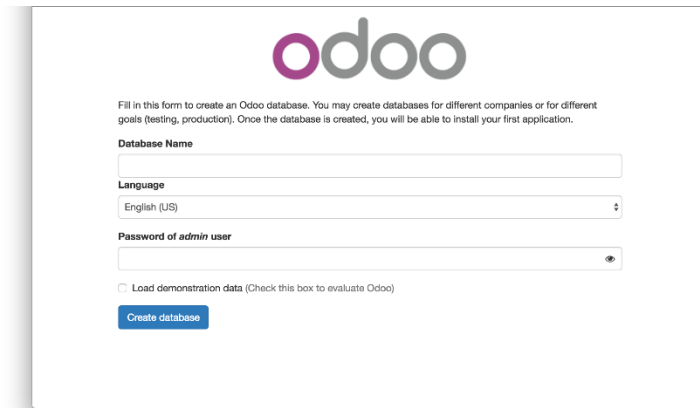


Fig. 3.6 Odoo login screen

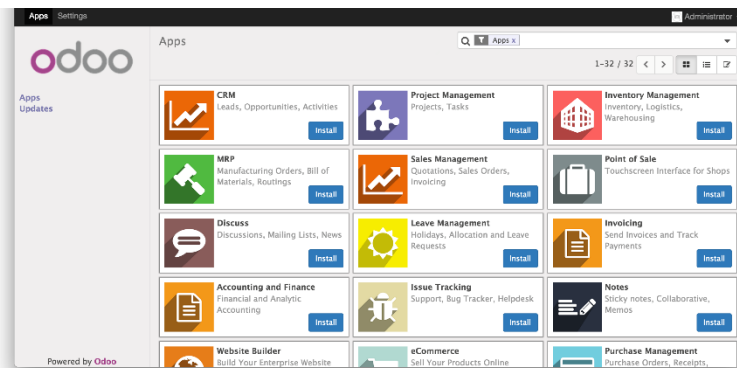


Fig. 3.7 Modules installation menu

### 3.3 ODOO V9 Modules setup

Once the Internet access is available, to install all the modules needed for the project it is necessary to login as administrator (login and password already established at the server website connection).

Administrator user: *gestclickagency@gmail.com*

Password: *gestclick123*

For a managing agency business that it is pretended to automatize itself offering all the services online, also with the client profile and file management, it is indispensable to have installed the following modules in the Odoo:

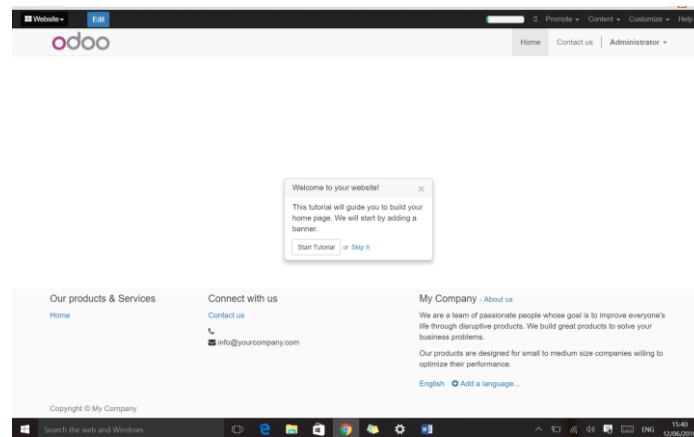
- 1- Website Builder
- 2- E-commerce (by default with this module, the system needs also the sales management module and invoicing module)
- 3- External client authentication
- 4- Knowledge management system

According to the business needs defined at Section 3.1 the direction of the project and the members have decided to focus on the following operations:

- Product/service hiring
- Service request
- Documentation uploading/File management
- Invoicing

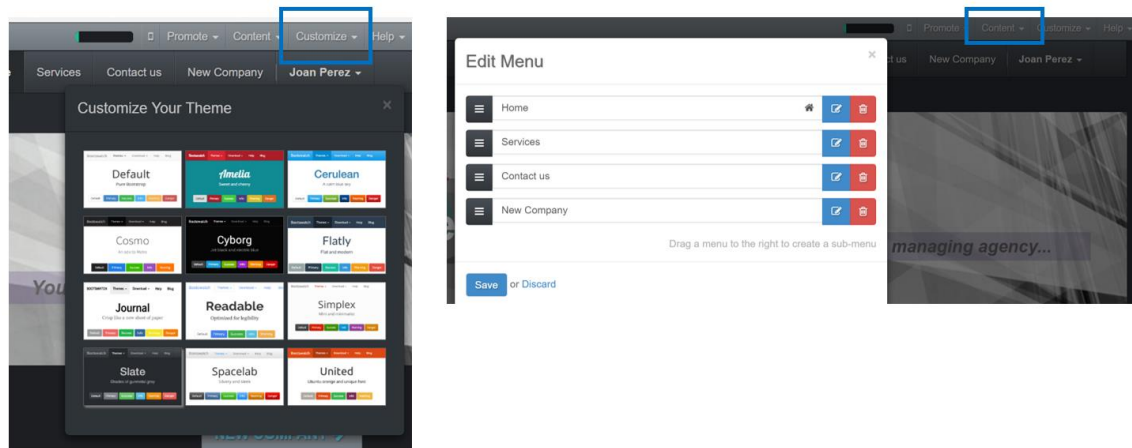
### 3.3.1 Website

To install the website builder module, after login as administrator, at the initial page of the Odoo it is listed all the applications available for the Odoo version 9. To install it is necessary to click at the button “Install” for the *Website Builder section*. After few seconds loading, the module it will show as “Installed”. The next step is choosing the type of web builder to use, in our case, we choose *Bootstrap*, because it is more intuitive. The Fig. 3.8 shows how the website is after the installation.



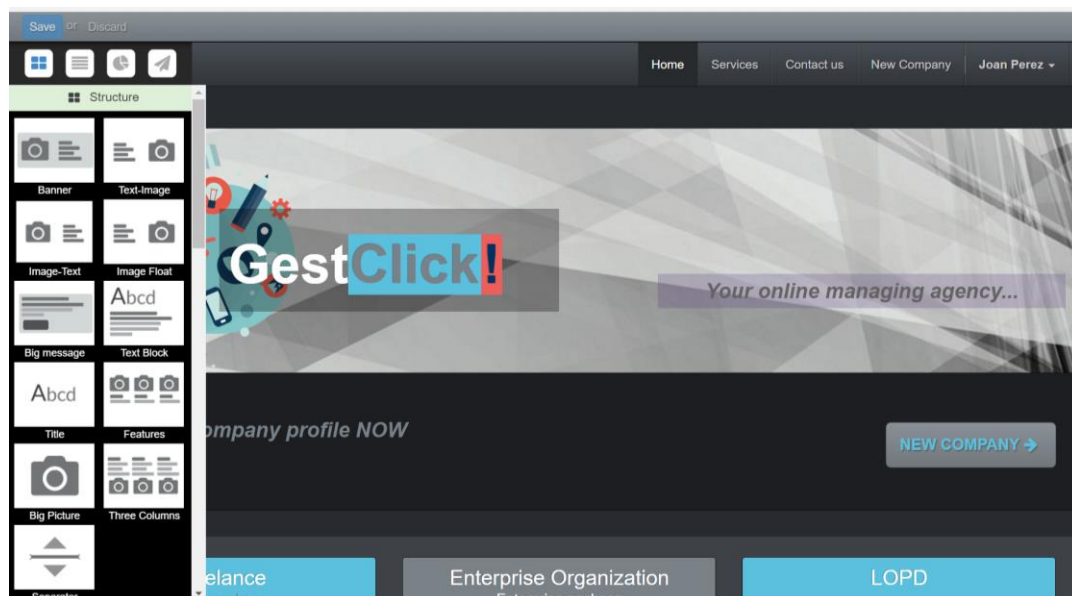
**Fig. 3.8** Detail of the website after the Bootstrap website builder installation

The theme chosen is *Slate* and can be selected as Administrator, in the website module. At the section “Customize”. Also, can be added tabs of content or change the tabs name at section “Content”. For example, it was changed the tab SHOP (installed by the e-commerce) by SERVICES, showed at Fig. 3.9.



**Fig. 3.9** Theme and menu content of the website

For the website design, it is used the Bootstrap design tools, because it very easy to set, and adapt to our needs. To show the logo and the slogan at the home page we used a structure block with a big image. Combined with a separator with a button that redirect to a custom template to create NEW COMPANY (intended to develop at futures updates). And for last we used a tool of features “comparisons” to order from the home page the most common products available at the e-commerce, with the direct link to order the product. The HTML code for the home page is available at the annex.

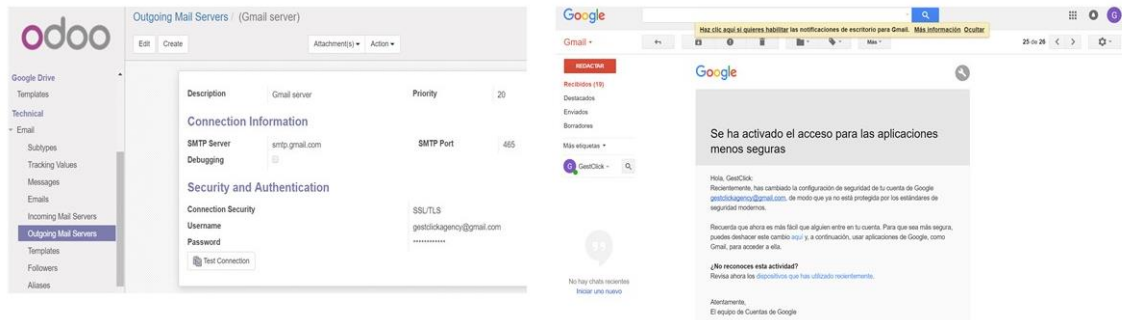


**Fig. 3.10** Bootstrap design tools

To configure the outgoing email server, it is necessary to login as administrator.

