DEPLOYMENT OF A WEB PLATFORM IN THE CLOUD WITH MULTITENANT CAPACITY

Miguel Gallego Martín

Master’s Thesis presented to the
Telecommunications Engineering School
Master’s Degree in Telecommunications Engineering

Supervisors
Fernando Pérez González
Ignacio Fernández Bahamonde

2019-2020
Abstract

In the last years, with the success of Internet and the rapid development of storage and processing technologies computing resources have become cheaper, more available and powerful. This technological development has triggered the emergence of Cloud Computing, a new paradigm for services over the internet. In Cloud Computing resources can be rented to users following an on-demand payment model through Internet, this payment model is very attractive to enterprises because they do not have to care about maintenance or provisioning, availability or connectivity, neither binding contracts, they can begin with a small amount of resources and resize them immediately in order to supply an increase in the incoming traffic, following the business evolution. Because of these advantages nowadays a lot of companies are adopting the Cloud Computing paradigm, mainly enterprises that realised that exchanging the initial costs for a pay-per-use model was worth it, in addition to others that offer on-premise services installations and desire to increase their reach and expand horizons. This is just the case that we are dealing with here, 2Mares Demil, S.L.

2Mares has been 22 years working on technological specialized innovations, manufacturing intelligent software products for Contact Centers. They are leaders in Spain in their sector, serving the largest companies and now they want reach small entities, but there are a few problems about this, the on-premise installation cost is too much for this companies, and for 2Mares the maintenance cost in terms of support staff in comparison with the incoming benefit is not admissible. These are the main reasons because 2Mares decided jump to the Cloud Computing environment, initial installation costs would be removed and the need for support staff greatly reduced, in this way reaching small entities would be possible.

Key words: Cloud computing, Multi-tenancy, Software as a Service, Business evolution.
# Contents

Abstract iii

List of Figures vii

List of Tables ix

1 Introduction 1
   1.1 Context ........................................... 1
   1.2 Objectives ....................................... 2

2 Basic concepts 3
   2.1 Cloud Computing ................................. 3
      2.1.1 Cloud Computing definition .................. 3
      2.1.2 Cloud Computing features .................... 3
      2.1.3 Layered model ................................ 4
      2.1.4 Business model [1] ............................ 5
      2.1.5 Types of clouds [1] ............................ 5
   2.2 Multi-tenancy [2] ................................. 7
   2.3 Multi-tenancy in databases ....................... 8
   2.4 Contact Center ................................... 10

3 Development 11
   3.1 Understanding .................................... 11
      3.1.1 2Mares solution .............................. 11
      3.1.2 Logical servers ............................... 14
      3.1.3 Technology .................................... 15
   3.2 Design ............................................ 16
      3.2.1 Databases ..................................... 16
      3.2.2 Services and applications ................... 19
   3.3 Results ........................................... 24

4 Tests 33

5 Future lines 35
# List of Figures

2.1 Cloud computing architecture [1]. .............................................. 4
2.2 Multi-tenancy levels [2]. ....................................................... 7
2.3 Separate Database approach. .................................................. 8
2.4 Shared Database, Separate Schemas approach. ........................... 8
2.5 Shared Database, Shared Schema approach. ............................... 9

3.1 SQL Server pricing [3]. ......................................................... 17
3.2 Cloud with/without licensed editions. ....................................... 18
3.3 Configuration service & Bd Master functioning. ......................... 22
3.4 Master database architecture approximated drawing. .................. 24
3.5 Configuration service and some queries. ................................... 25
3.6 Drawing of the customer management tool, main view. ............... 26
3.7 Drawing of the customer management tool, customer management frame. 27
3.8 Drawing of the customer management tool, customer creation frame. ... 28
3.9 Drawing of the customer management tool, updating DBs. ............ 29
List of Tables

3.1 Task table with the amount of hours dedicated to each one. . . . . . . . . 31
1 Introduction

1.1 Context

2Mares want to reach as many small entities as possible, in order to achieve this, costs have to be minimized. Cloud computing has features that make it ideal for our scenario, however these characteristics have to be analyzed and compared with others from the rest of alternatives, in order to choose the best option.

There is a long path to walk in order to reach the Cloud computing environment, we have divided this path in different stages with different requirements:

- Demo: in the process of attracting new customers they need to have a preview of the 2Mares product, for this reason individual demo environments have to be deployed and configured, this process is time consuming and cumbersome. All these demo environments could be one unique environment, solving this issue. In order to achieve this stage the 2Mares product, databases and services have to be adapted to serve several customers.

- Customer with sub customers: sometimes one company is (in some way) divided internally, for example: they could have very differentiated departments or have bought other companies. The main difference from the previous stage is that this is not a testing environment, therefore harder tests have to be passed and the product adaptation completed.

