PEOPLES DEMOCRATIC AND REPUBLIC OF ALGERIA
MINISTRY OF HIGH EDUCATION AND SCIENTIFIC RESEARCH

IBNKHALDOUN UNIVERSITY – TIARET

Master Thesis

Introduced to:

Mathematics and Computer Science faculty
Computer Science department

For obtaining:

MASTER DEGREE

Specialty : Computer Engineering

by :

BENKHALLEF Silia **ABER Amel**

About:

Design and Realization of Cloud SaaS Multi-Tenant Application

Submitted on: 19 / 09 / 2022

Academic year: 2021-2022

Acknowledgments

Above all, we thank God who has given us the strength, will and patience to accomplish this work.

We would like to thank our supervisor, Mr HATTAB Noureddine, for suggesting this topic to us and for his guidance, his advice and encouragement were invaluable to us.

We also extend our heartfelt thanks to the members of the jury who graciously devoted their time to evaluate this modest work.

We hold a very special place for our dear families and friends, and to all of them we express our deepest gratitude for their encouragement and support.

Dedicate

I dedicate this crop of my work to My father, without whom i would not have reached what i am now, and the credit and support i received from my mother, this is all the fruit of your love and support, may God prolong your life.

To my dear brothers Ghali and Amine

To the jewel of my life, my sisters Nabila and Nour elhouda

To the supervising professor, thank you for your trust that we are capable of carrying and working

To my partner in this business BENKHELLAF Silia, it was a pleasure working with you

To all my friends Dalila, Farida, Hafidha, Nesrine and Asia and everyone who supported me and encaouraged me.

Amel

Dedicate

I dedicate this work:

To my family who endowed me with a worthy education, her love made me who I am today.

To my dear parents "Slimane and Amel" for all their sacrifices, their love, their tenderness, their support and their prayers throughout my studies.

To my dear sisters Ibtissam, Kamelia, Lidia, katia for their constant encouragement and moral support.

To my dear brothers Kamel and Chakir for their support and encouragement.

To my partner in this business ABER Amel, it was a pleasure working with you

To the entire IT team at the National Unemployment Insurance Fund (CNAC) Mr. Hiresche Sofiane, Mr. Farsi Mohammed and Ms. Haider Mahjouba for all their efforts and support

To all my family for their support throughout my .

Silia

Abstract

With the rapid and increased technological development, several architectures and approaches have been superseded by new ones that have appeared to correct the deficiencies of their predecessor; and multi-tenant architectures are indisputably among those that have become a necessity as an added architectural value for companies with virtualized systems (Cloud, Fog, Edge..etc), and as a skill for engineers and developers.

In order to remove all the ambiguity which resides in this architectural tendency, our contribution behind this work (of application and not of research), is based on the use of the multi-tenant architecture of the highest degree "unique data base" when designing and building a simple Cloud SaaS application. This will allow to demystify the conceptual and programming complexity of this architectural trend, based on a SaaS application for healthcare establishments that supports the demands of its tenants.

In order to realize the distinguished multi-tenant SaaS application, the Laravel Framework is targeted by this thesis.

Keywords: Virtualized System, Cloud, SaaS Model, Multi-Tenancy, MIMT, SIMT.

****

Résumé

Avec le développement technologique rapide et accru, plusieurs architectures et approches ont été remplacées par de nouvelles qui sont apparues pour corriger les carences de leur prédécesseur; et les architectures multi-tenantes font indiscutablement partie de ceux qui sont devenus une nécessité en tant que valeur architecturelle ajoutés pour les entreprises avec des systèmes virtualisés (Cloud, Fog, Edge..etc), et en tant que compétence pour les ingénieurs et les développeurs.

Afin de levez toute l'ambigüité qui réside dans cette tendance architecturelle, notre contribution derrière ce sujet d'étude (non pas de recherche), repose sur l’utilisation de l'architecture multi-tenante du degré le plus élevé "Base de donnée unique" lors de la conception et la réalisation d'une simple application Cloud SaaS. Cela permettra aux utilisateurs de démystifié la complexité conceptuelle et surtout de programmation de cette vision architecturelle, en s'appuyant sur une application SaaS pour les établissement de santé qui prend en charge les demandes de ses tenantes.

Afin de réaliser l'application SaaS multi-locataire distingués, le Framework Laravel est visé par ce mémoire.

Mots-clés: Système Virtualisés, Cloud, Modele SaaS, Multi-Tenancie, MIMT, SIMT.

****

ملخص

مع التطور التكنولوجي السريع والمتزايد ، تم استبدال العديد من الهياكل والنهج بأخرى جديدة ظهرت لتصحيح أوجه القصور في سابقتها ؛ والبنى متعددة المستأجرين هي بلا منازع من بين تلك التي أصبحت ضرورية كقيمة معمارية مضافة للشركات ذات الأنظمة الافتراضية وكمهارة للمهندسين والمطورين) سحابة ، ضباب ، حافة .. إلخ (من أجل إزالة كل الغموض الذي يكمن في هذا الاتجاه المعماري ، فإن مساهمتنا وراء هذا الموضوع من الدراسة (وليس البحث) تستند إلى استخدام بنية متعددة المستأجرين لأعلى درجة "قاعدة بيانات موحدة" عند تصميم وبناء تطبيق سحاب بسيط.
سيسمح ذلك بإزالة الغموض عن التعقيد المفاهيمي والبرمجي لهذا الاتجاه المعماري ، استنادًا إلى تطبيق سحابي لمؤسسات الرعاية الصحية التي تدعم .متطلبات المستأجرين .
من خلال هذه المذكرة. تم استهداف اطار laravel من أجل تحقيق التطبيق المميز المتعدد المستأجرين .

الكلمات المفتاحية:

النظام الافتراضي ، السحابة ، نموذج SaaS ,الإيجار المتعدد ، MIMT، SIMT

Contents

List of Figures

List of Tables

Abbreviations

SaaS	Software As-a-service
PaaS	Plateforme As-a-service
IAAS	Infrastructure As-a-service
MIMT	Multiple instance multiple Tenant
SIMT	Single instance multiple Tenant
NIST	National Institute of Standards and Technology
CISCO	Commercial & Industrial Security Corporation
SLA	Service level Agreement
Amazon EC2	Elastic Cloud 2
ERP	Enterprise Resource planning
CRM	CRM for Small-Medium
QoS	Quality of Service
MVC	Model view Controller

General Introduction

1 Introduction

The Cloud is a distributed computing paradigm in full growth based on virtualization, widely adopted for delivering services over the Internet. The adoption of cloud computing has revolutionized how we look at service science (SS).Among the three models offered by the Cloud, software as a service (SaaS) represents the most attractive service delivery model . In SaaS, and unlike traditional single license models, SaaS allows consumers to pay a subscription for on-demand services. The work predicted that the market for cloud-based services would evolve dominantly and rapidly. [1]

1.1 Context

With the rise of SaaS and the popularity of service-oriented paradigms, the concept of multi-tenancy has recently aroused great interest in the cloud application community. Multi-tenancy in the Cloud represents a promising **research** avenue, which still arouses a great deal of interest. This interest comes mainly from its ability to share (according to levels of maturity) hardware/software resources with several "consumer" tenants of the Cloud, i.e., to increase the profits of current Cloud applications.

In the era of the proliferation of software as a service (SaaS) and cloud applications, the impact of the Single Instance Multi-Tenante (SIMT) model is enormous on the cloud market.

Multi-tenancy is seen differently with respect to different service models. However, Multitenancy has different meanings in the different service layers of the cloud model "IaaS, PaaS, and SaaS". This work focuses on the "SaaS" multi-tenancy category, with the highest isolation level. Several consumers share a Cloud application instance. At this stage, two main models are described in: the Multi-instance Multi-tenant (MIMT) and the Single-Instance to Multi-tenant (SIMT). In the MIMT model, a tenant uses a dedicated runtime application instance. On the other hand, in the SIMT model, the tenants simultaneously share the same application instance. Among the advantages of these two models , we can cite the reduction of service costs due to resource sharing, which will lead to an increase in utilization.

1.2 Problem, Goal, and Thesis Statement

Consumer and firms prefer to pay as little as possible for hosting or software use. With applications that are designed and modified for a specific tenant and then hosted as an individual instance, hosting costs increase linearly per tenant.

The gap between both the traditional hosted and cloud paradigms has necessitated the creation of multi-tenant solutions. In order to serve numerous application tenants, these multi-tenant solutions construct a single application instance, allowing each customer to enhance, customize, or alter the application as if they were the only tenant.

Multi-tenant solutions are a powerful alternative for companies looking to gain differential benefit. This is the reason the study introduces exactly such a firm as a case study.

Our dissertation fits in this context of software system as a multi-tenant service, a context that mainly contains two conceptual voices, firstly multi-database and secondly single-database.

This study focuses on understanding what multi-tenancy is and why it is relevant to the design of cloud based solutions. A case study is used to provide a relevant context to research from which design are drawn and finally a simple app implementing multi-tenancy for the specific case requirements is derived.

The ultimate goal of this work is to develop an application for health firm, facilities with this non-traditional architectural vision, which is based on registration (basic/advanced) to benefit from its services. The Uni-database direction has been selected to carry out this work via the Laravel framework.

1.3 General Structure

In the first chapter, we will talk about one of the most commonly used paradigm and virtualized environment today is being cloud computing and making a quick bibliographic research on this paradigm, in particular the SaaS service model.

In the second chapter, we have given an overview of multi-tenant architecture its definition, advantages and disadvantages. Additionally, the various SaaS maturity level are described in relation to multi-tenancy, highlighting how crucial multi-tenancy is to creating quality SaaS applications.

In the third chapter, we have detailed the Functionalities and behavior of the proposed application using the UML language , additionally, we presented the implementation process to express the programming language, the development environment used and finally we illustrated the results of our application.

CHAPTER ONE

1.1 Introduction

The Cloud is a distributed computing paradigm in full growth based on virtualization , widely adopted for delivering services over the Internet. In the rest of this chapter we try to quickly detail the terminology and the essential models of cloud paradigm.