1. Activate the developer mode (About→ Activate developer mode)
2. Go to settings, then Email and Outgoing mail server

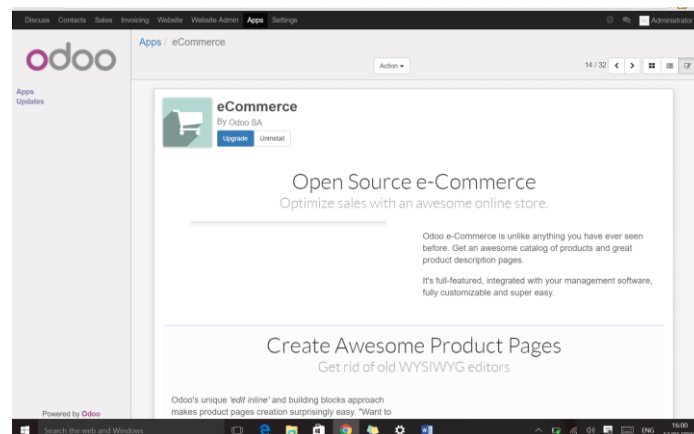
3. Delete the default configured email server
4. Create a new email server with the administrator email account ([gestclickagency@gmail.com](mailto:gestclickagency@gmail.com)), named Gmail server.
5. Define the priority as 20, SMTP server: smtp.gmail.com, SMTP port: 25, connection security: SSL/TLS, and set the email credentials, click on test the connection and after receive the email that indicates how to ALLOW APP at Gmail, the outgoing email server will be set up.



**Fig. 3.11** Outgoing mail server set up

### 3.3.2 Ecommerce

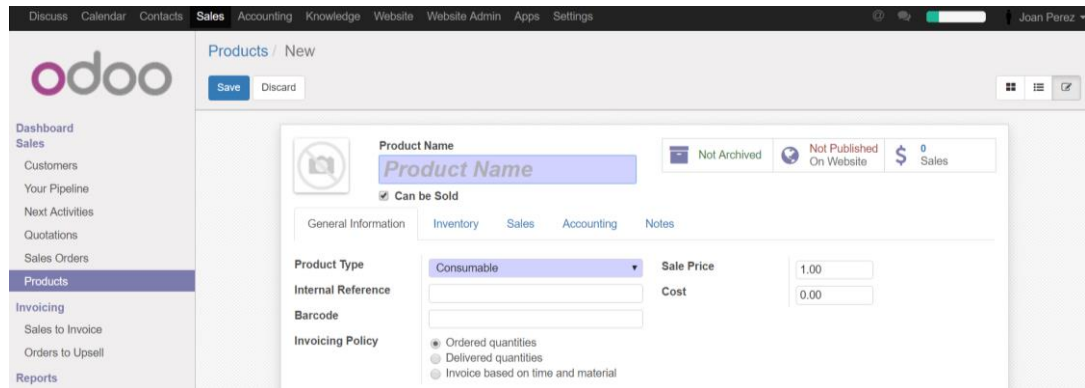
As the previous module, the eCommerce application is installed just clicking the “Install” button at the application section. By default, this module will install also the applications of sales management and invoicing, to manage all the sales process once the client contract an online product.



**Fig. 3.12** eCommerce module installation detail

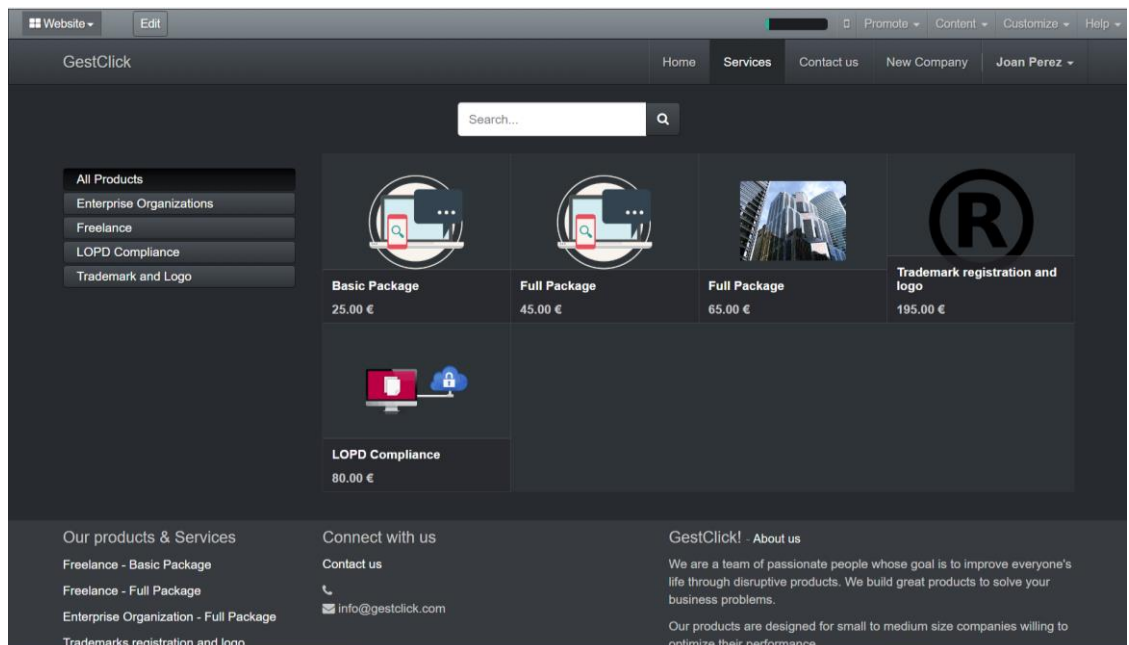
Once the e-commerce module is installed, it should be added all the services offered by the managing agency.

1. Login as Administrator
2. Go to the SALES section, and then the subsection PRODUCT.
3. Click on “Create”, to create a new product. The Fig. 3.13 shows the template of the product creation.



**Fig. 3.13** Creation of products

4. Specify the product name, and characteristics. It is important to define the website category (Freelance, Enterprise organizations...). It is necessary to create these categories. All the details and available products are defined at section 3.1.1.
5. Change the option “Not published on Website” to “Published” to have the product displayed at the e-commerce section. At Fig. 3.14 is shown the list of products available in the website.



**Fig. 3.14** Products available by category at GestClick



### 3.3.3 External client login

To activate the authentication and new external user registration, it is necessary to login as Administrator, and set the debug mode (Administrator → About → Activate developer mode) as it is shown in the Fig. 3.15.

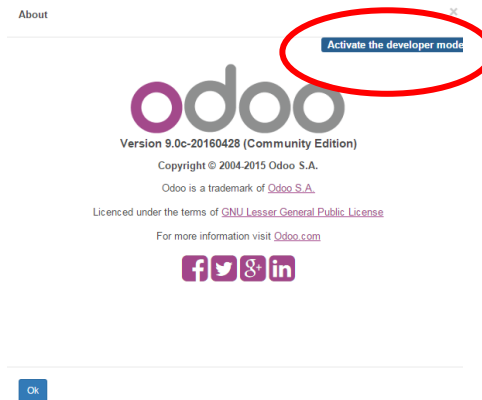


Fig. 3.15 Developer mode activation button

Then activate the option ***“Allow external user to sing up”*** at the “General settings” options as shown in the Fig. 3.16.

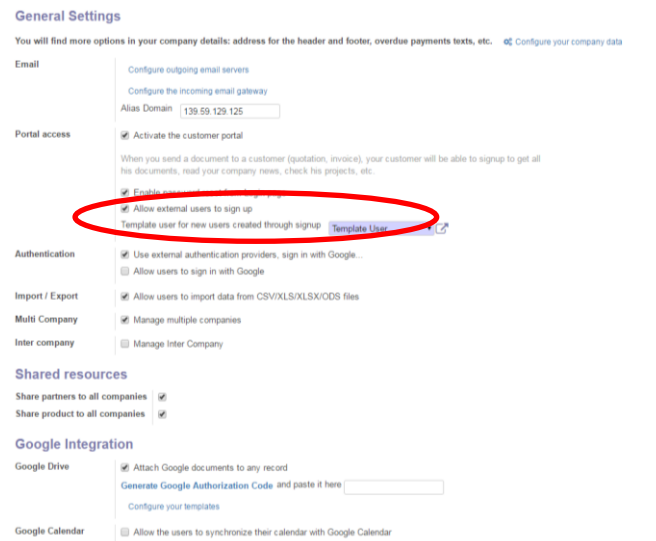


Fig. 3.16 Activation of the template form for external users

### 3.3.4 Knowledge management system

To install Knowledge management system module, it is needed to2 download the module from the Internet. The first step is open the website <https://www.odoo.com/apps> and make a search for this module, Knowledge Management System. Once we have found the module we want to install we have to click to get into the details. In the Fig. 3.17 we can see the details for this module. Then we have to download the module for the correct Odoo version, in this case, v9.0.