- Cloud service: the final stage where 2Mares could finally serve several small companies, in order to reach the final stage is necessary to fulfill some privacy requirements, this would imply deployment and perhaps also product changes together with choosing the most suitable provider.

Taking into account the 2Mares product extension and the time available we have decided to redirect this work to the Demo stage.
Chapter 1. Introduction

1.2 Objectives

In this work, our goal is to reach the Demo stage, to achieve a multitenant environment where the 2Mares product can serve and be personalized for different customers. Below we list the objectives of this master’s thesis in detail:

- Study how 2Mares product architecture, services and databases, can be adapted from single to multi-client.
- Adapt the necessary services in order to be able to offer a product visual preview to the different customers.
2 Basic concepts

2.1 Cloud Computing

2.1.1 Cloud Computing definition

NIST definition of Cloud Computing [4]: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

2.1.2 Cloud Computing features

- Service automation.
  Typically Cloud Computing providers have a console, a wide API or both, a payment method has to be set and after that using these tools users can rent or release resources automatically, avoiding to lose time on human interaction.

- On-demand payment option.
  Users only pay the resources they are using, it is typically measured in hourly intervals. One of the most attractive features of Cloud Computing is that users can avoid the big initial costs of On-premise installations or hosting services.

- Fast scalability.
  Using the API or console users can scale their resources very fast, it is suitable for business that may tend to grow quickly.

- Elasticity.
  Elasticity is the property of an object that allows it to increase in size when it undergoes tension and recover its initial state when said tension remits.
  In this line most providers offer features that allow users of services to rent more resources automatically when the resource consumption grows and release them
when said consumption decreases.

- High availability.
  According to the Cloudonomics Laws [5] Cloud Computing providers offers a high availability infrastructure. The Cloudonomics Law #6 shows cloud service providers have the necessary scale to repel a bigger number of attacks in comparison with most of enterprises, as for example botnets’ attacks. And the Cloudonomic Law #9 shows cloud providers offer a high reliability system, having large and numerous data centers.

- Low latency.
  In the Cloudonomics Laws [5] way the Cloudonomics Law #8 shows how cloud infrastructures achieve a very low latency, deploying a few dozens of nodes, more nodes than an enterprise would want to deploy, taking into account their low utilization.

### 2.1.3 Layered model

The Cloud Computing environment architecture is conformed by four layers.

![Figure 2.1: Cloud computing architecture [1].](image)

- Hardware layer.
  This layer includes the physical resources and their management, responsibility of data centers.
2.1. Cloud Computing

- Infrastructure layer.
  Using virtualization technologies physical resources can be partitioned allowing dynamic resource assignment and maximizing resource usage.

- Platform layer.
  Consists of operating systems and application frameworks, the purpose of the platform layer is to minimize the burden of deploying applications directly into VM containers [1].

- Application layer.
  The cloud applications.

2.1.4 Business model [1]

Each layer described in the previous section can be a service to the layer above. In Figure 2.1 Cloud computing architecture [1] has been shown that cloud services can be classified into three categories:

- Infrastructure as a Service (IaaS).
  The provisioning of infrastructural resources is the service, typically VMs. Examples: Amazon EC2 and Flexiscale.

- Platform as a Service (PaaS).
  Operating system and application frameworks, platform layer resources. Examples: Microsoft Windows Azure and Google App Engine.

- Software as a Service (SaaS).
  The applications are the service, refers to the application layer. Examples: Salesforce.com and Rackspace.

2.1.5 Types of clouds [1]

Typically there are three types of clouds, choosing the right option depends on the enterprises’ interests, there exists a trade-off between reliability and security plus costs.

- Public cloud.
  Among the public cloud advantages stand out the disappearance of the initial costs and shifting infrastructure risks to providers. However there are some disadvantages such as lacking of precise control over network, security and data, crucial points in business.

- Private cloud.
  A private cloud is used by a single organization, therefore the highest level of
security, reliability and control over performance can be reached. However customer has to assume initial costs.

- Hybrid cloud.
  This type of cloud is a combination of the two types already mentioned, one part of the service infrastructure runs in a public cloud and the other part runs in a private one. Hybrid cloud offers flexibility but public and private components have to be determined carefully.

There are two other types of clouds not always mentioned.

- Community cloud [6].
  Provider clouds are replaced by resources of user machines to form a Community Cloud.

- Virtual private cloud (VPC) [7].
  Allows users to create a private network inside the public cloud. A private and isolated section of cloud infrastructure where a virtual network can be defined by the user, keeping benefits of public cloud like on-demand resource launching.
2.2 Multi-tenancy [2]

An architecture where resources are shared between different customers is called a multi-tenant architecture (tenants can be companies or individuals). In this way a higher utilization of resources is achieved as well as scalability and operational costs reduction. There are different ways to achieve multi-tenancy, we can distinguish different multi-tenancy levels, some examples are shown in Figure 2.2, each level has benefits and drawbacks, higher levels imply lower operational costs and better scalability, though customer flexibility and isolation decreases while architecture complexity increases.