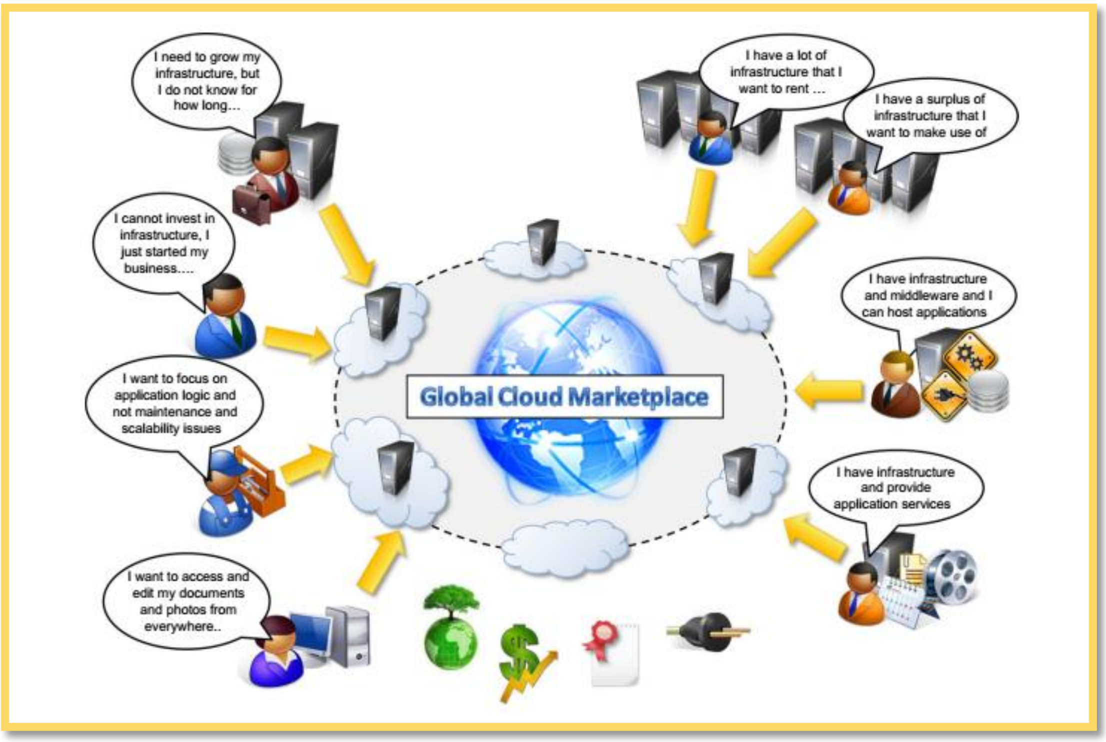

I need to grow my infrastructure, but I do not know for how long…
I have a lot of infrastructure that I want to rent …
I have a surplus of infrastructure that I want to make use of
I cannot invest in infrastructure, I just started my business….
I have infrastructure and middleware and I can host applications
Global Cloud Marketplace
I want to focus on application logic and not maintenance and scalability issues
I have infrastructure and provide application services
I want to access and edit my documents and photos from everywhere..

Figure I.1: Cloud Computing vision.[2]

1.2 History of Cloud Computing

The foundations of cloud computing can be traced back to the 1960s when John McCarthy, a pioneer of artificial intelligence, first formulated the idea of "Utility computing". The idea is to be able to provide the user with computing power, storage capacity, and communication capacity, in the same way, they are supplied with electricity or water in public networks. [3]

Long before the birth of cloud computing, computer scientists already used cloud computing services such as webmail2 or online data storage or sharing information about social networks. [4]

In the 90s, another concept had already prepared the ground for cloud computing. This is the ASP that allowed the customer to rent access to the software installed on the remote servers of a service provider, without installing the software on its own machine. [4]

Cloud computing adds to this offer the concept of elasticity with the possibility of adding new users and services with a single mouse click.[4]

Cloud computing has finally appeared with the different technological progress made during the last 50 years, as long in terms of hardware, software, and conceptual, as to the progress of the safety mechanisms and the development of standardized networks such as the Internet, and Experience in software publishing and managing data, services, infrastructure, and data storage.[4]

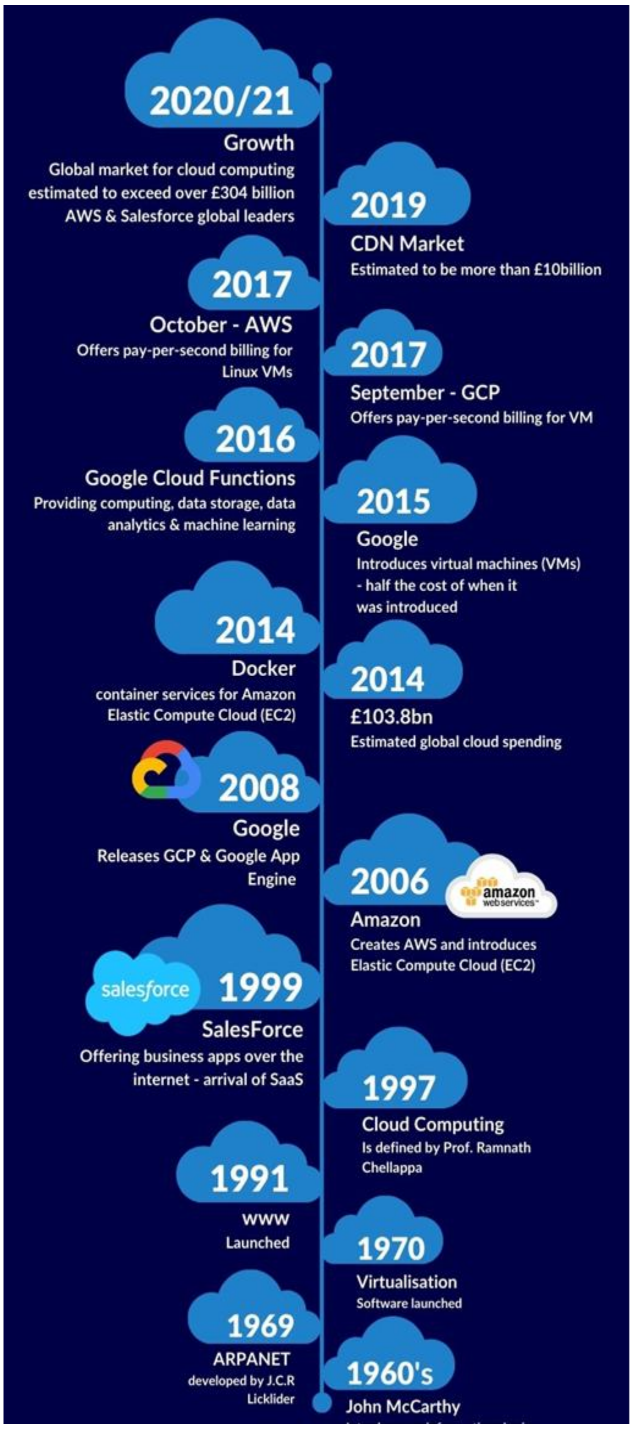

2020/21
Growth
Global market for cloud computing estimated to exceed over £304 billion AWS & Salesforce global leaders
2019
CDN Market
Estimated to be more than £10billion
2017
October - AWS
Offers pay-per-second billing for Linux VMs
2017
September - GCP
Offers pay-per-second billing for VM
2016
Google Cloud Functions
Providing computing, data storage, data analytics & machine learning
2015
Google
Introduces virtual machines (VMs) - half the cost of when it was introduced
2014
Docker
container services for Amazon Elastic Compute Cloud (EC2)
2014
£103.8bn
Estimated global cloud spending
2008
Google
Releases GCP & Google App Engine
2006
amazon web services
Amazon
Creates AWS and introduces Elastic Compute Cloud (EC2)
salesforce
1999
SalesForce
Offering business apps over the internet - arrival of SaaS
1997
Cloud Computing
Is defined by Prof. Ramnath Chellappa
1991
www
Launched
1970
Virtualisation
Software launched
1969
ARPANET
developed by J.C.R Licklider
1960's
John McCarthy

Figure I.2: History of Cloud Computing [5]

1.3 Definition

There are countless definitions and interpretations of cloud computing to be found from multiple sources. In the simplest terms, cloud computing means storing and accessing data and programs over the Internet from a remote location or computer instead of our computer's hard drive. This so called remote location has several properties such as scalability, elasticity etc., which insignificantly different from a simple remote machine. The cloud is just a metaphor for the Internet.[6]

The definition of cloud computing proposed by the National Institute of Standards and Technology "NIST" is considered the most cited definition in the literature, according to NIST:

"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".[7]

According to CISCO:

"Cloud computing is an IT pooling platform providing companies with on-demand services with the illusion of infinite resources".

One of the essential points of these definitions is the notion of "scalability"; scalability on demand, elasticity, i.e. you only pay for what you use. This is a considerable advantage compared to a company-specific infrastructure where servers are very often underutilized. [8]

1.4 Essential Characteristics of cloud computing

On-demand self-service

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider. [9]

Broad network access

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations. [9]

Resource pooling

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).

Examples of resources include storage, processing, memory, and network bandwidth.[9]

Rapid elasticity

Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.[9]

Measured service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, active user accounts). Resource usage can be monitored, controlled, audited, and reported, providing transparency for both the provider and consumer of the utilized service. [9]

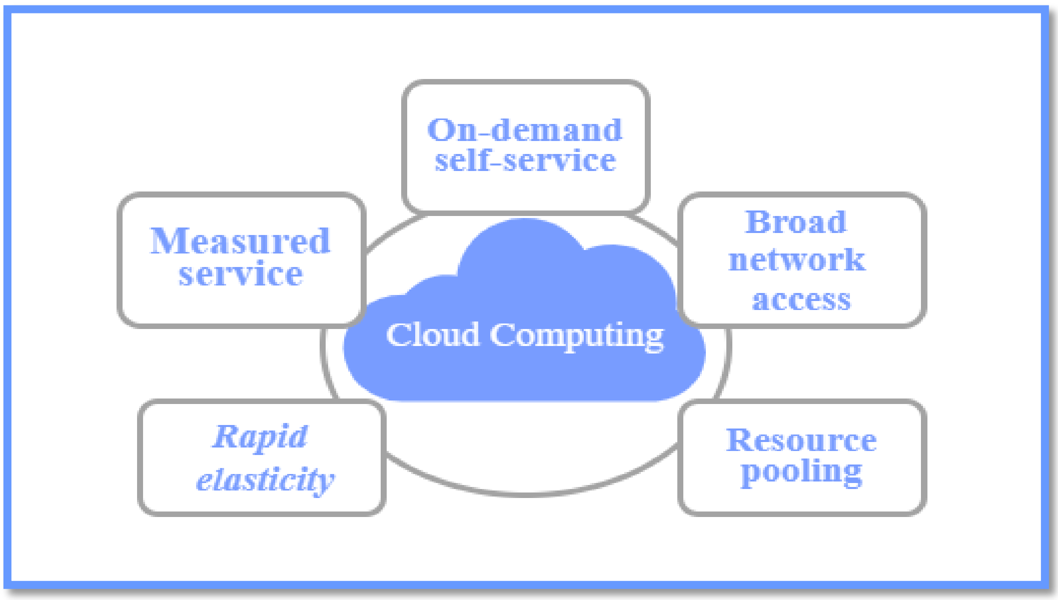


Figure I.3: Essential Characteristics of Cloud computing. [3]

1.5 Types of cloud computing

For deploying a cloud computing solution, the major task is to decide the type of cloud to be implemented. There are the four cloud computing deployment models which are available to service customer.