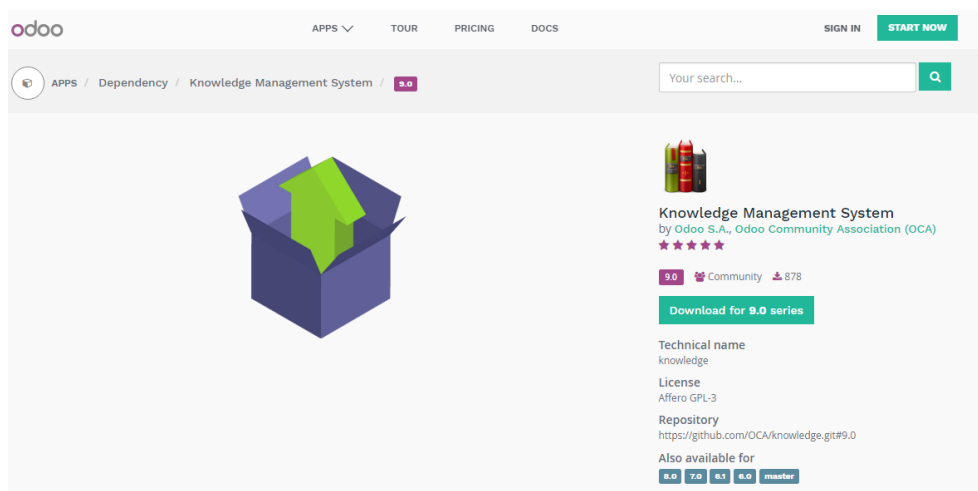


Fig. 3.17 Knowledge Management System module website

Once downloaded the modules, we have to unzip the file. Next step is sent it to our server. To do it we will use WinSCP, a common file transfer software. With this software, we can map a specific path in our server, in this case our path is /tmp/ in our server. In the Fig. 3.18 we can see the file already transferred.

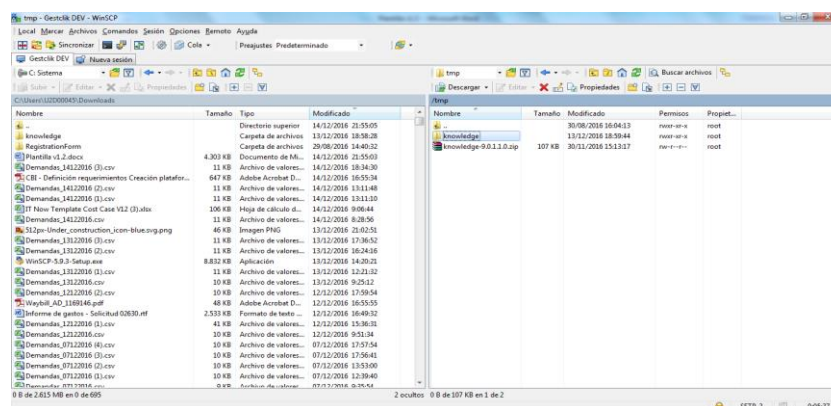


Fig. 3.18 File transfer to the server screenshot

To install this module, we just have to copy this folder on the addons path of Odoo. If we don't know this path we will have to check it going to the *openerp-server.conf* file. In our case, this file is in the path */etc/odoo*. The only problem is that we firstly have to enter to the odoo docker with the next command:

```
docker exec -it odoo bash
```

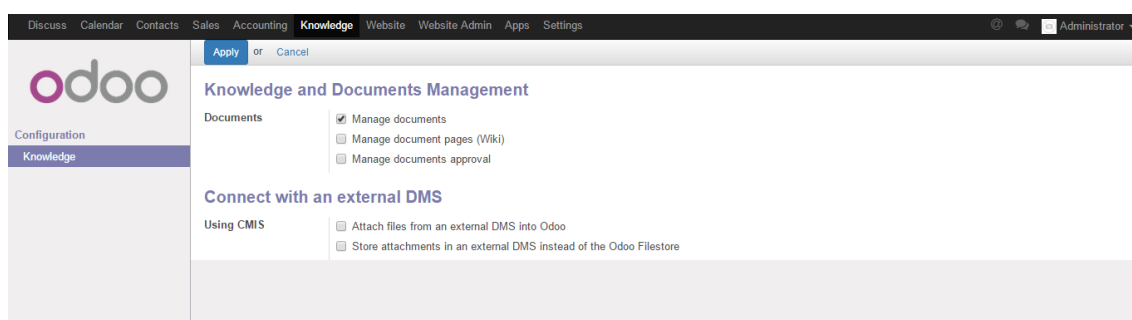
Once inside we go to the path and we open the file *openerp-server.conf* where we can find the addons path:

```
addons_path = /mnt/extra-addons,/usr/lib/python2.7/dist-packages/openerp/addons
```

Next step is to copy the folder *knowledge* from */tmp/* on the virtual machine to */usr/lib/python2.7/dist-packages/opener/addons* on the odoo docker. To copy a file or folder to a docker we have to use specific commands. In this case the command we have to use from the */tmp* in the virtual machine is:

```
docker cp ./knowledge odoo:/usr/lib/python2.7/dist-packages/openerp/addons/knowledge
```

Once we have copied the folder we can sign in to Odoo via browser using the administrator user and we will see the new module installed as shown in the Fig. 3.19.



**Fig. 3.19** Knowledge Management System module settings menu

## CHAPTER 4. MOBILE APPLICATION

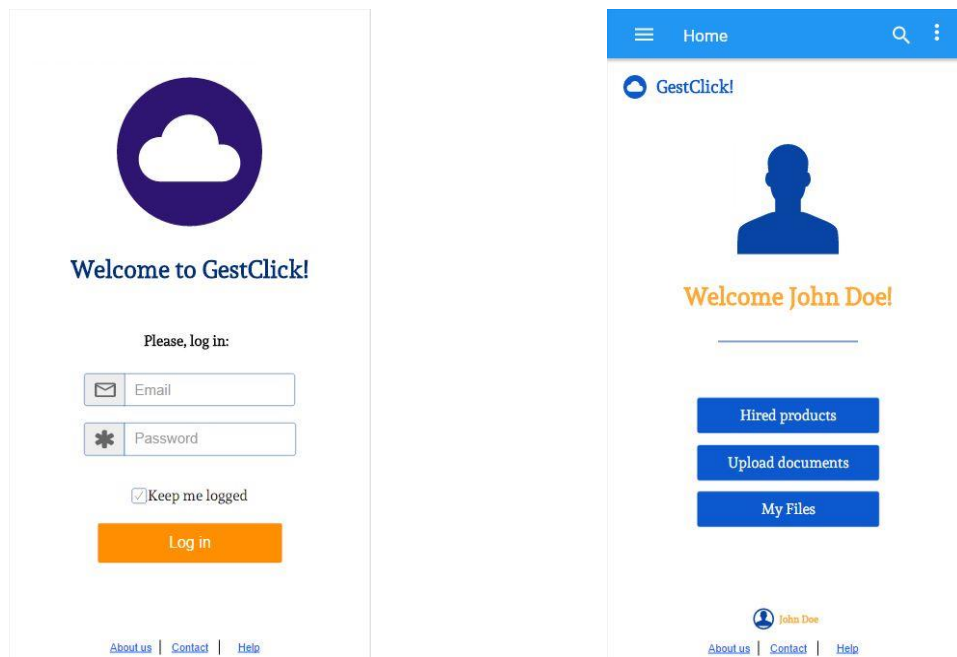
### 4.1 Webapp Design

As demanded for the client, it is a requirement to design a mobile application compatible with smartphones with the main OS in the market, Android and iOS.

In the first place, we have to make the mockup of the web app. The client requires to include at least the following functions in the app:

- Login
- List hired products
- Upload documents from a file
- Upload documents from the camera
- Check the saved files

Once determined the requirements of the webapp it is time to design the different screens of the website. To simplify the navigation for the users our proposal is shown in the following captures.

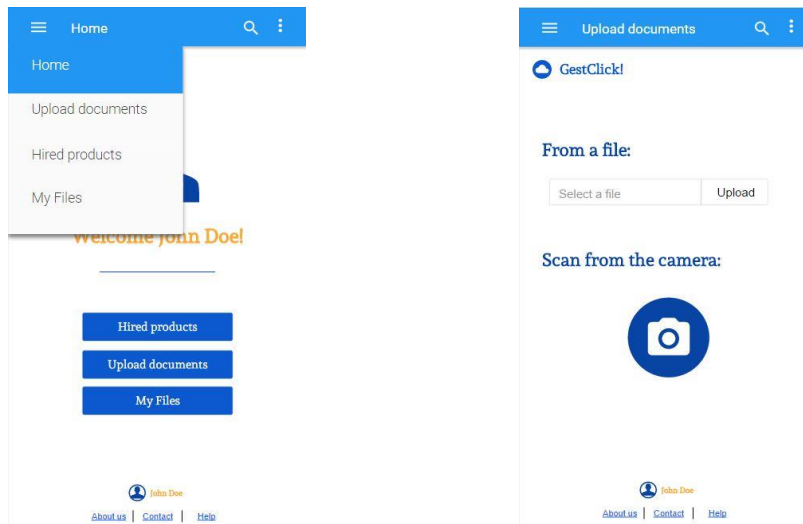


**Fig. 4.2** Login page design proposed (left)

**Fig. 4.1** Home page

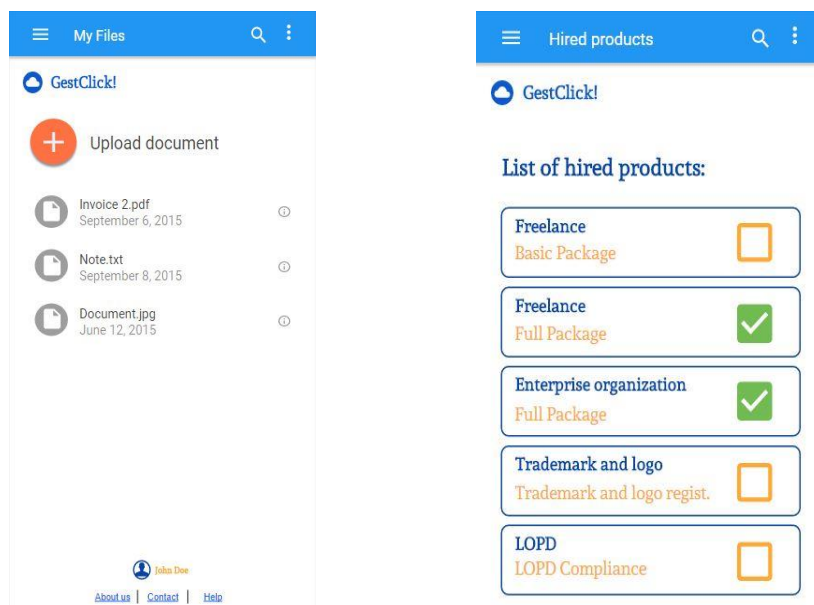
In the Fig. 4.2 we can see the login page with two text box to fulfill with the e-mail address and the password. In the Fig. 4.1 we can see the welcome page with three main functions available from a button.