![Multi-tenancy levels](image)

*Figure 2.2: Multi-tenancy levels [2].*
2.3 Multi-tenancy in databases

As stated in [8], the essential characteristic of SaaS is a multi-tenant database, a flexible data model has to be provided in order to satisfy the tenants’ needs. In [8] they also describe 10 different techniques to achieve a multi-tenant database, their benefits and drawbacks and performance experiments. From these 10, the 3 most typical techniques are the following:

- **Separate Database.**
  Each customer has his own database, it is the most common and simplest approach. Maximum isolation, security and customization are achieved compared with the rest of approaches. Backup restorations are the easiest too, though there is one drawback which is the cost. This technique has a lot of advantages but is the most expensive. For this reason it is suitable for services where the security and flexibility are the most important requirements.

  ![Figure 2.3: Separate Database approach.](image)

- **Shared Database, Separate Schemas.**
  Each customer has his own schemas and his own set of tables, sharing the same database, it is easy to implement and as flexible as the Separate Database technique. Tenants’ data are not completely isolated but there is some degree of logical isolation. In comparison with the Separate Database model more tenants can be supported by each database but backup restoration is more complicated.

  ![Figure 2.4: Shared Database, Separate Schemas approach.](image)
2.3. Multi-tenancy in databases

- Shared Database, Shared Schema.
  The last technique implies using the same database, schemas and tables for all tenants. Inside each table a tenant ID column would associate each row with his tenant.
  This approach in comparison with the two already mentioned has the lowest computational costs because it can serve the largest number of tenants per database. However, tenant isolation and backup restoration become big drawbacks. If it were necessary to restore certain tenant data a partial table recovery would have to be applied, only table rows associated to this tenant would have to be deleted and the old ones reinserted.

![Shared Database, Shared Schema approach](image)

*Figure 2.5: Shared Database, Shared Schema approach.*

Other approaches shown in [8] will not be mentioned because they are not interesting from the point of view of this work, we are looking for techniques that allow us to adapt an already existing data model, on this line techniques that imply a lot of changes at table level are not as good as others because a bigger engineering effort would be required.
# 2.4 Contact Center

A contact center is a centralized department within a company that manages customer interactions. Contact centers handle outbound and inbound omnichannel customer communication, using multiple channels such as telephone, email, fax, social media... among others. Customer issues are solved quickly through advanced technology.

The companies’ contact centers are the main channel used for most of the customer communications, they guarantee the best customer experience routing the customer requests to the most suitable agents, offering them self-service options and multiple communication channels.

Contact centers also help to improve business productivity, being able to scale staff if the number of customer requests grows and improving decision making using customer feedback and all data recollected from customer interactions.

Examples of companies that uses contact centers are the biggest telephone companies, when they call you offering new contracts or when you call them because your service is not working properly you are interacting with a contact center.
3 Development

The project development has been split in three different phases.

3.1 Understanding

I have been in 2Mares before a few months, however, my knowledge about their different services was not enough in order to proceed with this project, so a previous phase dedicated to the understanding of the product was totally necessary.

3.1.1 2Mares solution

2Mares offers a web solution that connects the Contact Center of Trademarks and Outsourcing systems sparse data to add intelligence, warrant the legal compliance, understand the customer experience and automate the continuous improvement.

- Smart Data
  Connects and normalizes the customer experience data in an unified repository. Warrants the legal compliance (GDPR, PCI-DSS, MiFidII).
  - Organizational model
    Integrates in one unique repository the organizational company structure, the work concepts and personal data, enabling the access to them in a user-friendly and tidy way.
  - Operative model and FCR
    Eases a typification master and the capacity to integrate the data generated in each contact point with the customer, being 100% compatible with the existent systems. This automates the KPIs (key performance indicator) obtaining as FCR (first contact resolution), CES (customer effort score), etc, in order to implement best practices.
Chapter 3. Development

- Omnichannel model
  Through Smart Data an omnichannel vision of customer is structured and normalized around the customer experience. It allows the integration of social network and the rest of usual channels (voice, email, chat...), behind the same operative system. The detailed data analysis obtained allows to elaborate strategies and ease the efficient decisions, that helps to reach the company’s goals.

- Team optimization
  Automates the continuous improvement cycle to boost the results and the engagement (Quality, Performance, Coaching, Training and Feedback).

- Quality management
  Eases the collaborative design of processes, knowledge management and best practices, ensuring the quality maintenance in an unique and integrated repository.