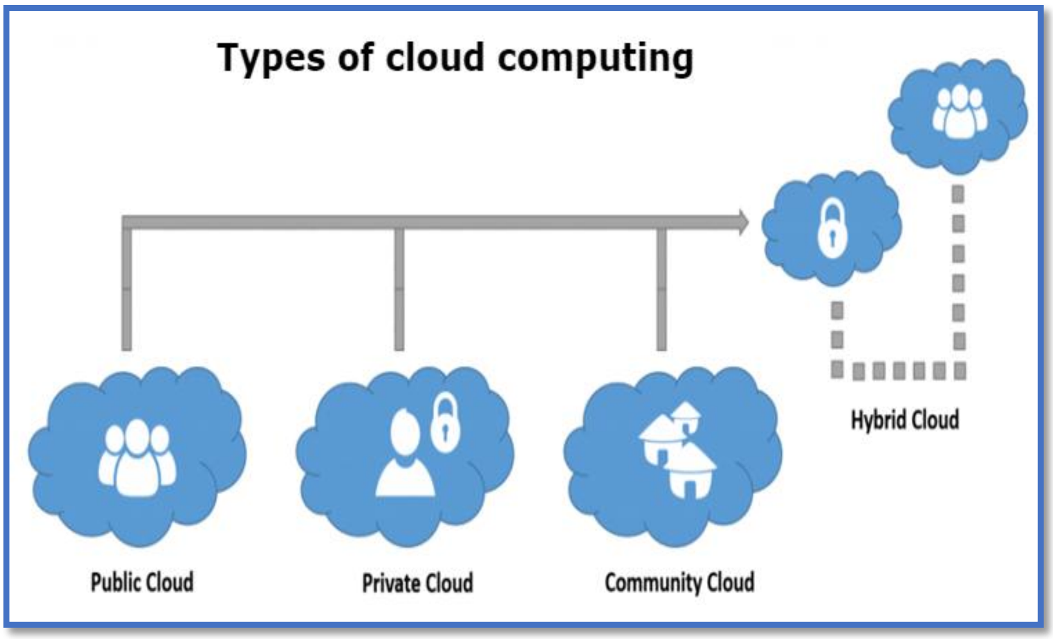


Figure I.4: Types of cloud computing [10]

1.5.1 Public Cloud

Definition

A public cloud is a platform that uses the standard cloud computing model to make resources, such as virtual machines, applications, or storage, available to users remotely. Public cloud services may be free or offered through a variety of subscription or on-demand pricing schemes, including a pay-per-usage model. [10]

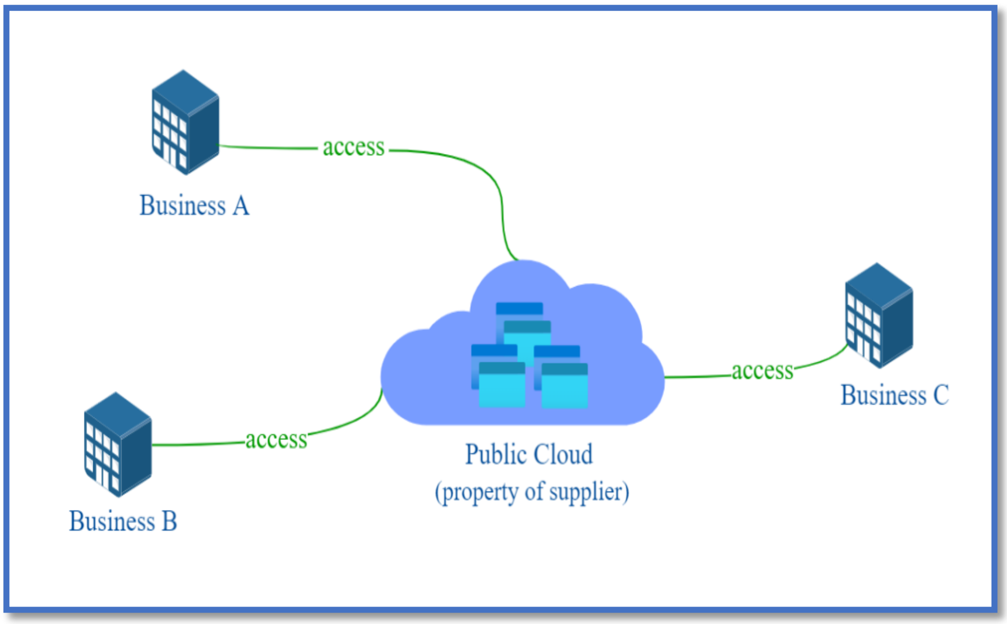


Figure I.5: Public cloud deployment model.[11]

Advantages [12]

- There is no need of establishing infrastructure for setting up a cloud.
- There is no need for maintaining the cloud.
- They are comparatively less costly than other cloud models.
- Strict SLAs are followed.
- There is no limit for the number of users.
- The public cloud is highly scalable.

Disadvantages [12]

- Security issue.
- Confidentiality and organizational autonomy are not possible.

1.5.2 Private Cloud

Definition

Private cloud operation is within an organization's internal enterprise data center. The main advantage is that it is easier to manage security, maintenance, and upgrades and also provides control over deployment and use. As compared to the public cloud where all resources and applications were managed by service providers, in the private cloud these services are pooled together and made available to users at the organizational level. These resources and applications are managed by the organization itself. As only the organization's users have access to private cloud the security is enhanced.[10]

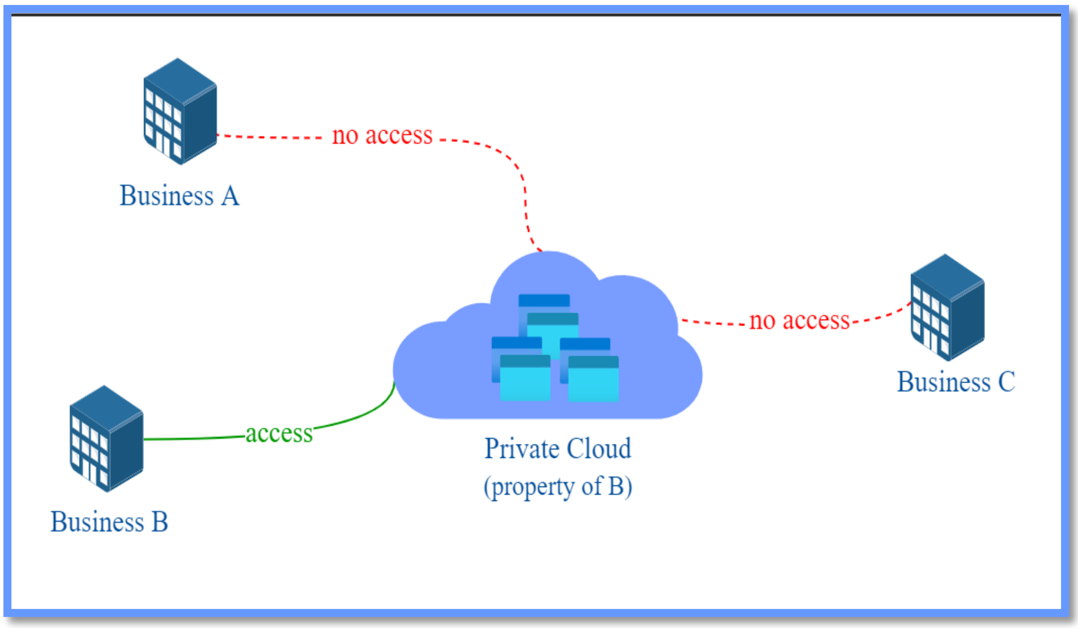


Figure I.6: Private cloud deployment model [11]

Advantages [12]

- The cloud is small and is easy to maintain.
- It provides a high level of security and privacy to the user.
- It is controlled by the organization.

Disadvantages [12]

- For the private cloud, budget is a constraint.
- The private clouds have loose SLAs.

1.5.3 Hybrid Cloud

Definition

This model is composed of both public and private cloud models, where the cloud computing environment is hosted and managed by third party but some dedicated resources are privately used only by an organization. In this model, a private cloud is linked to one or more external cloud services. It is a more secure way to control data and applications and allows the party to access the information over the internet. It enables the organization to serve its need in the private cloud and if some occasional need occur it ask the public cloud for intensive computing resources.[10]

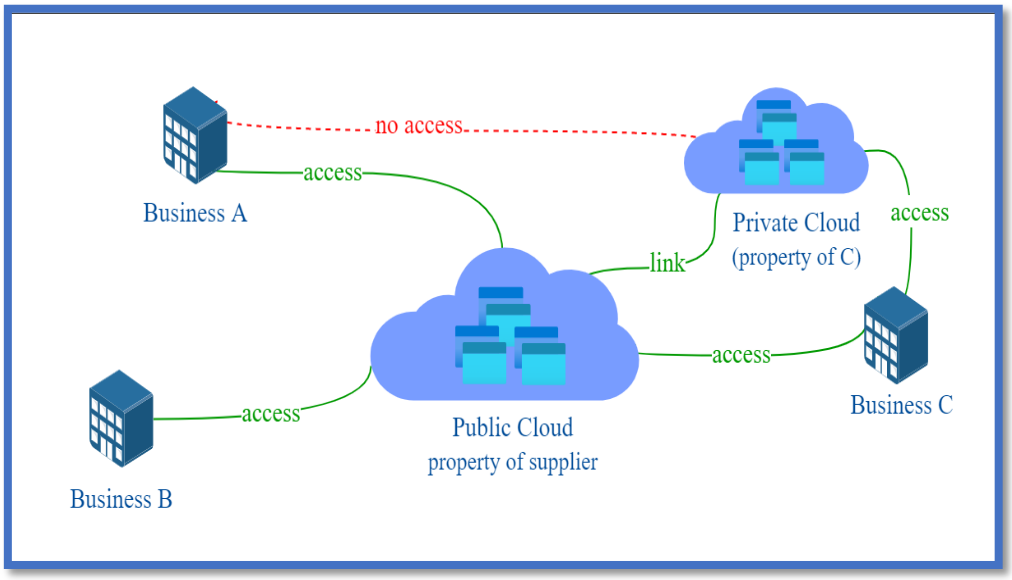


Figure I.7 : Hybrid cloud deployment model.[11]

Advantages [12]

- It gives the power of both the private and public clouds.
- It is highly scalable.
- It provides better security than the public cloud.

Disadvantages [12]

- The security features are not as good as the public cloud.
- Managing a hybrid cloud is complex.
- It has stringent SLAs.

1.5.4 Community Cloud

Definition

It allows the cloud computing environment which shared or managed by number of related organizations.

When many organizations combine construct and share the cloud infrastructure, their policies and requirements then such a model is called as a community cloud. The cloud infrastructure is hosted by a third-party provider or within one of the organizations in the community.[10]

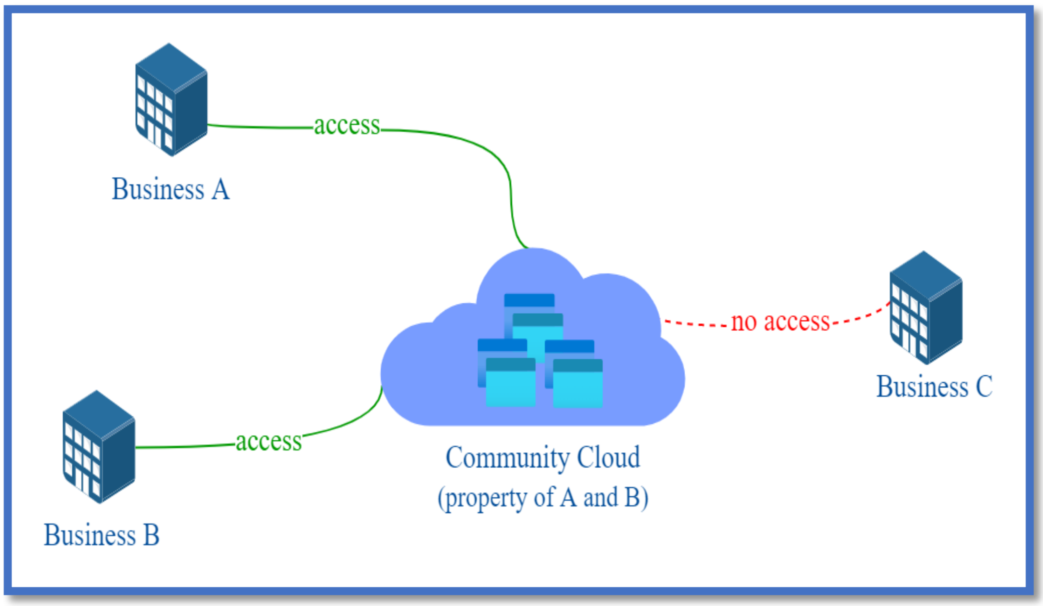


Figure I.8: Community cloud deployment model[11]

Advantages [12]

- It allows establishing a low-cost private cloud.
- It allows collaborative work on the cloud.
- It allows sharing of responsibilities among the organization.
- It has better security than the public cloud.