In the Fig. 4.3 we can see the options available from the menu that is always available in the top frame. In the Fig. 4.4 we can see the screen to upload documents both from a file and from the camera.



**Fig. 4.4** Upload documents screen design proposed  
**Fig. 4.3** (left) Welcome menu design proposed

In the Fig. 4.6 the client can check the saved files, and also click to upload new ones. In the Fig. 4.5 we can see a list of the products available and the ones



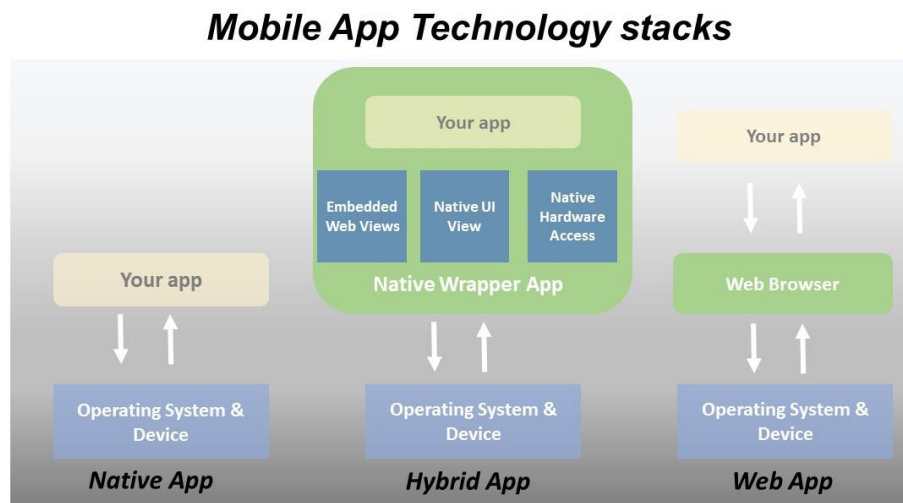
**Fig. 4.5** Files screen design proposed  
**Fig. 4.6** (right) Hired Products screen design proposed

that are already hired for the client.

## 4.2 Technology stack

In order to be able to implement the design showed on the previous section (4.1), we will need to do research to know which technologies work better to accomplish our goal, to create an Android and iOS smartphone application.

The first step when creating a smartphone application (from that point an App) is decide if we want a Native App, a Webb App or an Hybrid App. Every type of App has some advantages and throwbacks. To understand the difference between the types of Apps we can see the Fig. 4.7 where we can observe the different types of technology stacks for each type. As we can observe Native Apps are built specifically for a particular mobile platform and it uses platform's native programming language and tools. Native Apps invokes directly the operating system and hardware. On the other hand, Web Apps are just a mobile-optimized website or page that runs in a browser on the smartphone device. Languages used for this type of Apps are HTML5, CSS, JavaScript, Angular, etc. The las type of Apps are the Hybrid Apps, which are similar to Native Apps but the core of this Apps are built using standard web technologies as the ones mentioned before. That core is stored and runs from within a "wrapper" native app. The definitive App is built using the same tools as native App.



**Fig. 4.7** Mobile App technology stacks contemplated

In the Table 4.1 we can compare the strengths and weaknesses of each type of App:

Type	Benefits	Caveats
Native Apps	Performance Connection to phone tools (camera, sensors...) Standard UI components Quality of the development support from vendors	Require knowledge of the languages of the platform Code is written for each platform Long development time High costs
Hybrid Apps	Reusable code basis Cost-effective Related technologies are universal and not controlled by a single vendor	Potential performance penalization Associated to platform idiosyncrasies Limited access to device frameworks Difficult debugging process
Web Apps	Few resources needed Development time reduced Related technologies are universal and not controlled by a single vendor Control of the distribution	Limited access to device hardware Require an Internet connection Difficult to earn confidence in the app from the users (security, origin,...) Difficult debugging process

**Table 4.1** Strengths – weaknesses comparison for different mobile app technology stacks

Once we have analyzed the different type of mobile Apps, it is time to decide which kind of App fits better in our needs to develop a mobile app. considering the size of our client, a Native App for each platform required would suppose a huge cost not affordable for our client. As a throwback, not using native apps will make more difficult the access to the device hardware, as for instance, the camera. Considering the other extreme, a Web App would fit in our client's budget but it has some caveats. The main one, and the one that would make reconsider using Web Apps, is the difficulty of earning confidence from the users of the App. The goal of the app is not only hire new products but also to upload documents and file. These documents are confidential and they might contain very secret information and users would not trust a Web App to upload this kind of documents. Because of these constraints, the best option to develop our Mobile App is to build a Hybrid App that will let us to reuse code for the different platforms, will have an affordable cost for our client and the technologies used will be universal. As a throwback, we might have some performance penalization, will have to work on each platform idiosyncrasies and will be more difficult to debug in case of errors in our code.

Once we have chosen the type of App it is time to define more concretely the technology stack. One of the benefits of developing a Hybrid App is the use of

universal languages to code the App. In this case, there are several languages but our selection based on our previous experiences is the following:

- **HTML:** is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. It is relatively easy to learn, with the basics being accessible to most people in one sitting; and quite powerful in what it allows you to create. It is constantly undergoing revision and evolution to meet the demands and requirements of the growing Internet audience under the direction of the W3C, the organization charged with designing and maintaining the language.
- **SASS:** is a scripting language that extends CSS by allowing developers to write code in one language and then compile it into CSS. Some examples of CSS preprocessor include: Sass, LESS and Stylus.
- **Angular 2:** is a complete JavaScript-based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications.
- **Ionic 2:** Ionic is a complete open-source SDK for hybrid mobile app development. Built on top of AngularJS and Apache Cordova, Ionic provides tools and services for developing hybrid mobile apps using Web technologies like CSS, HTML5, and Sass. Apps can be built with these Web technologies and then distributed through native app stores to be installed on devices by leveraging Cordova.
- **Apache Cordova:** is a mobile application development framework that enables software programmers to build applications for mobile devices using CSS3, HTML5, and JavaScript instead of relying on platform-specific APIs like those in Android, iOS, or Windows Phone. It enables wrapping up of CSS, HTML, and JavaScript code depending upon the platform of the device. It extends the features of HTML and JavaScript to work with the device. The resulting applications are hybrid, meaning that they are neither truly native mobile application nor purely Web-based.

In the Fig. 4.8 we can see the relation between each technology that defines our Hybrid App technology stack.





Fig. 4.8 Technology stack proposed to develop the App

### 4.3 System requirements and software installation to develop Ionic 2 mobile application

The mobile application will be made with ionic 2 and programmed with Angular 2. To manage the code, it is used the IDE Visual Studio Code. Visual Studio Code is potent source code editor which is characterized by its weight lightness. The software is available for every operating system (Windows, Mac and Linux). It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Python, PHP, Go) and runtimes. It is available to download at their website.

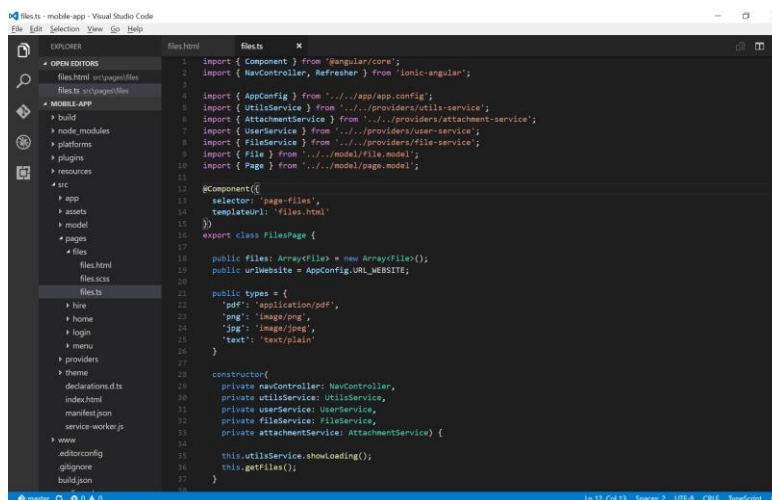


Fig. 4.9 Visual Studio Code

The first steps to develop with Angular 2 and Ionic 2 is to install Node.js and npm. These components are essential to modern web development. Node powers client development and build tools. The npm package manager, itself a node application, installs JavaScript libraries. The npm facilitates to the developers share, reuse and update JavaScript code that you are already sharing. The Node.js installer comes with the npm installed too: <https://nodejs.org/en/download/>. The latest version: v6.9.4 (includes npm 3.10.10).

To check the version running at the pc, can be use the windows command prompt and write the commands:

```
Npm -v (to check the npm version)
Node -v (to check the node.js version)
```

All the information about Node.js and npm package manager can be find at: <https://docs.npmjs.com/getting-started/what-is-npm> and <https://nodejs.org/en/about/>.