- Performance
  The 2Mares performance module normalizes the business metrics between organizational entities, services and systems. The real time operational state can be known through alarms and forms allowing the team to intervene in a proactive way in order to reach the goals.

- Coaching
  The 2Mares coaching tool allows the personalized action plans creation for each employee, based on the results obtained from the different evaluations. It also offers the possibility of following these plans and measure their effectiveness.

- Training
  This module goes hand in hand with the coaching module easing the training plans development that complements the identificative actions for employee’s improvement. In addition, the training module recommends the best training pills to cover the employee’s lacks and identify growing areas. It also measures the training impact.

- Employee feedback
  The Employee Feedback tool is the 2Mares solution that allows to obtain the employee’s opinion. This opinion is obtained through the employee’s comments about the evaluations, coaching and training sessions received. The result is translated into a better employee’s implication and company processes improvement.

- Speech analytics
  NLP technologies (Natural Language Processing), AI (Artificial Intelligence) and Machine Learning to analyze the customer voice and ensure the best practices.
3.1. Understanding

- Root causes analysis
  Finding the root-causes about the KPI breach is the base to boost effective action plans that impact on improvement metrics like the NPS (Net Promoter Score), reduce the CHURN (customer cancellation rate) or reduce the complaints. 2Mares crosses data from different fonts in an intelligent way to ease this analysis.

- Natural language processing (NLP)
  This module allows the analysis and data extraction from the phone conversations between customer and company. 2Mares automatically extracts key data to analyse the quality offered and perceived by the customer, adding elements for the processes improvement and company strategy definition. Thanks to previously defined models it allows to analyse the language, oral and written, employed in the customer contact points. This analysis helps to determine the customer experience with respect to the company, easing the strategy management because it is easier to direct this strategy towards the features most valued by the customer.

- Voice of the customer
  The Speech & Text Analytics application allows tagging automatically the topics and sentiment that difference the customer feedback. The feedback analysis through the customer or employee comments are a big source of time saving because it consolidates these data in forms focused on the action that allows “to close the cycle” of Customer Experience understanding.

- Artificial Intelligence (AI)
  2Mares employs Artificial Intelligence and in particular automatic learning tools (Machine Learning) in order to perform tasks autonomously that nowadays are done by people, in this way “robots” able to take automatic decisions with an error level under the human error level on repetitive tasks (effort and “no quality” costs now disappear) are implemented.

- Robot autocompliance
  Tool that uses the context data (Smart Data), the collective intelligence (Team Optimization) and the data extracted from conversations (Speech Analytics) to automate, without human effort need, the compliance verification of legal normatives, good practices fulfillment, automatic alarms generation for supervisors... among others. It allows the team to focus on value added tasks, improving the legal cover, the customer experience and costs saving in the offered and perceived quality assurance.
Chapter 3. Development

3.1.2 Logical servers

The 2Mares product is composed by a set of logical servers, each one is responsible of certain tasks within the product operation:

- Web Server
  2Mares web application that acts as the visible interface of the 2Mares product for the customer.

- Application Server
  Services set that manage the internal product logic.

- DB Server
  Server that contains the 2Mares databases, where customer data and product configuration parameters are stored.

- Reporting Server
  Services set that allows the generation of forms.

- Media Server
  Responsible of connecting to the multimedia data silos in order to serve to the users recordings, videos ...etc.

- Analytics Server
  Perform the Speech & Text Analytics processing over the multimedia files.
3.1.3 Technology

- **Visual Studio** [9]
  It is a Microsoft IDE (integrated development environment), used to develop web apps, web services, mobile apps... between others. Very user-friendly, includes a lot attractive features not only development-oriented also for version register and teamwork. The most remarkable ones are the teamwork features, using Visual Studio the company developers can work over different projects and even on the same using an unique and unified repository, combining their own changes with the changes of the rest of the team on an organized way, easily and even automatically. Minimizing the risks, preventing conflicts and errors.

- **ASP.NET** [10]
  .NET is a development platform composed by tools, libraries and programming languages for building applications.
  ASP.NET is an open source web framework that extends this platform with components in order to building web apps with libraries and tools. Between the elements included in the .NET platform we can find the C# programming language, libraries for working with different types of objects, editors and tools.

- **HTML, CSS and JavaScript**
  Front-end languages employed by any web app, used to define document structure, style, layout and client side operations.

- **C#** [11]
  It is a simple and modern programming language object-oriented developed and standardized by Microsoft, was approved as standard by ECMA and ISO.
  C# is similar to Java and to the C languages family and stands out for:
  
  - Exceptions control
  - Unused elements recollection
  - Security of types
  - Having an unified system
  - Code versioning

  This features allows to develop solid and durable applications.