Disadvantages [12]

- Autonomy of an organization is lost.
- Security features are not as good as the private cloud.
- It is not suitable if there is no collaboration.

1.6 Cloud Services Model

Cloud computing can be divided into three categories for providing public services, as shown in the figure 1.9.

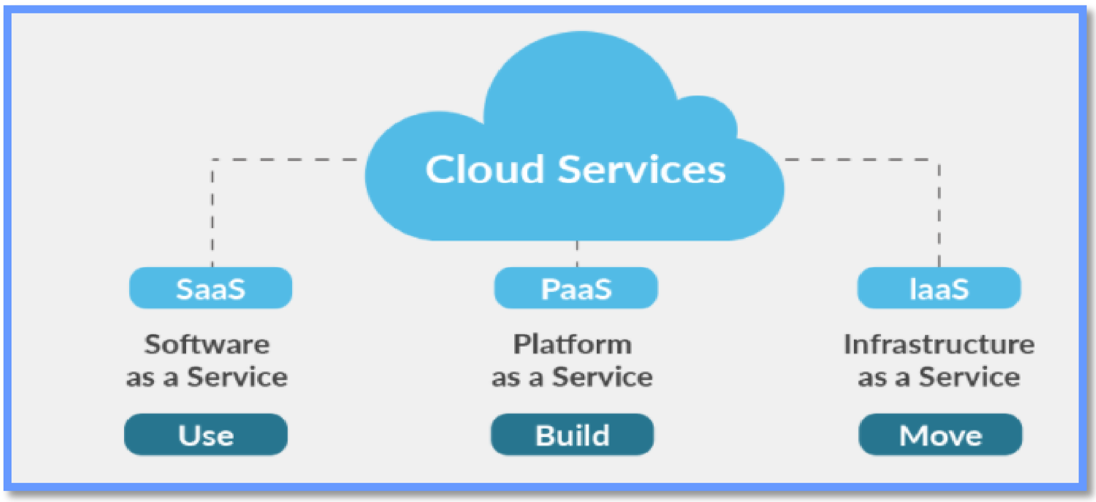


Figure I.9: The Different Services of Cloud Computing. [13]

1.6.1 IaaS (Infrastructure as a Service)

Definition

Cloud computing services of the IaaS type correspond to infrastructure resources offered on demand. These resources are computing, storage or network resources and can be either virtual or physical. The supplier manages the Calculation layers, Storage, Network and Virtualization. The user of IaaS resources is responsible for the management of all layers from and above the operating system. The user does not have neither the control, nor the management, nor the visibility of the underlying infrastructure.

Examples of IaaS providers: Amazon EC2, IBM Blue Cloud, Eucalyptus, Flexi Scale, Joyent, Rackspace. [14]

Benefits [14]

- Pay for What You Use: Fees are computed via usage-based metrics.
- Flexibility: Material resources are adjustable on demand and according to need.
- Increase Security: IaaS providers invest heavily in security technology and expertise

Drawbacks [15]

- Internet access essential with a satisfactory connection.
- Dependence on the supplier.
- Have the skills in-house because you manage the middleware and applications.

1.6.2 PaaS (Platform as a Service)

Definition

Cloud computing services of the PaaS type mainly correspond to development environments offered on demand. The user is only responsible for data and application layers. To do this, there is the use of libraries, languages, and tools offered by the supplier to structure its data and develop its apps. Once developed, the vendor must deploy and maintain the correct operation of the application by managing all the lower layers ranging from infrastructure to runtime environments. [16]

Typical examples of PaaS are: Google App Engine, Windows Azure, Engine Yard, Force.com, Heroku, MTurk. [16]

Characteristics [17]

- Multi-tenant architecture.
- Customizable /Programmable User Interface.
- Unlimited Database Customization.
- Robust Workflow engine/capabilities.
- Granular control over security/sharing (permissions model)
- Flexible “services-enabled” integration model.

Benefit [14]

- Flexibility: Allows employees to log in and work on applications from anywhere.
- Time and financial savings: Installation and maintenance are carried out by the service provider.
- Cost Effective: No need to purchase hardware or pay expenses during downtime.

Drawbacks [11]

- Limitation to one or two technologies (Python or Java).
- No control of virtual machines.
- Customers do not have control over VM or processing of the data. This leads to major security risks, as they are not aware of what is happening with their data.

1.6.3 SaaS (Software as a Service)

Definition

Among the three models offered by the Cloud, software as a service (SaaS) represents the most attractive service delivery model. In SaaS, and unlike traditional single license models,

SaaS allows consumers to pay a subscription for on-demand services. Among the examples of business applications offered by this layer include: CRM, collaborative tools, messaging, Business Intelligence, ERP, etc.[14]

Characteristics [17]

- Multi-tenancy model
- Automated provisioning
- Single Sign On
- Subscription based billing
- High availability
- Elastic Infrastructure
- Data Security
- Application Security
- Rate limiting/QoS

Benefits [14]

- Scalability: Easily scale a solution to accommodate changing needs.
- Increase Security: SaaS providers invest heavily in security technology and expertise.
- Pay for What You Use: Fees are computed via usage-based metrics.
- Data migration.

Drawbacks

- Customers do not have control over VM or processing of the data. This leads to major security risks, as they are not aware of what is happening with their data.[11]
- Connectivity requirements. Since SaaS software is web-hosted, you can't use these applications without an Internet connection.[18]

1.7 Conclusion

The first chapter aims to provide a comprehensive description of cloud paradigm, its types, and its model of services. In the end the SaaS service model represents the most attractive and finally and promising marketing model in the cloud.

CHAPTER TWO

2.1 Introduction

Multi-tenancy architecture represents a promising research avenue in cloud computing, which still arouses a great deal of interest. This interest comes mainly from its ability to share (according to levels of maturity) hardware/software resources with several "consumer" tenants of the Cloud, i.e., to increase the profits of current Cloud applications.

This chapter introduces multi-tenancy concepts, ideas, approach and briefly explains some works that express the steps of transition from traditional application to multi-tenant SaaS.

2.2 Single tenancy and Multi-Tenancy overview

With the advent of IaaS, PaaS and SaaS, multi-tenancy becomes more and more important, as these new delivery models allow to share different parts of an application (infrastructure, platform or software) between different tenants.

This section's purpose is to discuss and define different styles of how Customers may be served through the instance of an Application Component.

A tenant is a user (or a group of users) sharing the same view on an application they use. This view includes the data they access, the configuration, the user management, particular functionality and related nonfunctional properties. Usually the groups are members of different legal entities. This comes with restrictions (e.g. data security and privacy).[19]

Traditional software model is a model of single-tenant isolation, where an organization purchases and installs a software application on their servers. Application on the server is only available for users of the organization's group. This model is typically a multi-user approach. Unfortunately, the increasing number of users require additional basic package costs (servers, hardware, supply network, data backup and IT support) and burden the whole organization.

SaaS model using multi-tenancy architecture where the hardware infrastructure is shared among many different customers, but logically unique for each customer.[20] In a multi-user application, all users share the same application with limited configuration options. In contrast, multitenancy allows each tenants to customize some parts of the application such as color of the user interface or business rules except the application's code. Even if the tenants share the same building blocks in their configurations, but the appearance or workflow may be completely different among each other.[21]

The multi-tenancy architecture pattern has established itself in the past decades as the de-facto standard for Software-as-a-Service applications. The definition of multi-tenancy has been debated extensively and various definitions have been proposed by researchers.[22]

The multi-tenancy is considered as an essential property of cloud services. Indeed, it is an architecture in which a single instance of a service serves multiple users/organizations. Each one is identified as a tenant. This latter can be a single user or a whole company consisting of several distinct users. The multi-tenancy attempts to replace many small services instances with one or few great instances in order to maximize the utilization of infrastructures. In particular, a tenant should trust cloud services providers based on isolation between private data and virtual machines regardless of his concerns or goals.[23] Therefore, the multi-tenancy is based on two basic modes which are[24]:

• Native multi-tenancy: it means that all tenants are supported by a single shared service instance over various hosting resources. Indeed, the same instance denotes that the same service using the same code on the same infrastructure is used by different tenants.

• Multiple instances: it denotes that each tenant has his dedicated service instance over a shared hardware, operating system or a middleware server in a hosting environment.

A service instance should require a tenant specific behavior for a service.

In this context, the multi-tenancy awareness aims to deal with different multi-tenancy modes in order to provide a transparent interaction between each tenant and the business processes. Otherwise, the tenants do not feel of using a dedicated application and doubt of the security of their data and consequently they may move to other service provider.

According to the NIST definition, "effective resource pooling and sharing represents a critical factor for the success of cloud computing. Effective resource pooling and sharing increases the average resource utilization, by what total costs are reduced. Moreover, it reduces the number of resources which are required to fulfill the same demands as without resource

pooling and sharing. A lower number of resources reduces management efforts, energy consumption, compute center floor space, cooling efforts and so forth".[20]

Multi-tenancy is a web architecture in which a single instance of a soft-ware application serves multiple customers/Tenants. Tenants may be given the ability to customize some parts of the application, such as color of the user interface (UI) or business roles but they cannot customize the application's code.[21]

In order to conclude the discussion about Single-Tenancy and Multi-Tenancy it is necessary to address one more aspect. All Application Components are built and deployed on a stack of infrastructure that they utilize. This infrastructure stack may vary from Application Component to Application Component.

For the Multi-Tenancy model, where instances are shared, it is quite obvious that the underlying infrastructure must be shared as well. For Single-Tenancy, however, this statement may not be made that easy. It may be possible that Single-Tenancy only applies to the instances Application Component, but the underlying infrastructure or only parts of it are shared by different Customers. It is up to the Operator of an application to determine for which parts of the infrastructure stack Single-Tenancy shall be offered.[25]

2.3 Benefits and challenges of multitenant architecture

Many benefits of multitenancy can be seen in the widespread use of cloud computing, whether for the seller or the buyer of the cloud service.[20] [21] [26] [27]

For the vendor

- Reduce software development cost reduction, since all tenants are shared the same software and there is no need of developing different applications for different customers (tenants).
- Database backup and archiving strategies can be implemented for single database.
- Ease of updating and maintenance.

For the Customers

- Efficient use of hardware resources.
- Low cost: pay just what you use.

Despite the benefits of multitenancy, some use cases, including employing your own data center or a private cloud, are better suited for single-tenant computer systems.
Like two sides of the same coin, multi-tenancy poses several challenges and difficulties as well. [22][26] They are:

For the vendor

- customizable: less easily able to customize the application to customers. It is still possible to lock some features in the code but, it may lead to complexity related to development.