With all this installed we already can develop application with Angular 2. In order to use Ionic 2, it is necessary to install all the Cordova and Ionic command-line tools in our terminal; Open the windows console, and run it as Administrator and enter this command:

```
npm install -g cordova ionic
```

```
Administrator: Command Prompt
C:\>npm install -g cordova ionic
npm WARN deprecated node-uuid@1.4.7: use uuid module instead
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated minimatch@0.3.0: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated minimatch@2.0.18: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated node-uuid@1.3.3: use uuid module instead
C:\Users\Karina\AppData\Roaming\npm\cordova -> C:\Users\Karina\AppData\Roaming\npm\node_modules\cordova\bin\cordova
C:\Users\Karina\AppData\Roaming\npm
+--- cordova@6.4.0
+-- ionic@2.1.18

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.0.0 (node_modules\ionic\node_modules\chokidar\node_modules\fs
sevents):
npm WARN deprecated SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.0.17: wanted {"os":"darwin","arch":"ar
v") (current: {"os":"win32","arch":"x64"})
npm WARN In ionic@2.1.18 replacing bundled version of cross-spawn with cross-spawn@4.0.2
npm WARN In ionic@2.1.18 replacing bundled version of mime-types with mime-types@2.0.14
npm WARN In ionic@2.1.18 replacing bundled version of semver with semver@4.2.0
npm WARN In ionic@2.1.18 replacing bundled version of form-data with form-data@0.2.0
npm WARN In ionic@2.1.18 replacing bundled version of request with request@2.51.0
npm WARN In ionic@2.1.18 replacing bundled version of ionic-app-lib with ionic-app-lib@2.1.9
C:\>
```

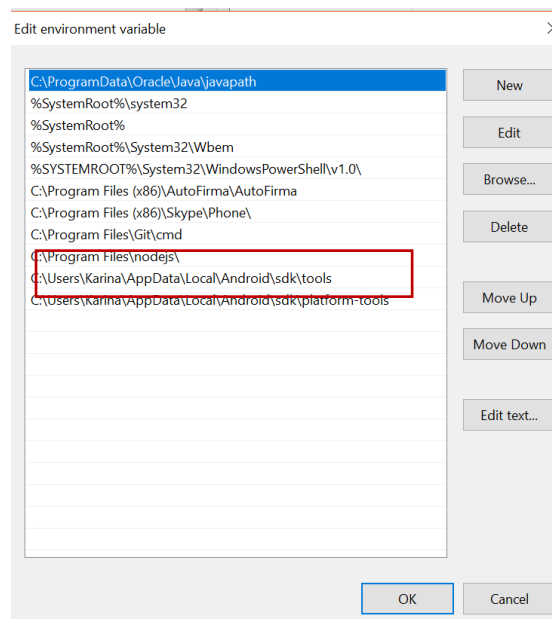
**Fig. 4.10** Installation of Cordova and Ionic command-line tools

Then it must be installed the Android and IOS (only with Mac) platform, and all the required tools for development.

### 4.3.1 Android SDK set up

The Android SDK environment has to be set up to deploy Ionic 2 (Cordova + Angular2) applications.

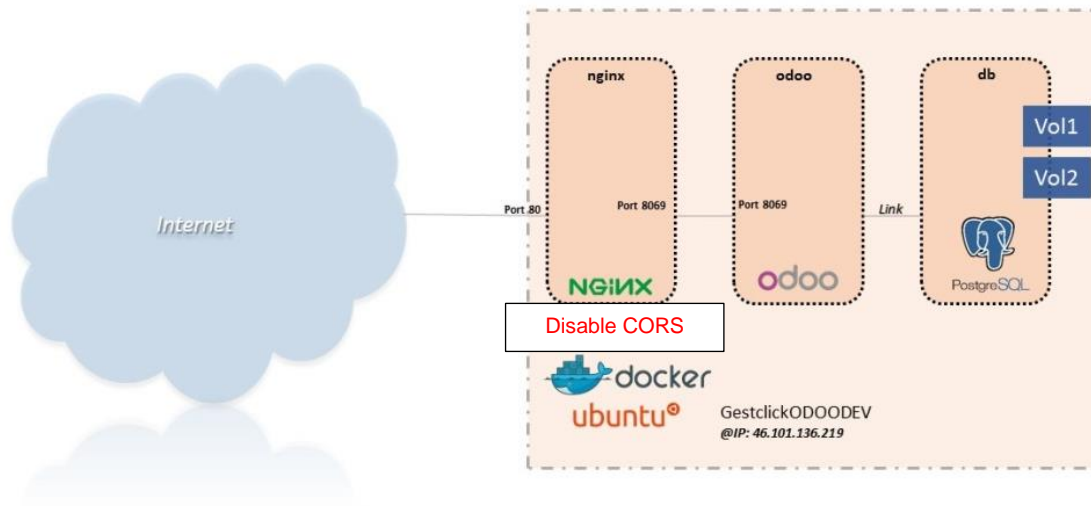
1. Install the latest version of Java Development Kit (JDK), version: 8.1: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html> , following all the recommended installation steps detailed at the downloading link.
2. Create a new Environment Variable on your system named: JAVA\_HOME, which points where it is installed the Java SDK: *C:\Program Files\Java\jdk1.8.0\_111*.
3. Install the Android Studio: <https://developer.android.com/studio/install.html?pkg=studio> .
4. Create another Environment Variable on your system named: ANDROID\_HOME, which should point where it is installed the Android SDK (in our case): *C:\Users\USER\AppData\Local\Android\sdk*.
5. Edit the path of the Environment Variable of the system, to indicate the routes for the SDK tools and SDK platform-tools:



**Fig. 4.11** Editing path for environment variable

6. Open the Android Studio, and run the Android SDK Manager, in order to install all the missing components:





**Fig. 4.13** CORS solution diagram

The code for the custom NginX proxy, it is at the `default.conf` and executed during the creation of the containers with the docker-compose. The file is detailed and explained at section 3.2.2.

## 4.5 Coding the app

Once we have set the technology stack and the environment is created, it is time to start coding the app. As a first step we have to create the folders structure required by Ionic 2. In the Annex we can see the detailed folder-file structure but to introduce the different folder we are mentioning the most important ones as follows:

- **src:** In this folder are contained all the files related with the webapp, divided in the following sub-folders:
  - **app:** This folder contains the initialization commands for the app, the common variables declaration, modules declaration and the scenarios (in this case development and production).
  - **model:** This folder contains the declarations of the classes. In our case we will just define the class *User*.
  - **pages:** The different pages of our app are defined here. We will create a subfolder for each page. Inside of the subfolders we are creating an HTML code, an SCSS style file and a TypeScript component.

- **providers:** This folder contains the services required by the pages/components in order to get or put information from the classes.
- **theme:** This folder contains the declarations of variables related with the style of the pages.
- In addition, this folder contains a few files needed to make the app works. This files are the following:
  - declarations.d.ts: in this file is declared the module *xmlrpc*.
  - index.html: this file contains the HTML code called to open the app.
- **platforms:** this file contains definitions for the different platforms selected and the native files to embed the web application to a native container.
  - platforms.json: in this file are set the versions of the different operative systems.
- **.tmp:** this file is created automatically for the runtime environments and is used by them to store temporary files.
- In addition, in the root folder there are a few files very important for the definition of the app:
  - config.xml: in this file are set:
    - preferences for the app as minimum version, if screen rotation is allowed or not, etcetera.
    - Cordova plugins as camera plugin, file browser plugin, etcetera.
    - platforms supported by the application.
  - ionic.config.json: in this file are set some parameters for the app as name, app\_id that must be unique in the world if we want to distribute the app through the stores as Apple Store or Google Play.
  - package.json: in this file are set the web dependences for Angular and Ionic managed by npm (the package manager for node.js).

## 4.6 Connection to Odoo

In order to be able to show data from our Odoo instance or to write data from our app to the Odoo instance, we will need to connect and interact with Odoo and its data.

In this case, Odoo includes an API to make easier to extend Odoo features and data to external applications or tools. All the documentation related with the Odoo Web Service API is available in the next URL:

[http://www.odoo.com/documentation/9.0/api\\_integration.html](http://www.odoo.com/documentation/9.0/api_integration.html)

In further subsections we will find how to use this API and how to implement it in our Mobile App.

#### 4.6.1 XML-RPC

As we can appreciate in the documentation, the use of XML-RPC is mandatory to call the different services defined in the API.

XML-RPC is a specification and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet. The `xmlrpc` module is a pure JavaScript XML-RPC server and client for node.js.

It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned<sup>2</sup>.

The version available online for the XML-RPC (<https://github.com/baalexander/node-xmlrpc>) as long as we use it, it repeatedly caused an error at the same lines once we compile and debug the first version of the app (`lib/deserializer.js`), to solve this problem we download the whole repository and just comment the lines that continuously gave the error, and it works perfectly like this. Then the XML-RPC repository that we used is a customized one that it is commented at the lines mentioned: <https://bitbucket.org/gestclick/node-xmlrpc>. The dependency it is referred at the `package.json` file of the app. Below the lines commented at `lib/deserializer.js` (line 46 and line 69):

```
//stream.setEncoding(this.encoding)
stream.on('error', this.onError.bind(this))
stream.pipe(this.parser)
}

Deserializer.prototype.deserializeMethodCall = function(stream,
callback) {
  var that = this
```

<sup>2</sup> [XML-RPC] (1999). UserLand Software, Inc. <http://xml-rpc.com>

```

this.callback = function(error, result) {
  if (error) {
    callback(error)
  }
  else if (that.type !== 'methodcall') {
    callback(new Error('Not a method call'))
  }
  else if (!that.methodname) {
    callback(new Error('Method call did not contain a method name'))
  }
  else {
    callback(null, that.methodname, result)
  }
}

//stream.setEncoding(this.encoding)
stream.on('error', this.onError.bind(this))
stream.pipe(this.parser)

```

#### 4.6.2 Services published in the API

As any API it has different functions published and we have selected the next ones to use in our mobile application in order to cover the requirements:

- **authenticate:** it is the only call we are using that does not require authentication. This is because it is the service used to authenticate to the API and allow the users to query most data. The *authenticate* function returns a user identifier (*uid*) used in authenticated calls instead of the login.
- **search():** this function is used to filter and list records from the database. The outcome of this function is a list of identifiers of the records matching the filter.
- **read():** this method is used to access the record data. The *read* method takes a list of ids (usually returned by *search*) and optionally a list of fields to fetch.
- **search and read():** it is a merge of the two previous functions to avoid having to perform two requests and keep the ids.
- **create():** new records of a model are created using this method. The method creates a single record and return its database identifier.