- **SQL Server** [12]
  The central part of the Microsoft data platform. SQL Server is leader in ODBMS (operational database management systems).
3.2 Design

After understanding the 2Mares product functioning we enter the design phase, a new product architecture have to be designed in order to be able to server several customers in one unique environment.

3.2.1 Databases

The product base is his databases, so the first step is to take a decision about them. Now we have to take a look to the “Previous Concepts” chapter, “Multi-tenancy at databases” section, and taking into account the benefits and drawbacks described there take a decision.

Additional drawbacks has to be added to the “shared database” and “shared schema, shared database” options, we are facing a very extensive and already developed product and these options would imply too much query level changes and in the “shared schema, shared database” option practically rework the database architecture. These drawbacks are translated into an increase in development time.
3.2. Design

<table>
<thead>
<tr>
<th>Enterprise</th>
<th>Standard - per core</th>
<th>Standard - server + CAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal for</td>
<td>...</td>
<td>Core data management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and business intelligence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>capabilities for non-critical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>workloads with minimal IT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>resources.</td>
</tr>
<tr>
<td>model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td>Volume licensing,</td>
<td>Volume licensing,</td>
</tr>
<tr>
<td>availability</td>
<td>hosting</td>
<td>hosting,</td>
</tr>
<tr>
<td>level price</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(US$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Core data management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and business intelligence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>capabilities for non-critical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>workloads with minimal IT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>resources.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Server + CAL[4]</td>
</tr>
<tr>
<td>Web</td>
<td>Secure, cost effective and highly scalable data platform for public web sites. Available to third party software service providers only.</td>
<td>Free entry-level database that's ideal for learning, as well as building desktop and small server data-driven applications of up to 10 GB.</td>
</tr>
<tr>
<td>Ideal for</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Licensing</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td>Hosting only</td>
<td>Free download</td>
</tr>
<tr>
<td>availability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open no</td>
<td>See your hosting partner for pricing</td>
<td>Free</td>
</tr>
<tr>
<td>level price</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(US$)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[1] Customers who require a massively parallel processing data warehouse now have access to a parallel data warehouse through their Enterprise Edition core licenses with Software Assurance. Parallel data warehouse is part of the Microsoft Analytics Platform System.

[2] Editions sold in the per-core licensing model are sold as 2 core packs.

[3] Pricing represents open no level (NL) estimated retail price. For your specific pricing, contact your Microsoft reseller.

[4] Client access licenses (CALs) are required for every user or device accessing a server in the Server + CAL licensing model. See the product use rights for details.

Figure 3.1: SQL Server pricing [3].
An additional benefit has to be added to the “separate database” model, the database software used by 2Mares has a very high license costs, however, there is a free to use edition (Express) and although this edition has storage and computational resources limitations it is a very good option for a new project like this one, thinking about the cloud stage, we could allocate each customer databases set inside a different edition of our database software.

![Figure 3.2: Cloud with/without licensed editions.](image)

This would be also the more flexible option, if we had customers minded to pay more in exchange of having more storage capacity we could allocate these customers databases set inside a licensed edition.

Taking into account these new points and the company priorities, as data isolation and maintenance effort reduction, the “separate database” option was the option chosen.

However it is not enough to get a functional system, for every database access it is necessary to know to which databases set we have to aim, for this reason an additional database is needed, a database shared by all customers that registers which customer owns each databases set, the master database. This database will be also the place to store the product configuration parameters, from now on these parameters are split between common and specific of certain customer, because customer personalization need and software requirements.
Main components of master database:

- Customer registers: contains the customer identifiers, name, enabling flags and licenses.
- Common data: contains the data shared by all the customers, it is typically 2Mares system related data, as for example IPs, ports and load balancing parameters of the different 2Mares services.
- Customer specific data: contains the data related to certain customer, linked by a foreign key, as for example the parameters to connect to their databases and part of the configuration parameters of the product.
- Version registers: contains the versions that were applied since the database was created, version numbers and commentaries about the changes of each version.

Configuration parameters have to be divided between common and specific, it is easy split the parameters in the examples mentioned above but they are not the typical cases, most of parameters could be specific or common depending on if we are disposed to modify even more the services functioning. The product has hundreds of parameters and would not be viable replicate them for each customer, the support staff needed to configure and maintain this would be increased drastically with the number of customers, for this reason being able to define as much common parameters as possible is one of the key points of this project.

3.2.2 Services and applications

Over the databases we have the product services, with a services set for each customer and taking into account the “separate database” model we would achieve the biggest data isolation, however we also would suffer the biggest costs, there would not be a notable difference between this and deploying a different environment for each customer, for this reason multi-tenancy is applied at service instance level, although this solution implies the need to manage the customer data isolation at code level. In this way a single set of services would be able to serve all the customers, this compared to deploy for each customer the same set of services in a different machine is translated into cost and maintenance effort reduction, this reduction scale with the number of customers and means a lot for a little company like 2Mares, being this one of the key points that makes this project really worth it.