For the Customers

- Performance: Amount of resources distributed to each tenant may result in inefficient utilization of resources.
- Security: Any security breach has a massive effect because it will affect all users of the system.

2.4 Realization Approaches of Multi-Tenancy

With the increased pairing of the multi-tenant principle in SaaS applications provided by the Cloud, managing this type of application is increasingly becoming a vital subject.

In two main approaches for managing multi-tenancy in the Cloud were restored. The first is named Functional, and the second is named Database. The author states that Oracle is looking at multitenancy primarily from a database perspective, while Microsoft is looking at multitenancy more from a functional perspective.[28] Figure II.1 illustrates a classification of SaaS multi-tenant approaches.

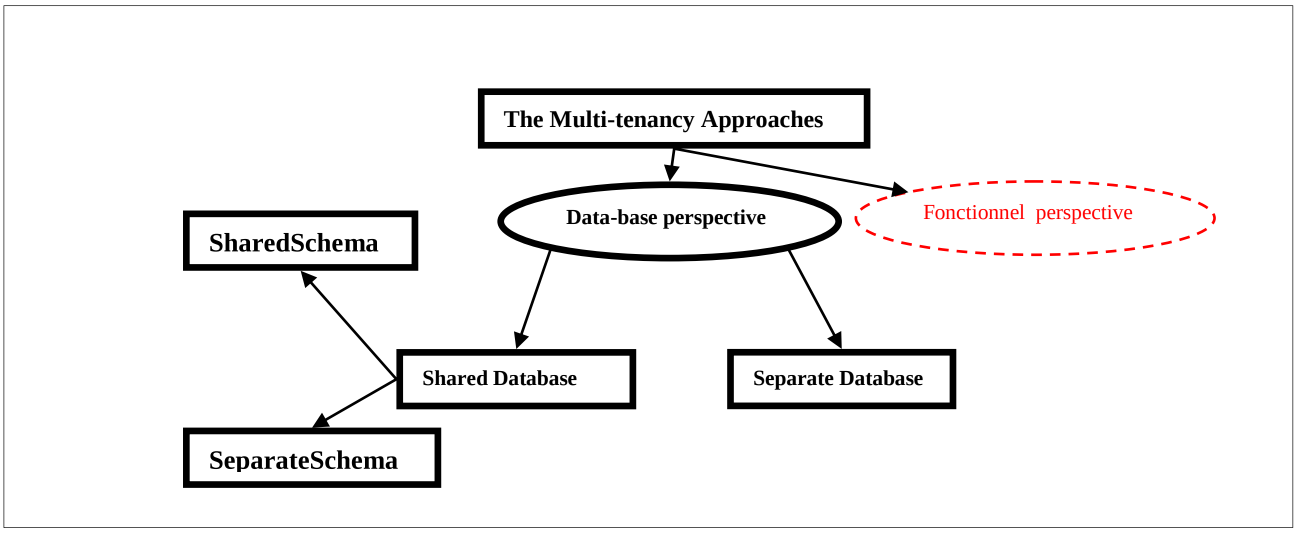


Figure II.1: A classification of SaaS multi-tenant approaches.

The study presented in this work focuses on the database perspective in prospecting for developing/managing the multi-tenant SaaS Cloud. This perspective has attracted the attention of many researchers in this effervescent context. The multi-tenant database perspective is the vision that focuses on the functionality of the database that it would provide to multiple tenants to create, store, and access their databases over the Internet. In this multi-tenant vision, the database must be highly configurable and secure to meet the expectations of tenants and their different business requirements. It is noteworthy that the Functional Perspective (Functionalism) is different from the Data-Oriented Perspective. The functional perspective emphasizes managing and controlling business processes that produce services.[29]

we adopt in this work the distinction found in [21] when studying multitenancy approaches with a database view. A first step towards doing this, is to report on how Customers' data may be separated. gives an overview by introducing three distinct approaches for creating data architectures to realize Multi-Tenancy. These are the following.

2.4.1 Separate Databases

The first model is actually not belonging to multi tenancy, but it is often refered as a form of multi-tenancy in a simple virtualization cloud with only shared hardware.

Separated databases are the simplest and easiest approach for data isolation. Tenants are shared in the same application, computing resources and hardware on the server, but each tenant maintains its own database.This why each tenant's data remains isolated from others' data and accidental or malicious access to data that belongs to other tenants is prevented (Figure II.2). The gain of this data model is that it is easy to extend to specific Customer needs, restoring backups for a specific Tenant may be done very easily, and it offers added security. Cost, hardware and maintenance, however, are higher than for the other alternatives since the number of Tenants that can be served by a database server is limited to the number of databases it supports.[21]

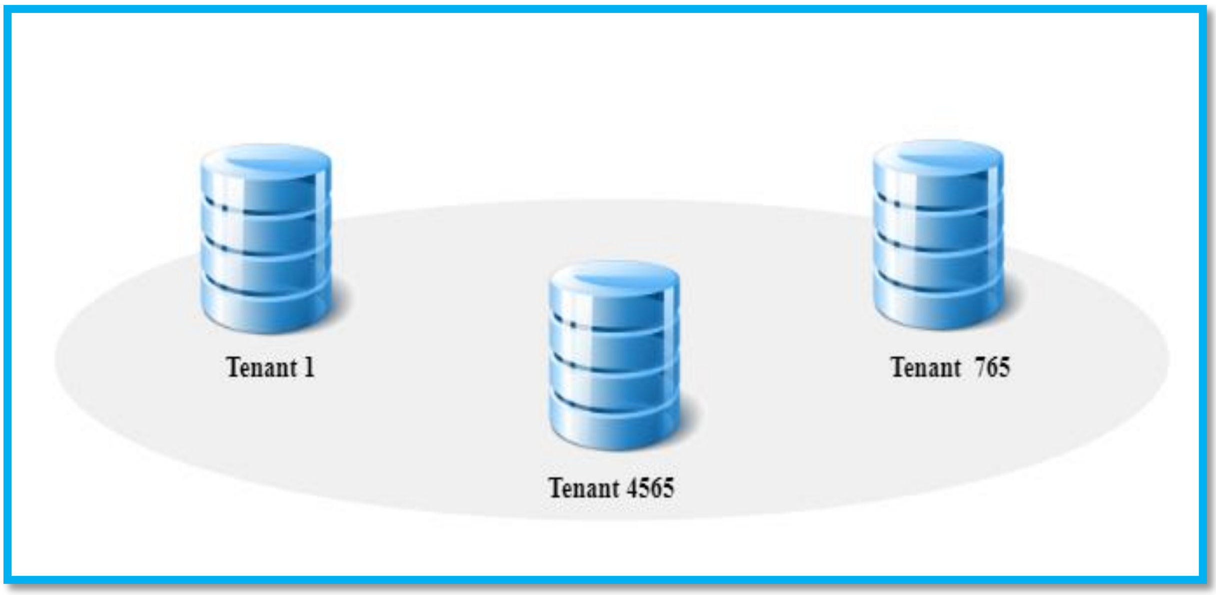


Figure II.2: Separate Database Approach.[21]

Advantages[30][31]

- Can be customized for each customer and vendor can apply new releases or customization as per customer requirement.
- Easy to restore tenants data from backup in case of failure.

Disadvantages[30]

- Expensive in terms of management, integration, updates, customer support.
- Every customer has unique configuration and vendor need to manage them all.

2.4.2 Shared Database, Separate Schemas:

The next approach is to store multiple Tenants' data in the same database but in separate schemas. Once a Tenant first subscribes to use a given application, it is necessary to create a set of database tables that are associated to this Tenant. Again, it is possible to restrict access to those tables only to those Users that belong to the authorized Tenant. This approach offers a moderate degree of logical data isolation for security-conscious Tenants, and supports a larger number of Tenants per database server. The most significant drawback of this approach is that Tenants' data is harder to backup and restore since it is not possible to restore a backup of an entire database. This would overwrite other Tenants' data. Overall this approach will result in lower cost for the Operator at the first approach.[32]

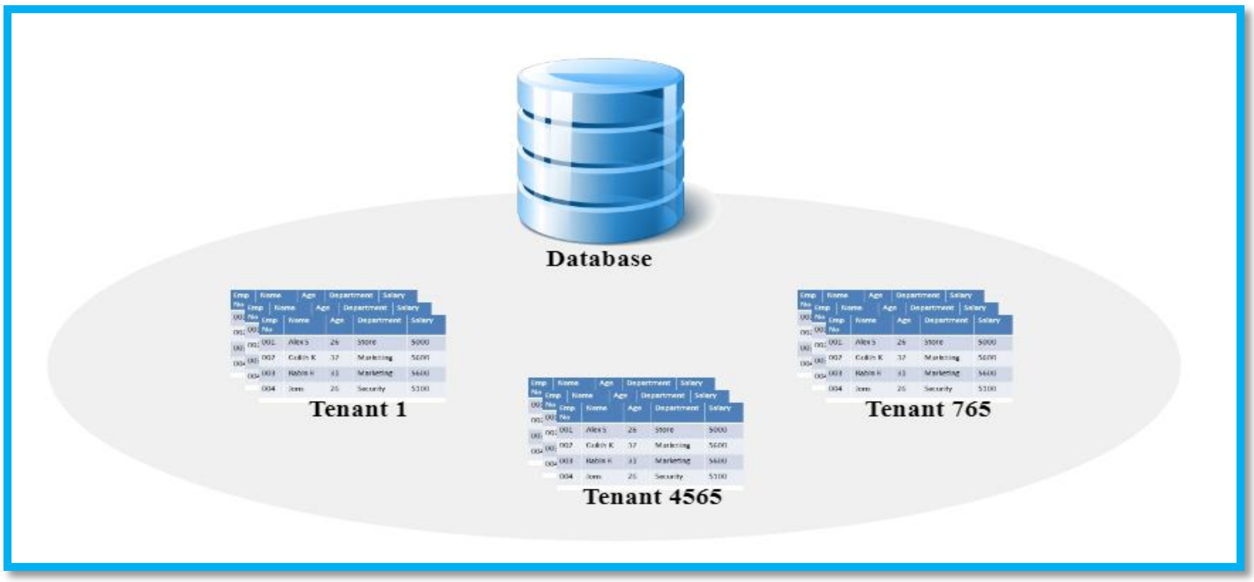


Figure II.3: Shared Database, Separate Schemas Approach[33]

Advantages[30][33]