### 4.6.3 Configuration of the XML-RPC

Now we have the protocol and we know the methods we can invoke, it is time to connect externally to our Odoo instance. To do that we have to declare the module in the *declarations.d.ts* file:

```
declare module 'xmlrpc';
```

After that in the *odoo-service.js* we have to initialize the XML-RPC client:

```
export var OdooService = (function () {  
  function OdooService() {  
    // Initialize the xmlrpc client  
    this.client = xmlrpc.createClient({  
      host: AppConfig.HOST,  
      port: AppConfig.PORT,  
      path: AppConfig.URL  
    });  
  }  
})
```

Before that, we have previously defined the connection variables in the *app.config.ts*:

```
export class AppConfig {  
  // Odoo connection  
  public static get HOST(): string { return '46.101.136.219'; }  
  public static get PORT(): number { return 80; }  
  public static get DB(): string { return 'gestclickdb'; }  
  public static get URL(): string { return '/xmlrpc/2/common'; }  
}
```

Once we have done the three previous steps we must have our mobile application connected to the Odoo instance.

### 4.6.4 Models definition

Inside the `src` folder we have created a folder named `model`. It contains the data structure then we will be able to share data between components (the app) and providers (third parties data).

In the annex we can see the code of any of the next files but following these lines there is a brief explanation about each of these models:

- **credentials.model.ts:** declaration of models for the credentials, in this case, e-mail and password.
- **file.model.ts:** this model gets the object from the Odoo and converts it to a *File* class.
- **ir.attachment.model.ts:** this model allows saving in the database the objects that contains a *File*.
- **order.line.model.ts:** this model allows converting an object to an *OrderLine* and vice versa.
- **order.model.ts:** same function than the previous model but allows creating an array of *OrderLine*.
- **page.model.ts:** this model is only for internal usage for managing pages along the applications and it is not used to share data with providers.
- **parent.model.ts:** Odoo structure creates always a partner and a parent so we have to create a model for both.
- **partner.model.ts:** same than the previous one.
- **product.model.ts:** this model allows converting an object to a *Product* and vice versa.
- **user.model.ts:** this model allows converting an object to a *User* and vice versa.

#### 4.6.5 Creating providers

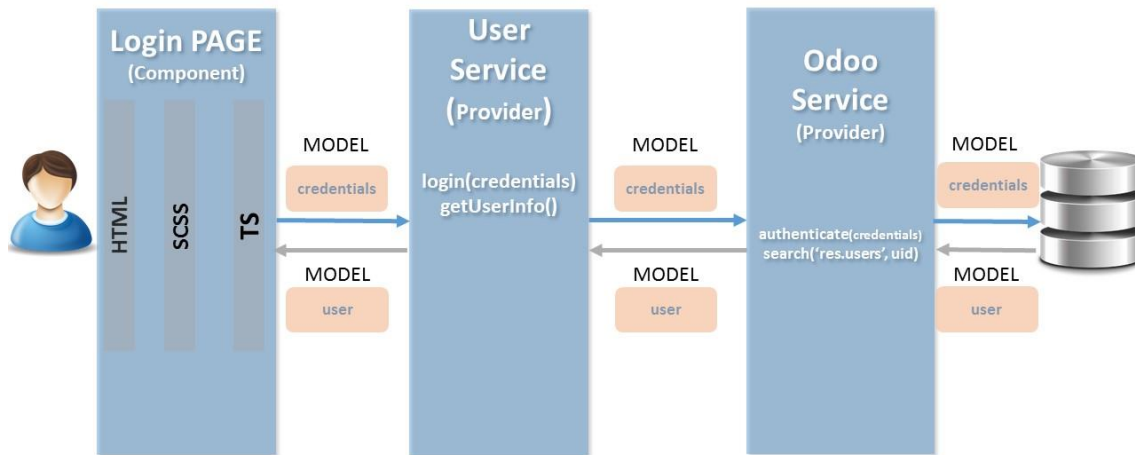
After defining the models is time to create the providers that will allow us to invoke to the services when we will be creating the components. In the Annex is detailed all the code for this providers but summarizing the most important points of it are:

- **attachment-service.ts.** This provider contains the following methods:
  - `getAttachments()`: returns an array of files configured in the odoo platform for a particular user.
  - `showActionSheet()`: opens an Action Sheet controller with some buttons in order to select a file from the camera or the gallery.
  - `openCamera()`: opens the camera or the gallery depending what the user selects on the action sheet. A boolean parameter defines if the image should come from the

gallery (true) or if it is from the camera (false). After that, it creates an attachment in the database for this user.

- **file-service.ts.** This provider contains the following methods:
  - `saveFile()`: saves a file in the storage of the device
  - `openFile()`: opens a file on the native platform for viewing the files. The viewer is selected from the mime type.
  - `64toBlob()`: converts a base64 string into a Blob.
- **odoo-service.ts:** this provider creates the connection to the Odoo instance using the XML-RPC protocol and invokes all the web services published by the Odoo API as seen in the previous section 4.6.2. Contains the following methods:
  - `authenticate()`: executes a login against the Odoo platform. The method returns an observable created from a Promise created from a callback of the `xmlrpcs` library. This method returns true if the login is successful or false otherwise.
  - `search()`: search and read all the objects of the entity defined in the parameters in the Odoo database. Returns an array of objects.
  - `read()`: search and read an object from the database. We have to pass the entity you want to read and the id of this entity. Returns an object.
  - `searchAndRead()`: This method search and read an object from the database. We have to pass the entity we want to read and the id of its entity. Returns an array of objects.
  - `create()`: creates an object into the database. We have to pass the name of entity we want to create and the object to store.
- **product-service.ts.** This provider contains the following methods related with the products:
  - `getProducts()`: returns an array of products configured in the Odoo platform (*product.product* entity).
  - `getOrdersByUser()`: returns the orders of a partner into an array.
  - `getLinesByOrder()`: returns the order lines of an order into an array of order lines.
- **user-service.ts.** This provider contains the following methods:
  - `login()`: performs a login against the odoo platform.
  - `getUserInfo()`: returns all the user information for the logged user. It takes the uid from the Odoo service after a successful login.
  - `logout()`: performs a logout on the platform.
- **Utils-service.ts:** This provider contains a few methods related with utilities to build the app as `showAlert()` or `showLoading()`.

The following diagram shown an example of how the system interact with “models” through different providers to interact with the Odoo database:



**Fig. 4.14** Diagram providers and models Login page

#### 4.6.6 Pages coding

Finally, when we have the connection with Odoo, the models defined the providers created is time to build (or code) the pages we will see in our app. In order to achieve that goal, we have to create three different files for each page:

- .html: this file contains the Ionic HTML code with the content of the page.
- .scss: this file contains all styles of the page.
- .ts: this file contains the Angular Component that performs as a view controller for relating the user interactions with the backend services.

The pages we have to code as part of the requirements set in the section 4.1 are the following:

- Login
- Home
- Menu
- Hired products
- Files

The important part of these pages is to configure properly the component of the TypeScript file. For this reason we are getting into details of the components of each page to understand better what they are doing.

#### 4.6.6.1 *login.ts*

In this component we are calling the method *login()* after inserting the email and password and pressing the submit button. This method returns a boolean as mentioned in the previous section and allows the login if this is true or show error messages otherwise.

```
public login(): void {
    this.utilsService.showLoading();
    this.userService.login(this.credentials).finally(() => {
        this.utilsService.removeLoading();
    }).subscribe(
        (allowed: boolean) => {
            if (allowed) {
                this.navController.setRoot(MenuPage);
            } else {
                this.utilsService.showAlert('ERROR', 'Access Denied');
            }
        },
        error => this.utilsService.showAlert('ERROR', error))
}
```

#### 4.6.6.2 *home.ts*

The first method called in this component is *getUserInfo()* to show in the home page the name, picture,... of the user as we can see in the following code:

```
private getUserInfo(refresher?: Refresher): void {

    this.userService.getUserInfo().finally(() => {
        refresher ? refresher.complete() : null;
        this.utilsService.removeLoading();
    }).subscribe(
        ((value: User) => this.user = value),
        error => this.utilsService.showAlert('ERROR', error));
}
```

The second method called is a basic navigation method that gives the page to go by parameter:

```
public openPage(page: Page): void {
    this.navController.setRoot(page.component);
}
```

The last method called is used for opening an action sheet *showActionSheet()* for selecting the source origin for uploading documents to the server.

```
public showActionSheet(): void {
    this.attachmentService.showActionSheet();
}
```

#### 4.6.6.3 menu.ts

The menu page only calls to the *openPage()* method and to the *logout()* method because it is just used to flip to other pages or to logout the application. The second method call is used as follows:

```
public logout(): void {
    this.userService.logout().subscribe(
        result => this.nav.setRoot(LoginPage));
}
```

#### 4.6.6.4 hire.ts

The hired products page only calls one method, the *getProducts()*. This method get a list of all the products, create a new list and move the products already hired for the user from the first list to the second one. This way the result is a list of hired products and another one of available products that are shown finally in the page. The code for this call is the following:

```
private getProducts(refresher?: Refresher): void {

    this.productService.getProducts().subscribe(
        ((products: Array<Product>) => {
            this.availableProducts = products.sort((p1, p2) => {
                return p1.id > p2.id ? 1 : -1;
            });
        }));
}
```

```
// Get the orders of the current user
this.productService.getOrdersByUser(this.userService.user.partner.id).
subscribe(
  ((orders: Array<Order>) => {

    this.hiredProducts = new Array<Product>();

    for (let order of orders) {

      // Get the orders lines of the current orders
      this.productService.getLinesByOrder(order.id).finally(()
=> {

        refresher ? refresher.complete() : null;
        this.utilsService.removeLoading();
      }).subscribe(
        ((ordersLine: Array<OrderLine>) => {
          // Mark the products ordered by the user
          for (let line of ordersLine) {
            this.availableProducts.forEach((product, index) =>
{
              if (line.productId[0] === product.id) {
                this.hiredProducts.push(product);
                this.availableProducts.splice(index, 1);
              }
            });
          }
        })),
        error => this.utilsService.showAlert('ERROR', error));
      }
    },
    error => this.utilsService.showAlert('ERROR', error));
  }
),
error => this.utilsService.showAlert('ERROR', error));
}
}
```

#### 4.6.7 Compilation, execution and distribution

To install all the npm dependencies of the project (Angular, Ionic, other libraries), and Cordova dependencies (plugins and platforms), is necessary to run this command from the cmd prompt:

```
npm install
mkdir www
cordova prepare
```

The `npm install` command basically create the `node_modules` directory in your current directory – mobile-app - (if one doesn't exist yet), and download the packages into that directory.