3.2.2.1 Customers management tool

The first step, before changing any service, is to develop a tool to manage the different customers, now will show a list of the different use cases defined for this tool:
Chapter 3. Development

- Create a new customer
  A way to introduce new customers in the system is needed, together with associated data, for example: an alias, company name, contact information, additional identifiers, license, enabling indicator... between others.

- Create and assign automatically a database set on customer creation
  Taking into account the scalability need (from the point of view of a cloud environment) and the maintenance effort reduction requested by the company we focused ourselves on achieving the highest automation level we can.

- Enable/Disable the different customers
  If for some reason we need temporarily to take out a customer from the system, the services need an indicator to know which customers are in the system and which are not. The tool has to be able to switch on or switch off the mentioned indicator.

- Upload customer licenses
  Each customer will have his own licenses and these have to be uploaded after customer creation or when the current license expires or the contract terms change.

- Update the databases of various customers at same time
  Taking into account the maintenance effort reduction point, it would not be suitable updating customers databases one by one, so we need be able to update various types of databases of different customers at same time.

- Update the system configuration parameters of various customers at same time
  The product have to be configurable depending on the customer, for customer needs, personalization priorities, or just because the software requires that, so there are some configuration parameters replicated for each customer with different values. This implies that when new parameters are introduced to the system the already existing customers parameters have to be updated, as we have said in the last points it is because the scalability point that is directly translated into a maintenance effort reduction.

- Delete a customer together with his database set
  Following the argumentation of the last points deleting database by database after customer removing would not be suitable.

- Visualize and be able to modify the system configuration parameters of certain customer comfortably.
  There is a need to access the configuration parameters of the different customers, modifications could be required after customer creation, so a user-friendly interface has to be developed.
3.2. Design

3.2.2.2 Customers management libraries

In order to guarantee the customer data isolation at code level libraries destined to the customer identification and management was designed.

At code level the product is composed by several software layers, each one of these layers consists of a set of components or libraries: generic libraries, DAO libraries, service-access libraries, file-access libraries, between others.

The new libraries have to integrate with the rest of components (DAO libraries need to know which database set they have to use, service-access libraries need send to the services the right customer identifier, file-access libraries need to know which customer files they have to access) and at the same time with all layers. Once the customer identifier is set this identifier has to be available from any layer in an unified and real-only way, in order to ensure data isolation.

The time and effort needed to adapt the rest of services depends on the facilities to access the customer identifier in the different layers, the easier to access the customer identifier the less changes at level code will be needed taking into account that we are working on an extensive product, for this reason the design of these libraries turns into one of the key points of this project, together with in-depth researching and exhaustive tests in order to find the best mechanisms to carry out this task.

3.2.2.3 Configuration service

This service belong to the Application Server and is queried by the rest of services to obtain the product configuration, it is the product basis, the rest of services depend on him to obtain the necessary parameters to develop their tasks.

From now on this service does not handle one unique configuration anymore, different configuration depending on the customer requested will be returned. It is translated into a partial service rework, a challenging task taking into account that the changes made over this service must not have any consequence over the current customers (on-premise customers).

This service has to work over the master database from now on, the new database that contains the customer specific data and common system data, common system data as IPs or ports to communicate with the rest of services or to initialize themselves, customer specific data as connection parameters to access their databases, customer licenses to find out the enabled features... between others.
Figure 3.3: Configuration service & Bd Master functioning.
3.2. Design

3.2.2.4 2Mares Web

This service belongs to and conforms the Web Server, the visual interface of the 2Mares product. Customer users use this service in order to login on the 2Mares system and begin to use the 2Mares solution features (mentioned in understanding section). This service query the service previously mentioned in order to get the customer features and configuration, with these parameters access to databases and query other services in order to get the necessary data, depending on the functionality the user is employing, interaction with other services could be required. The service also allows the administrators personalize the shown visual themes, uploading company logos or configuring identificative colors.

In order to achieve the objectives defined by the company (be able to show to the customers a product visual preview) data and visual themes have to be shown depending on the customer logged. This implies almost all the database access, calls to the configuration service and other services necessary to perform the login operation and show certain data.

The first complication, already mentioned, is that the product is not going to be completely adapted at once, so the adapted product has to be compatible with the not adapted product, the services has to be able to receive customer filtered requests and the old requests.
3.3 Results

After the understanding and design phases the implementation can begin.

- First master database was implemented, databases are the product basis and for this reason this was the first step. The process was a bit slow, if mistakes were made would be difficult in future to redesign the database structure, so it was important to do a good quality design, around a month was needed in order to approve the database design together with the terminology that the project includes.