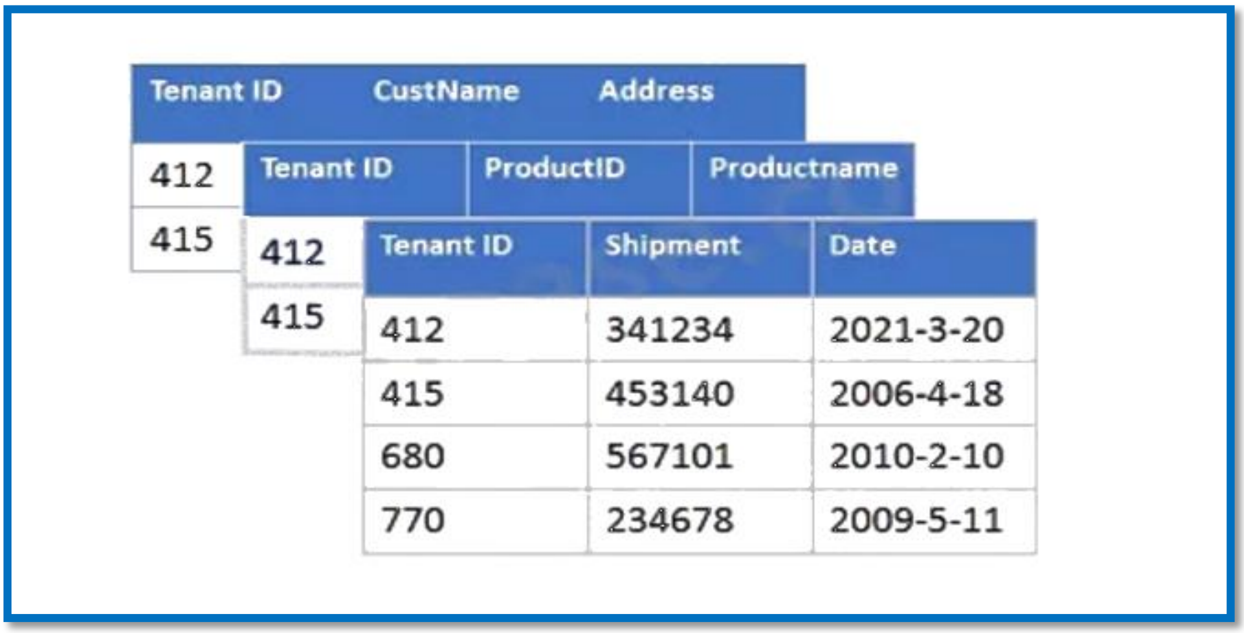
- Each database can support more number of tenants.
- Low cost of maintenance.
- Less memory consumption.

Disadvantages[33]

- Difficulty data recovery, in case of failure because the database will involve other tenants of the data.

2.4.3 Shared Database, Shared Schema

True multi-tenancy is shown in the third model, where users fully share the application and databases (the highest efficiency). This model requires a more onerous process especially for changing the databases scheme in adding tenant identification for each table and view, and rewriting any SQL access to add filter criteria for tenant .This one is an exceedingly preferred approach and called as pure-multitenancy approach, in this approach each tenant data is stored in the same database and same schema, the separation of tenant is done by assigning **Tenant_id** to columns in tables that the tenant possesses (Figure II.4). [32][33].



Tenant ID	CustName	Address
412		
415		

Tenant ID	ProductID	Productname
412		
415		

Tenant ID	Shipment	Date
412	341234	2021-3-20
415	453140	2006-4-18
680	567101	2010-2-10
770	234678	2009-5-11

Figure II.4: Shared Database, Shared Schema Approach.[33]

Advantages[30][33][31]

- Updates apply automatically to every tenant at the same time.
- Allows each database to support the largest number of tenants.
- Among the three approaches this one is the less costly.

Disadvantages[33]

- Minimum security.
- The lowest isolation level.
- The most difficult to data backup and restore.

Regarding the isolation of the database layer, the more isolated the data is among different tenants, the easier to customize, but the more expensive are hardware and maintenance (Figure II.5). The multi-tenancy is pure when using low degrees of isolation.[34]

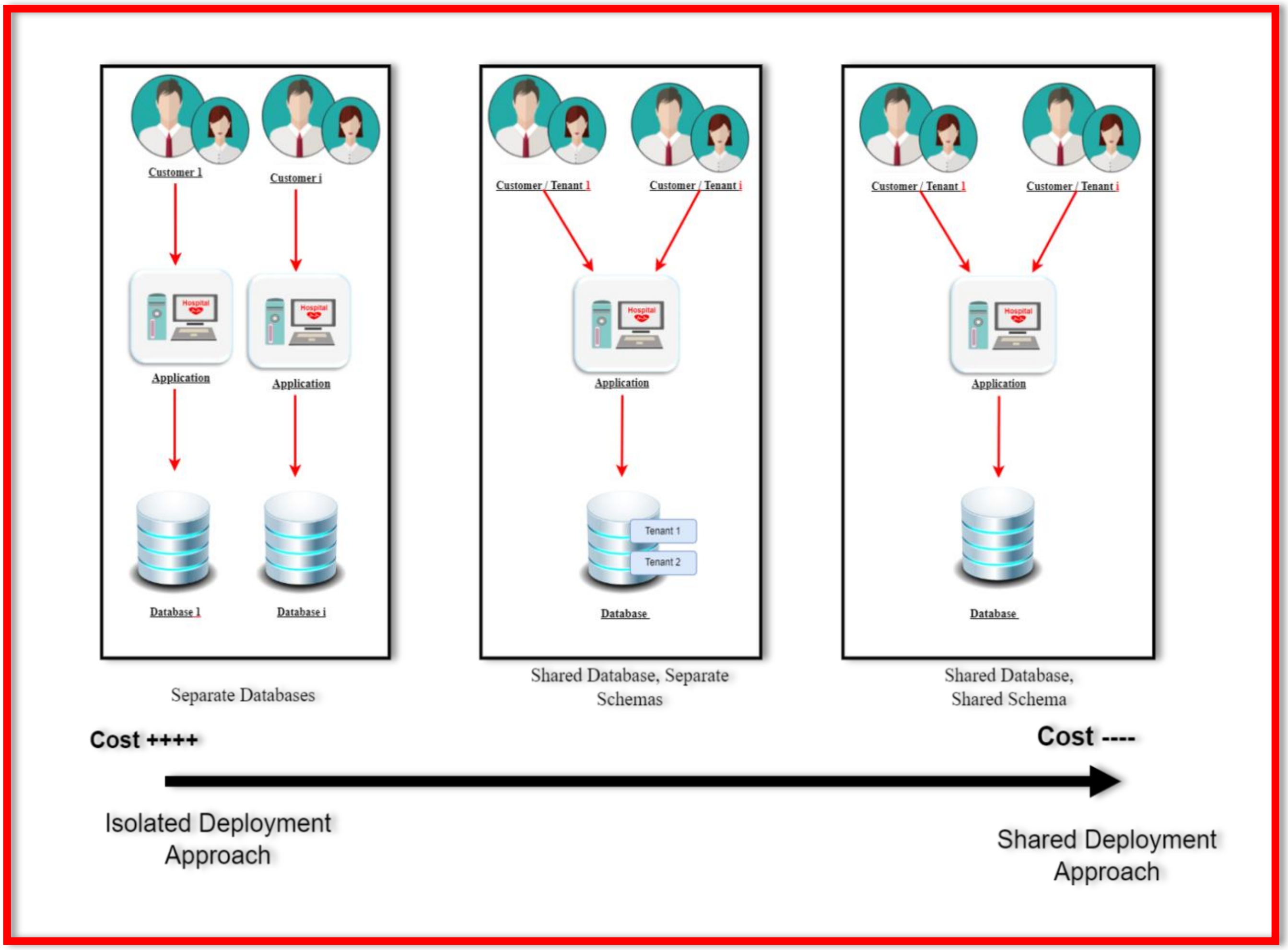


Figure II.5: The implication of the approach on the cost of the application [21]

2.5 Migration Into SaaS Multi-Tenant (Related Work)

Previous research has looked at a framework for guiding the conversion of conventional web applications to the multi-tenant SaaS paradigm. In this thesis, we use the two Cloud SaaS publications that have received the greatest citations.

Saleh et.al. [35] proposed a migration framework by incorporating the element of customization of user interface and workflow, business logic and database configuration.

Bezemer et.al. [36] conducted application migration patterns that takes into account hardware sharing, high degree of configurability and shared database instance. There are three components involved in those frameworks :

1. Authentication module to map end-users credentials to tenants,
2. Configuration module to handle tenant-settings, and
3. Database module to adapt insert, modify and query tenant oriented data.

2.6 Conclusion

This chapter discusses multi-tenancy as a general architectural paradigm for creating efficient SaaS applications that use many of the advanced capabilities offered by cloud computing. Additionally, the various SaaS maturity level are described in relation to multi-tenancy, highlighting how crucial multi-tenancy is to creating quality SaaS applications.

Finally, the economic impact of multi-tenancy is discussed as driving force behind the choice for it to be implemented. The objective of next chapter to do the maximum to implement a multi-tenant SaaS application using the selected approach.

CHAPTER THREE

CHAPTER III: Design and Realization

3.1 Introduction

Currently, the realization and implementation of the SaaS service model and multi-tenant architecture in today's applications remains difficult to implement both in design and programming.

In order to circumvent this problem, many frameworks and their libraries provide a more convenient way to implement the SaaS service model with subscription and multitenancy.

In this last chapter, we will describe the targeted application, the hardware and software tools and libraries used in our implementation. Next, we will present the approach adopted when migrating from the traditional application to the multi-tenancy SaaS application. and the different interfaces of our resulting SaaS application named MedCare.

3.2 Implementation and production tools

3.2.1 Hardware environment

Our project was carried out using an "Acer Aspire ES1-571" computer, the configuration of which is described in the following table:

Processor	i3-5005U (2.0 Ghz, 3MB L3 Cache)
Hard Drive	500 GB
Random Access Memory	4 GB DDR3 L
Operating System	Windows 10
Graphic Card	Intel HD graphique 5500

Table III.1: Computer characteristics

3.2.2 Software environment

There are a variety of programming languages and frameworks dedicated to developing applications. Therefore, choosing the technology to use is sometimes a complex matter.

Especially since this choice is very expensive in terms of time and money, this is what we were exposed to because we were unable to provide a Visa card in order to use a paid platform such as: azure, Google cloud, SAP...

But after several researches, we got to know the free Laravel framework that helped us develop our application.

A. Laravel

Created in 2011 by Taylor Otwell, Laravel has become the most popular free and open source PHP framework in the world. It is used in web application development while following the MVC architecture and based on Symfony. As each version of laravel is documented in detail, one can learn it easily and count on a continuous update of its knowledge base.

Laravel is designed for rapid application development. The framework has a well-built templating engine that enables a wide variety of common tasks such as authentication, caching, user management, and RESTful routing to make it easy for software developers.we used Laravel version 8.83.11[37]

Figure III.1 : Laravel.[38]

Why Laravel Framework? [39]

According to its founder Taylor Otwell, "Laravel is the most powerful of the PHP ecosystem simply because it includes the features needed to build modern web applications". It is therefore an elegant and clean framework, with an elegant syntax for creating large applications.

Among the advantages of this framework, we can cite:

- Uses the latest PHP features,
- Excellent documentation: You will find good detailed explanations of coding style, methods and classes,
- Integration with messaging services
- Has a built-in command-line tool called Artisan: The tool called Artisan helps them to create skeleton code and manage the database system successfully. Artisan helps generate the basic MVC files and manage the assets including their respective configurations.
- Supports popular cache backends Laravel supports caching backends like bootstrap and Redis out of the box. You can also configure multiple cache configurations.

- MVC architecture support : If you search on Google you will find that Laravel follows a Model-View-Controller architecture. And that's what makes Laravel a "great" framework to use for your web application development. It improves performance, provides clarity and allows for better documentation.