The `cordova prepare` command downloads the platforms and plugins files for embedding the web application into a native container.

To build and develop can be used all of these ionic commands, depending on which platform will be used to run the application:

```
ionic serve
ionic emulate {android|ios}
ionic run {android|ios}
```

With the `ionic serve` will start a live-reload server for the project. When changes are made to any HTML, CSS, or TypeScript files, the browser will automatically reload when the files are saved.

With `ionic emulate / run {android|ios}` commands will deploy the app to the specified platform devices/emulators (Android or iOS). It is possible to live reload with these commands adding the `-livereload` option, which similar to the `ionic serve`, that do not need to develop and debug every time that the files changes, because the hybrid compiled app itself is watching if it change.

#### 4.6.8 Publish the application and add new version with HockeyApp

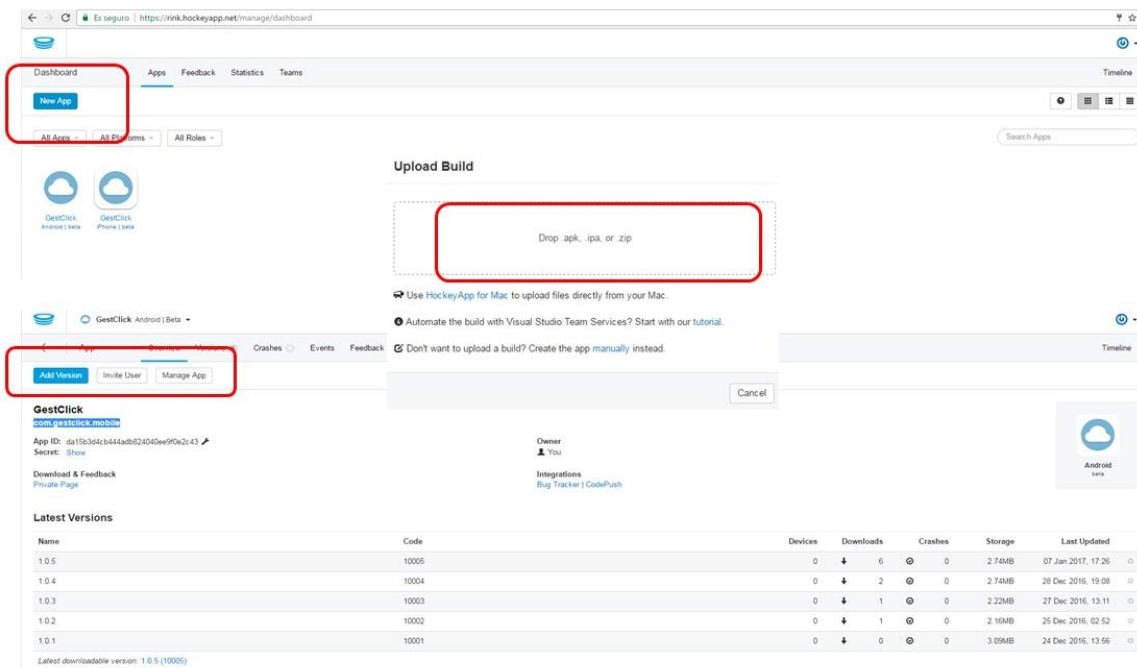
Once the application it is properly working as is desired, it is time to publish the application to download it from any device.

HockeyApp is a website that offers a service for mobile application developers. It helps them to publish mobile application in different platforms, manage and add testers, edit distributions and versions for this app. This service supports platform for app on iOS, Android, Mac OS X and Windows phone.

The Fig. 4.15 shows how to create new apps with HockeyApp:

1. Click on create New App
2. Import the APK file for Android or IPA file for iOS, these files are generated once the mobile application is build, see section 4.6.7.
  - a. APK file location:
  - b. IPA file location:





**Fig. 4.15** Publishing mobile app for tester with HockeyApp

To create the APK and IPA file after the building are used two scripts, located at (Android): `MOBILE_APP/build/build_android.sh`, (iOS): `MOBILE_APP/build/build_ios.sh`. The code for the Android script (iOS script is the same) is detailed below:

```
#!/bin/sh

# Import build utils script
. ./build_utils.sh

# Artifact information
APP_NAME=gestclick
APP_VERSION=1.0.5
APK_NAME=${APP_NAME}-${APP_VERSION}.apk
OUTPUT_APK=${PWD}/${APK_NAME}

# Build information
BUILD_DIR=${PWD}/../platforms/android/build/outputs/apk
BUILD_APK=${BUILD_DIR}/android-release-unsigned.apk
BUILD_TOOLS=${ANDROID_HOME}/build-tools/25.0.0

# Check env variables
check_env "${ANDROID_HOME}" "ANDROID_HOME environment variable must
define the location of the Android SDK"
check_env "${GC_KEYSTORE}" "GC_KEYSTORE environment variable must
point to a valid keystore"
check_env "${GC_KEYSTOREPWD}" "GC_KEYSTOREPWD environment variable
must contain the keystore password"
check_env "${GC_KEYSTOREALIAS}" "GC_KEYSTOREALIAS environment variable
must contain the keystore alias"

# Dependencies
```

```

execute_command "npm install --loglevel warn" "Downloading npm
dependencies..."
execute_command "cordova prepare" "Installing cordova dependencies..."

# Build the application
execute_command "ionic info" ""
execute_command "ionic build android --prod --release" "Building the
app..."
execute_command "echo ${GC_KEYSTOREPWD} | jarsigner -verbose -sigalg
SHA1withRSA -digestalg SHA1 -keystore ${GC_KEYSTORE} ${BUILD_APK}
${GC_KEYSTOREALIAS}" "Signing the app..."
execute_command "${BUILD_TOOLS}/zipalign -f -v 4 ${BUILD_APK}
${BUILD_DIR}/${APK_NAME}" "Aligning the app..."
execute_command "mv ${BUILD_DIR}/${APK_NAME} ${OUTPUT_APK}" "Moving
final apk..."
echo "[${APP_NAME}] apk: ${OUTPUT_APK}"

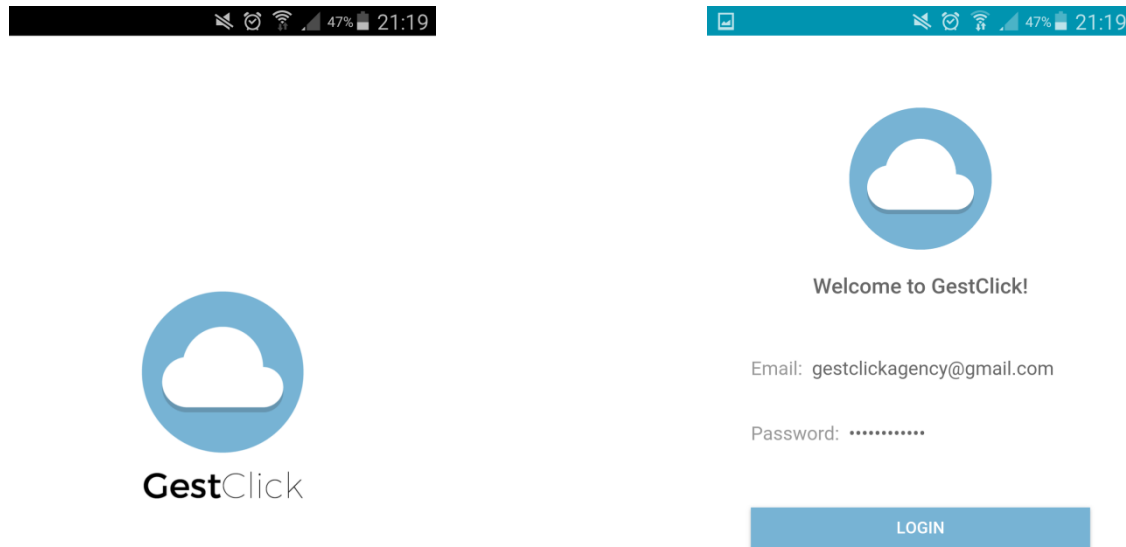
```

The script generates the APK file at: `.../android/build/outputs/apk`. Basically, it executes all the Ionic and Cordova commands to develop and build the mobile application, see section 4.6.7 for the explanation of these commands.

If something changes in the code, it is necessary to build it again, but changing the APP version. The version must be changed at these files: **config.xml**, **package.json**, **build\_android.sh**, **build\_ios.sh**.