![Master Database Architecture](image)

*Figure 3.4: Master database architecture approximated drawing.*
3.3. Results

- After the master database the configuration service was the next step, to be able to extract the customer registers and configuration parameters, they are a tandem. Old access was maintained in order to enable compatibility with no multi-Tenant adapted services and on this way avoid disturb the current system operation.

Figure 3.5: Configuration service and some queries.
Once we had the configuration service we began with the customer management tool, whose use cases were already commented in the design section. The application allows to manage the multi-Tenant environment.

1. A drop down list that allows select the parameters you want to manage, the common parameters or the parameters specific of certain customer can be selected. By default the common parameters are selected as we can see in the drawing.

2. The parameters classification, the parameters are ordered in categories and components, some of them related to a logical server of 2Mares or related to a specific service.

3. After selecting the desired classification the parameters are shown in this window, server and name of the parameter together with his value and a description. These parameters can be modified clicking them and selecting the "Apply changes” button.

4. This button allows to download the descriptions of the selected parameters, it is useful when someone is new using the system or new parameters are introduced.
5. This button opens the customer management frame that allows create, delete and update the different customers together with their databases.

<table>
<thead>
<tr>
<th>ID</th>
<th>Sec ID</th>
<th>Name</th>
<th>Enabled customer</th>
<th>Enabled flag X</th>
<th>DB1 Version</th>
<th>DB2 Version</th>
<th>DB3 Version</th>
<th>DB4 Version</th>
<th>DB5 Version</th>
<th>DB6 Version</th>
<th>Licenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CS1</td>
<td>Customer1</td>
<td>True</td>
<td>False</td>
<td>1.0.0</td>
<td>1.0.1</td>
<td>Not found</td>
<td>1.0.0</td>
<td>Conn Error</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CS2</td>
<td>Customer2</td>
<td>True</td>
<td>True</td>
<td>1.0.1</td>
<td>1.0.1</td>
<td>1.0.0</td>
<td>1.0.0</td>
<td>1.0.0</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CS3</td>
<td>Customer3</td>
<td>False</td>
<td>False</td>
<td>1.0.1</td>
<td>1.0.1</td>
<td>1.0.0</td>
<td>1.0.0</td>
<td>1.0.0</td>
<td>ERROR</td>
<td></td>
</tr>
</tbody>
</table>

... ... ... ... ... ... ... ... ... ...

Figure 3.7: Drawing of the customer management tool, customer management frame.

(a) This button allows to select the licenses file and upload it to the master database.

(b) This button allows to update the customer parameters, after selecting the version you want to apply.
(c) This button opens the create customer frame.

We can press the button on the top if we want a totally random secondary identifier. There is a checkbox that allows create the databases of the new customer automatically.

**Figure 3.8:** Drawing of the customer management tool, customer creation frame.
3.3. Results

(d) This button allow us select a script to apply over one database of the selected customers.

<table>
<thead>
<tr>
<th>ID</th>
<th>Sec ID</th>
<th>Name</th>
<th>Enabled customer</th>
<th>Enabled flag X</th>
<th>DB1 Version</th>
<th>DB2 Version</th>
<th>DB3 Version</th>
<th>DB4 Version</th>
<th>DB5 Version</th>
<th>DB6 Version</th>
<th>Licenses</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CS1</td>
<td>Customer1</td>
<td>True</td>
<td>False</td>
<td>1.0.0</td>
<td>1.0.1</td>
<td>Not found</td>
<td>1.0.0</td>
<td>Conn Error</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CS2</td>
<td>Customer2</td>
<td>True</td>
<td>True</td>
<td>1.0.1</td>
<td>1.0.1</td>
<td>1.0.0</td>
<td>1.0.0</td>
<td>1.0.0</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CS3</td>
<td>Customer3</td>
<td>False</td>
<td>False</td>
<td>1.0.1</td>
<td>1.0.1</td>
<td>1.0.0</td>
<td>1.0.0</td>
<td>1.0.0</td>
<td>ERROR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.9: Drawing of the customer management tool, updating DBs.

The script is executed over the database of all the selected customers until an error happens, then the execution stops and we can click over the cell in order to see the error message. After this we can press the "Continue execution" button if we want to continue the script execution or the "Stop execution" button if we want to stop the execution, every time an error happens the execution will stops and if there is more than one error we can download all them by pressing the "Download error logs" button. When the execution ends (if there are errors) a dialog will answer us if we want download the error logs.
Chapter 3. Development

• The last step was the 2Mares Web, the user interface, it was one of the hardest because of his size, at the end it was possible to visualize the correspondent data together with the visual themes, even some functionalities was working, achieving widely the project objectives.