B. PHP

Hypertext Pre-processor" a recursive acronym. PHP is a scripting language web developers use to create dynamic websites , we used php version 7.4.27[40]

C. MySQL

Installation of the apache web server with mysql: we have chosen the most popular PHP development environment XAMPP is a completely free and easy to install Apache distribution containing MySQL, PHP and Perl [we used version 7.4.27].[41]

D. Studio Code

Visual Studio Code is an open source, free and cross-platform (Windows, Mac and Linux) text editor, developed by Microsoft. Mainly designed for application development with JavaScript, TypeScript and Node.js, the editor can be adapted to other types of languages thanks to a well-supplied extension system.[37]

E. Mailtrap

Mailtrap is a service for the safe testing of emails sent from the development and staging environments. Mail trap catches your emails in a virtual inbox so that you can test and optimize your email campaigns before sending them to real users.[42]

3.3 App Description

Given the lack of time and patient congestion with the many pressures on doctors in healthcare facilities, our traditional application (non-SaaS and non-multi-tenant) ***MedCare*** [43] offers solutions to some of these problems, especially the management of appointment.
MedCare is an application that can be used in health establishments by the administrator of the establishment and the doctor, each actor having in particular his own interface and his own functions; this traditional application was created using several tools including Laravel which were chosen mainly on its free and open source version.
The major objective expected behind this modest work is to migrate this classic application to a modern SaaS Multitenant type application.
The following paragraphs show the process followed, the approach and the methodology adopted and the modeling designed and finally the resulting ***MedCareSaaS*** [44] application and its interfaces.

3.4 Development process

The development process of our application is illustrated by the following figure. Our primary objective is to switch to the SaaS model then integrate one of the multitenancy libraries offered by the LARAVEL framework to really refine the multitenancy whether it is uni-database or multi-database.

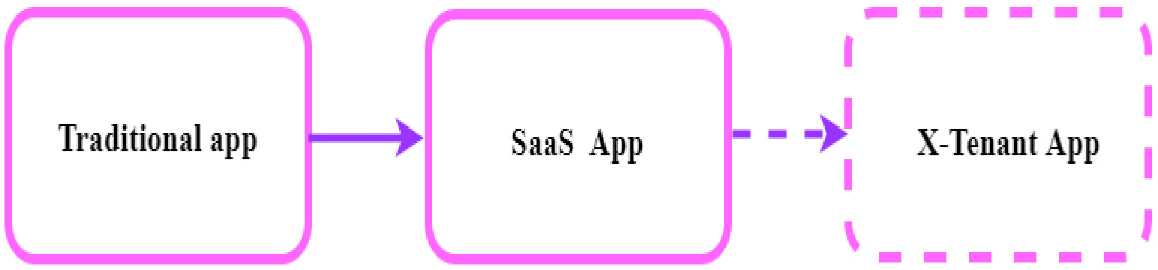


Figure III.2: Application process

3.5 The approach adopted

In the work [45] Nugraheni proposes an approach for the migration process to multi-tenant SaaS of multi-database category, detailing as follows:

- A new module should be created to handle registration procedures for tenants and their users.
- Building the database system to make it possible to create a database for each tenant of a registered group.
- Additional customization layer as users enter the system should be added: An authentication model will be used to map users into tenants and their holdings (including tenant_id as well as other pertinent information). Authentication module is used as a mechanism for identifying each tenant. Every tenant who successfully logs in to the application would have a tenant-id which can be used for configuration and customization of the application.
- Providing each tenant with an initiation page to configure and customize the interfaces components, such as logo, color, appearance, etc that will be recorded and forwarded to the application server.
- Configuration data on the database will be forwarded to the database servers for query transformation.
- Developing a new dashboard for system admin to monitor the performance and status of each tenant application.

This last work [45] will be personalized and adopted by our approach with a superb Udemy training learn[46].The main steps of our inspired approach designed specifically for Laravel Framework is detailed as follows:

Step 0

- Install and Configure the "Laravel" framework, and the associated database and libraries (eg Laravel UI.)
- Create the primary models of the MedCare SaaS system (User, patient, doctor,..)

Step 1

- Install and configure the Paddle-Cachier package by command:

composer -require laravel/Cachier paddle. At this stage three tables are introduced, customer(tenant) ,subscription , and receipt.

Don't forget to Refresh the database to integrate these new tables.

- Introduce and Configure the subscription plans with a consideration of its characteristics for each plan (exp. name of the plan (basic, pro), characteristic (number of patients, number of subscriptions), the price of each plan...).
- Extract attributes for each plan, such as name, plan ID, trial version interval, price from Sandbox-vendors-paddles website. then reintegrate these attributes into the plan model.
- Add the PlanFeature model (name, code, value).

Step 2

- Create views carefully for all the interfaces used by the tenants, and the doctors, by relying on the templates offered by the different libraries already installed, for example bootstrap ..etc.

Step 3

- Create all controllers to verify the different models developed in the previous phases.

3.6 General architecture of the System and modeling

In this part we want to give a general overview of the overall architecture of the MedCare SaaS system. as well as UML modelling in terms of use case diagram.

3.6.1 General architecture of the System

As shown in figure III.3, the service architecture developed in MedCare SaaS is based on the famous MVC 'Model view controller' architecture, which is as follows:

- The Tenant$_i$ represents the tenant of the service.
- The browser : represents the tenant's means of interaction with the system provider.
- The controllers: a file that contains all the classes that control the models and the views (we can cite the Tenant$_i$ controller as an example).
- The models: a set of files (classes) that constitute the interaction layer with the application database. (example: holding model, patient, doctor).
- The views: all the folders and classes that represent the templates of all the programming interfaces in CSS/HTML/JS would offer by the bootstrap for example.

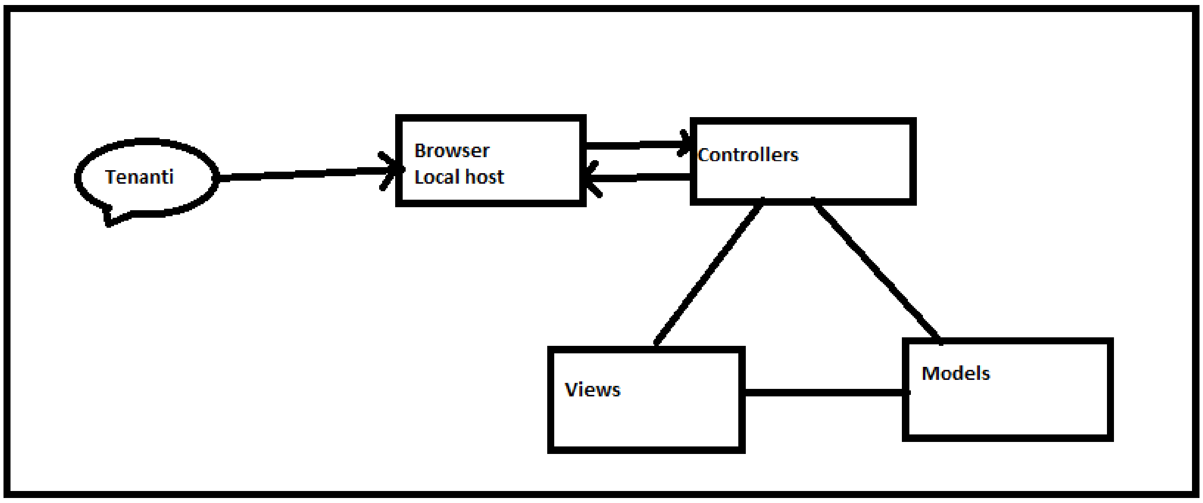


Figure III.3: MedCare SaaS Architecture

3.6.2 Modeling of the System

The Unified Modeling Language UML is a graphical modelling language based on pictograms designed to provide a standardized method for visualizing the design of a system. It is commonly used in software development and object-oriented design.

It thus includes several types of diagrams representing as many distinct views for represent particular concepts of the information system.[47]

We use this thesis the use case diagram to show the different actors as well as the different functional needs that corresponded to it. it makes it possible to identify the possibilities of interaction between the system and the actors (participants outside the system), ie all the functionalities that the system must provide.

Use case Diagram

The MedCare SaaS system is designed for several actors with their respective roles.

The actors in MedCare SaaS are:

i) MedCare admin, who acts as a verifier for registered tenant Admin;according to its subscription plan through the Paddle-Cachier gateway

ii) tenant Admin can register patient and doctors as well as manages and monitors appointments and doctors ; after of course the creation of an account and the subscription according to a well-defined plan (pro or basic), use of course on its dashboard.

iii) Doctor can manage on the one hand, edit and modify his profile, and see the list of doctors, and on the other hand manage the appointments by adding, editing and consulting.
and finally iv) The patient, get tenant services(appointment) after queuing.

The actors and their relationships in the MedCare SaaS system are explained using a use case diagram as shown in Figure III.4.

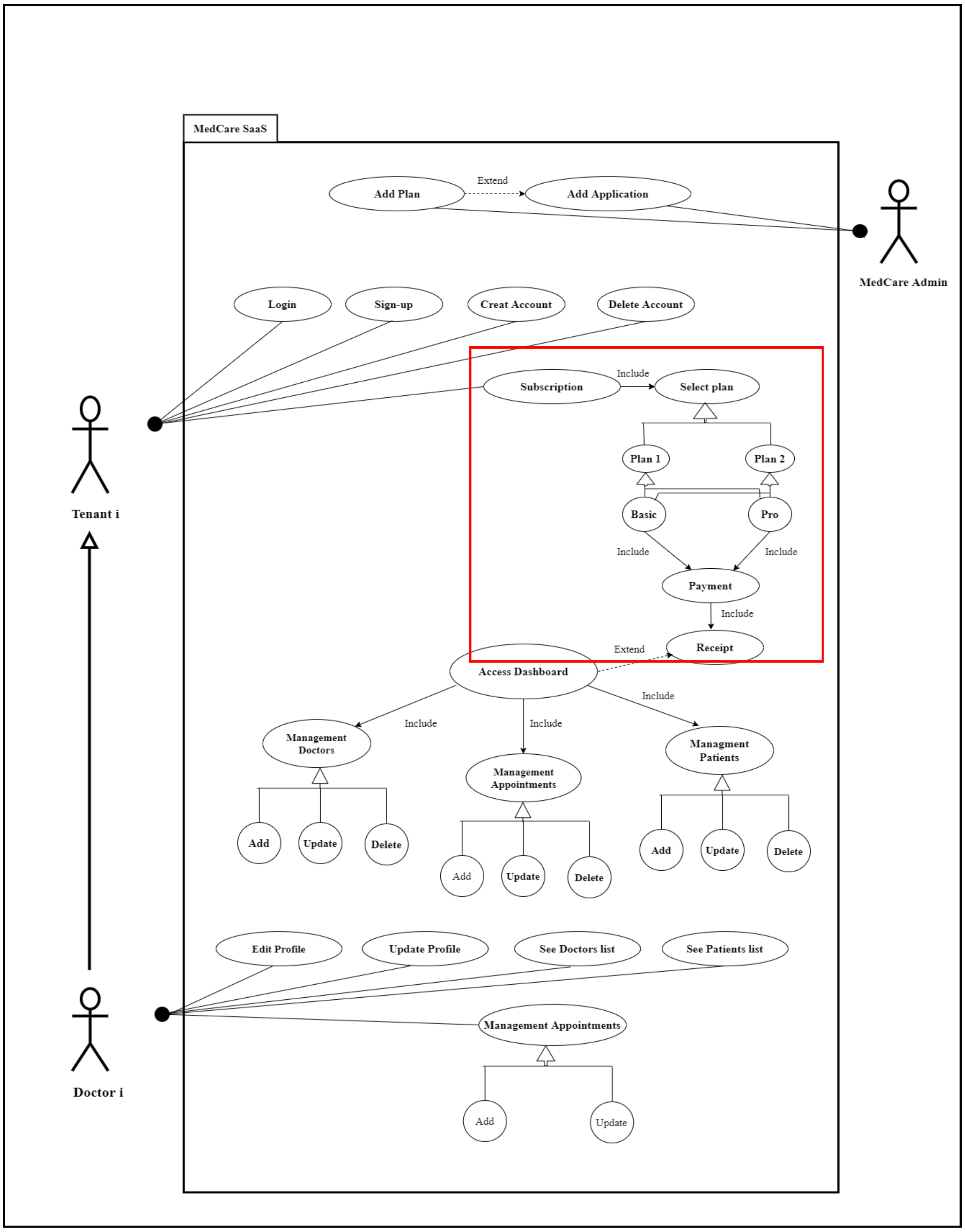


Figure III.4: MedCare SaaS use case diagram.