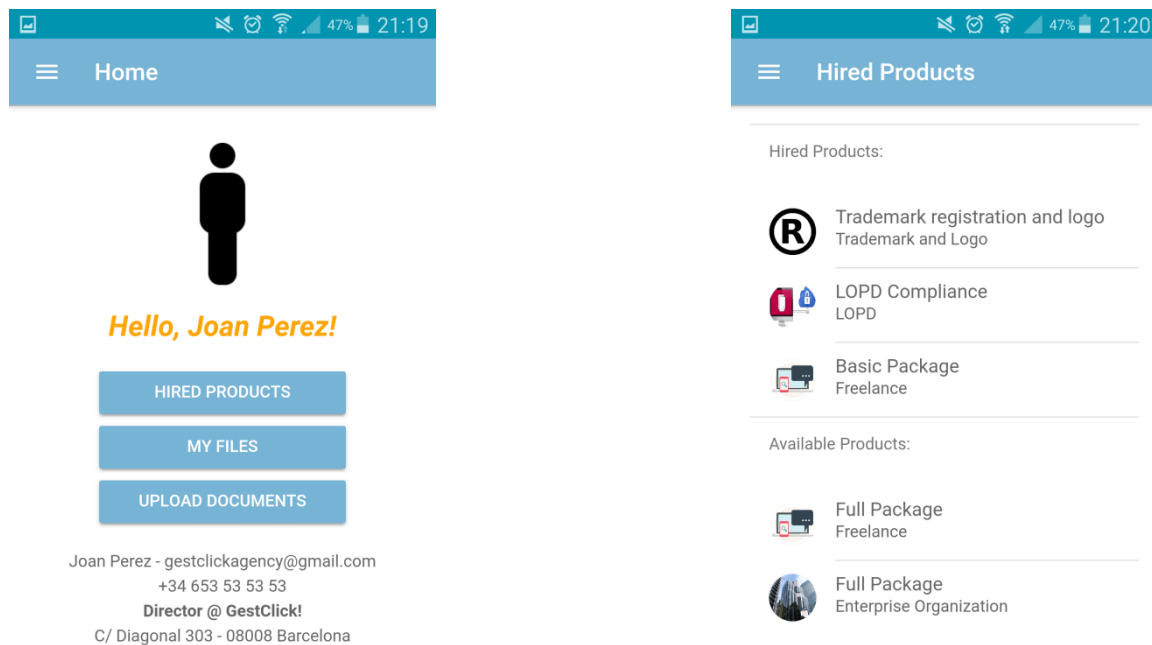
### 4.6.9 Final result

After compiling the app and distribute it we are getting the next screenshots from our phone. See to see the results.



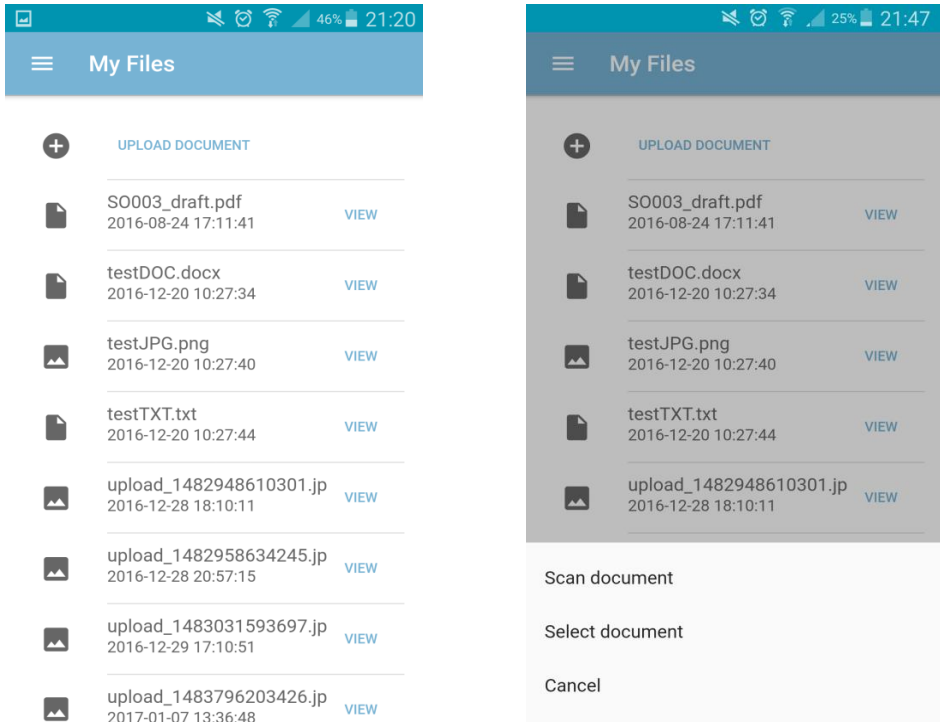
**Fig. 4.16 (Left)** Splash screen

**Fig. 4.17 (Right)** Screen of the login page of the app



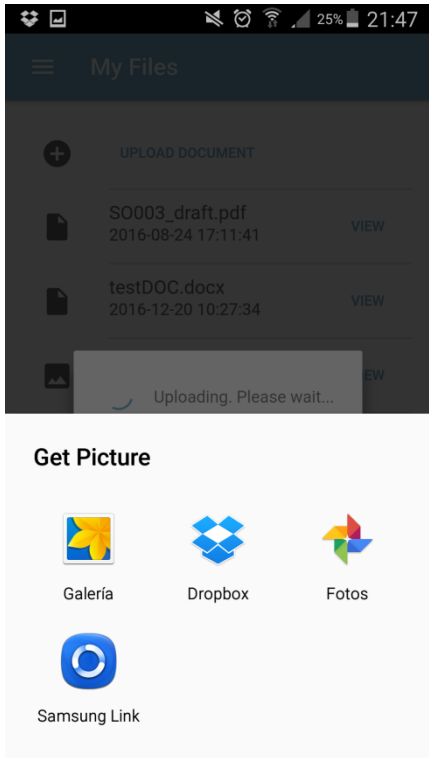
**Fig. 4.18 (Left)** Screen of the home page of the app

**Fig. 4.19 (Right)** Screen of the Hired Products page of the app



**Fig. 4.21** Screen of the My Files page of the app where users can read and upload documents

**Fig. 4.20** Screen of Upload document menu



**Fig. 4.22** Upload files from a document screen

## CHAPTER 5. CONCLUSIONS

World is changing very quickly and this project is a proof of it. Ten years ago, ERP solutions was just offered for the biggest companies in the world as IBM, Microsoft, SAP and so on. Nowadays there are reliable Open Source solutions used for the biggest companies in their emergent markets subsidiaries. Even more unbelievable is how the smartphone sector has changed in the last years. Also in that age the use of mobile applications (more known as Apps) was residual. Nowadays we cannot imagine our lives without Apps (almost 300.000M were downloaded during 2016).

All the sectors are moving so fast and it is very difficult to be updated in every technology even being a technology lover. For 2 students who were not familiar with ERP solutions neither mobile applications development, this project has been a huge challenge in many aspects.

First of all, having no previous experience at all working with ERP was an issue at the begging. But changing the point of view, we took it as a strength. The lack of knowledge of this technology let us do the benchmarking without any prejudice or pre-established idea about the products we were comparing. After this, choosing an open source solution made us get into a world where documentation and support are very different compared with proprietary software support. It was very helpful to get into the open source world where the development communities exchange support information on the Internet. But at the end, using Odoo, which is an open source solution still very new, added some extra complexity to the thesis because was not easy to find support information about specific terms online.

The usage of docker containers, to set the environment for the Odoo server, was also new for the team. We needed to study how to work in docker environments and with docker compose tools as well. Different from virtual machine usage, docker containers have their own tools to manage all the server instances. For example, copy an external file to the Odoo containers was not as simple as send this file to the server. Or vice versa, getting information from the server.

But the most challenging part of the project was to develop a mobile app from scratch. Without any experience in the field and without developing skills of any of the members, sometimes we thought it was an impossible mission. Studying the type of apps we can develop, getting updated about the technologies used in apps development or discovering new style-sheet protocols was just a small

portion of the many things we had to work during this thesis. The improvement of our problem-solving skills has been an important asset gained during this process. For instance, solving the CORS issue due to the technology chose and Odoo configuration. The learning of new coding languages as TypeScript was an important benefit, same as learn how to use the XML-RPC repository from zero. Also, designing an app and get in touch with design tools and try to make it as much user-friendly as we could was definitely a challenge. Getting in touch with this runtime environments as node.js or learning about Angular, Ionic or Cordova was to get into the latest technology. We also studied distribution tools like HockeyApp, which was interesting and useful for the future.

The requirements of the project were covered and the scope set at the beginning was accomplished. The first step for the technological evolution for the managing agency is completed and the owner proudly can show the new ERP system and the mobile applications to his customers.

The goals were difficult, especially for two people without a technical background on these fields but the fruits collected are even tastier after all the effort needed to build a solution like the one proposed in this project.

## **5.1 Global Enviromental impact**

This project consists in automatization using new technologies to develop an online managing agency.

The Odoo implementation of the website to offer an e-commerce with all the products available in a managing agency, it facilitates to the clients the inconvenience of the displacement to a physical office to contract any of these products. This feature helps the global environment by not producing CO<sub>2</sub> for the transportation. Furthermore, there is no need to use physical invoices, because once the product is ordered, the invoice is sent to the client via email and all of them are also available through the website.

Another feature offered by this solution that affects the global environment is the advantage offered by the Knowledge Management System. Using this module is possible to have an online file management system, through the website and the mobile application interface. Using this file management module of the ERP, there is no need to print all these documents and have a physical copy of them, saving money and waste of paper.