Despite in the design section only a couple of services and an application were mentioned, the rest of services were designed too, however these services were not mentioned due to they did not got implemented because were not necessary to achieve the objectives.

Before continuing with the implementation it was necessary:

– Test the changes done
  A lot of big changes have been done, before continuing it is necessary to check the performance did not decrease, the consumption did not got increased and solve any error that may appear.

– Solve bugs
  A guard period has to be taken into account because if errors appear more time would be needed in order to solve these bugs and end the project.

– Write records for each change done
  As has been said in the last points a lot of changes have been done, for this reason it is extremely important to register each step given, in order to solve future errors, to redesign if was necessary and to train staff, helping them to save time in product understanding and minimizing the error risk because of misunderstandings.

– Write user manuals
  The product deployment has also changed and in consequence user manuals have to be done to speed up the testing and ease the staff understanding. It is similar to the previous point but it is support staff oriented.

– Train the staff
  Together with documentation and user manuals it is important to perform a couple of meets to solve staff questions that could come up after the documentation reading.
3.3. Results

- In order to finish this chapter we have a table with most of 2Mares product components created or adapted in the course of this master’s thesis, together with the amount of hours invested on each one, breaking down this hours into: hours dedicated to understand the component, time dedicated to design the component together with the researching time on certain cases and to end the hours dedicated to implement the design previously made.

<table>
<thead>
<tr>
<th>Component</th>
<th>Understanding</th>
<th>Design</th>
<th>Developing</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases</td>
<td>24 h</td>
<td>120 h</td>
<td>30 h</td>
<td>174 h</td>
</tr>
<tr>
<td>Configuration service</td>
<td>18 h</td>
<td>60 h</td>
<td>48 h</td>
<td>126 h</td>
</tr>
<tr>
<td>Customer management libraries</td>
<td>-</td>
<td>90 h</td>
<td>12 h</td>
<td>102 h</td>
</tr>
<tr>
<td>Customer management application</td>
<td>-</td>
<td>78 h</td>
<td>48 h</td>
<td>126 h</td>
</tr>
<tr>
<td>2Mares Web</td>
<td>60 h</td>
<td>60 h</td>
<td>90 h</td>
<td>210 h</td>
</tr>
<tr>
<td>Other services implemented</td>
<td>48 h</td>
<td>80 h</td>
<td>110 h</td>
<td>238 h</td>
</tr>
<tr>
<td>Other services not implemented</td>
<td>60 h</td>
<td>94 h</td>
<td>-</td>
<td>154 h</td>
</tr>
<tr>
<td></td>
<td>210 h</td>
<td>582 h</td>
<td>338 h</td>
<td>1130 h</td>
</tr>
</tbody>
</table>

Table 3.1: Task table with the amount of hours dedicated to each one.
During the development some tests were performed in order to bear out the design viability.

- Customer management libraries
  The new libraries, destined to manage the customer identifier, received most of the tests. We have to ensure:
  - The libraries are valid for any case
    In other words, under normal conditions the customer identifiers can not be swapped or lost. A lot of researching was made for the used objects, trying find bugs, failure conditions or limitations that could produce malfunctions in said objects. An environment with more than 100 customers and 1000 users was simulated several times, achieving a 0% of failures on each test.
  - We have a complete knowledge of used objects
    Other environments was also simulated in order to ensure the objects perform in the way described by the Microsoft documentation.
  - The libraries are fast enough
    Tests to check that libraries acceses are fast enough was performed, in order to ensure that they are not going to slow down the product current processes.

- Resources consumption
  The product adaptation was accompanied by a lot of changes on product logical processes and the creation of new ones, for this reason we have to ensure that the resource consumption was not increased in the case of one unique domain, these are the most important test because we have to avoid by all means this project disturb the current customers. After testing it was demonstrated that the resource consumption did not change in spite of the product adaptation.

- Maintenance effort
  One of the problems this project had to solve, in order to be able to serve several
customers, was ease the product configuration, in order to reduce the support staff necessary to manage the product. The product (and above all the customer management application) was tested by the support staff of the company and they confirmed the maintenance effort reduction managing various domains.
From now on the Demo stage would be progressively completed once the main changes are applied and support and development staff understood the new product functioning.

Once the Demo stage is completed the Customer with customers stage would begin, more testing would be needed because the aim is a product to be used by customers, however, this would be of short duration compared to the Demo stage because it is about corrections.

Finally, we would reach the last stage, the Cloud stage, a cloud provider would be needed together with a new service in order to automate task through the cloud provider API, also the best cloud deploying has to be chosen, there is a lot of possibilities and a thorough study has to be done. However the most important points are the security and privacy, the product is not designed to be open to internet, so changes would be need in order to correct this.
Bibliography


Bibliography