3.7 Implementation and interfaces of the application

The first thing that appears for the user (admin or doctor) is this page and through it he can register in the application by pressing the button login , As shown in the Figure III.5 .

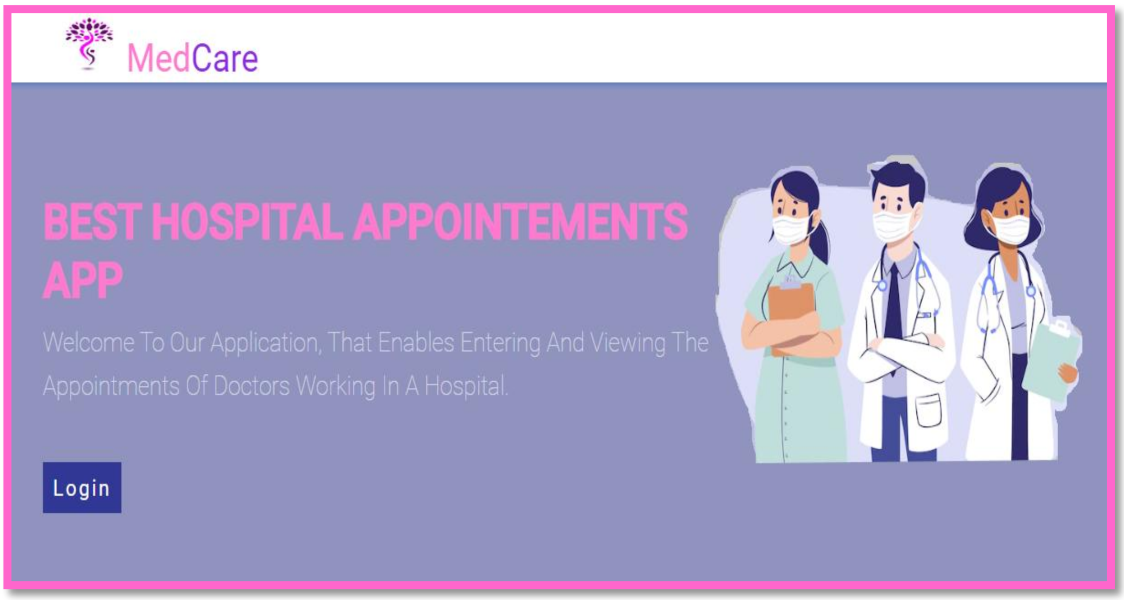

MedCare
BEST HOSPITAL APPOINTEMENTS APP
Welcome To Our Application, That Enables Entering And Viewing The Appointments Of Doctors Working In A Hospital.
Login

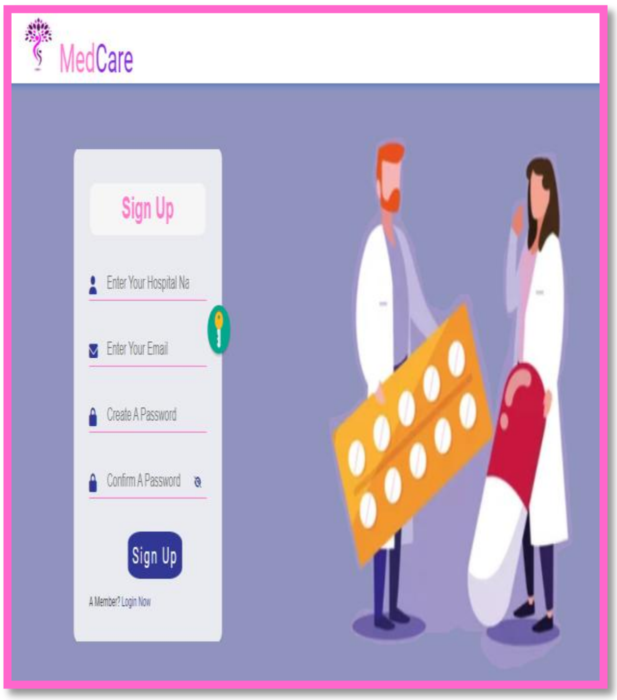
Figure III.5: MedCare interface.

After the main interface appears and pressing the login button, if the user is the admin , he must first register by entering the hospital name, email and password as shown in Figure III.5. But if the admin is registered before , he has to enter the email and password only as shown in Figure III.6.

If the user is the doctor, he has to enter the email and password only because he is registered before by the admin as shown in Figure III.6.

If the information entered by the user is incorrect, the system will return the authentication page to it with an error message.

If the user forgets the password, he can retrieve it by pressing the Forgot Password button.



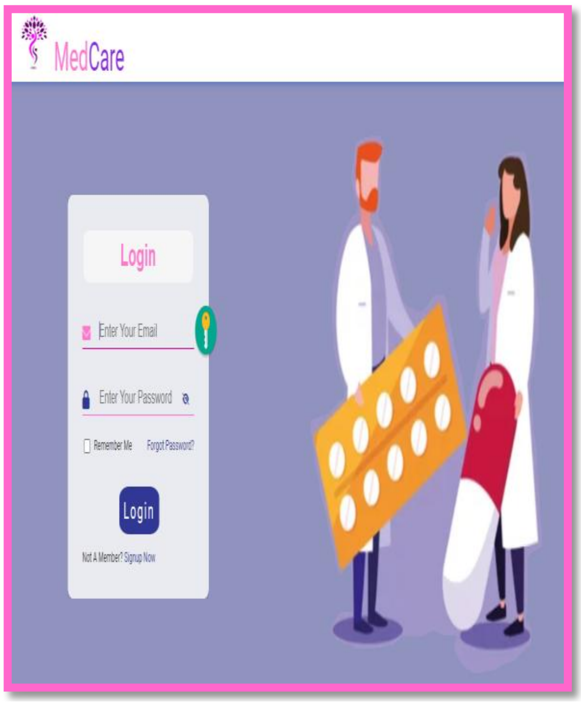


Figure III.6: Sign Up interface. **Figure III.7:** Login interface.

3.7.1 Admin interface

Account

After the admin enters his account, he updates his information as shown in Figure III.8.

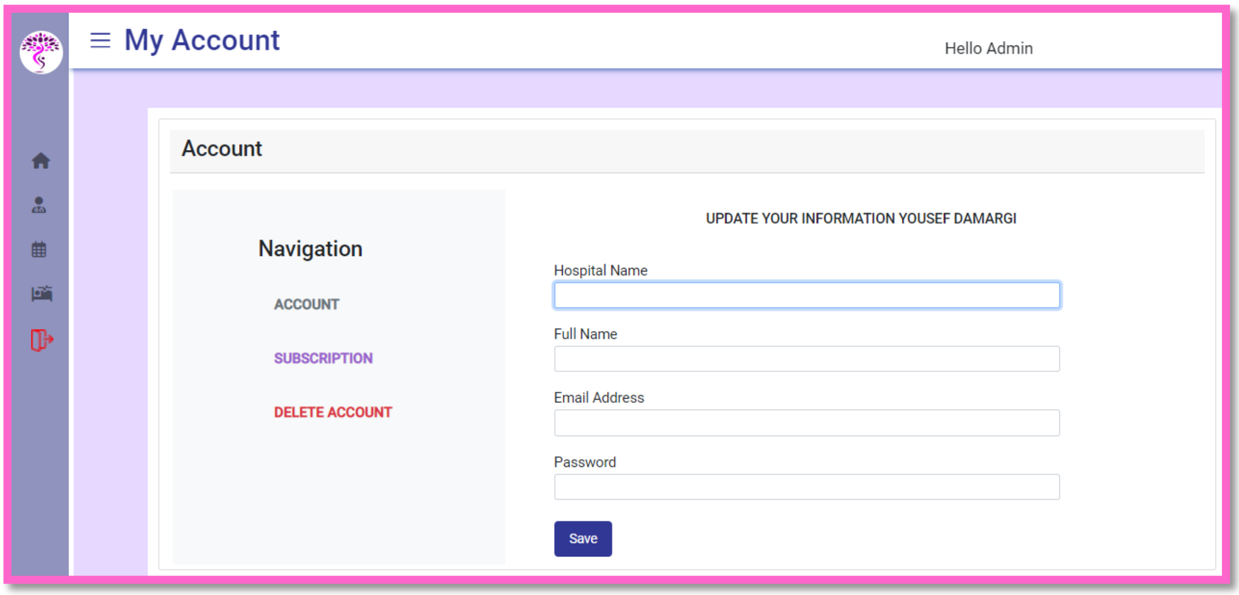
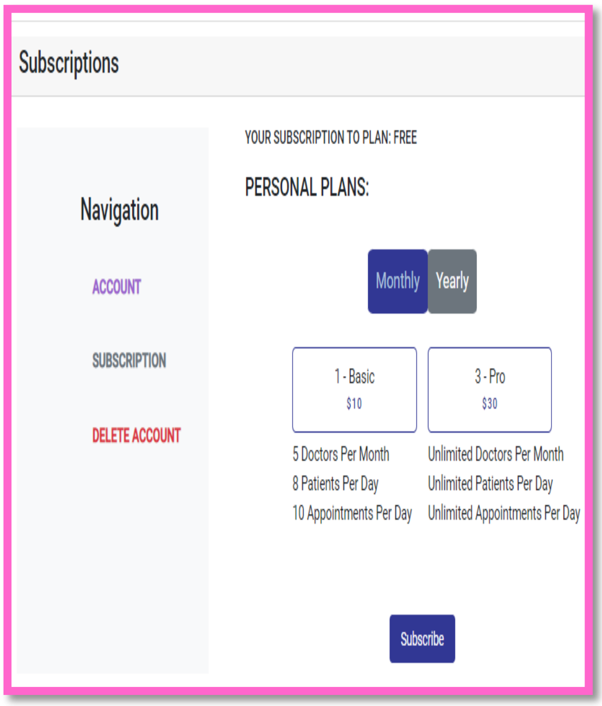


Figure III.8: admin account interface.

Since this app is not free, there are monthly and yearly subscriptions that vary depending on the administrator's request in the number of doctors, patients and appointments as shown in Figure III.9 and Figure III.10.



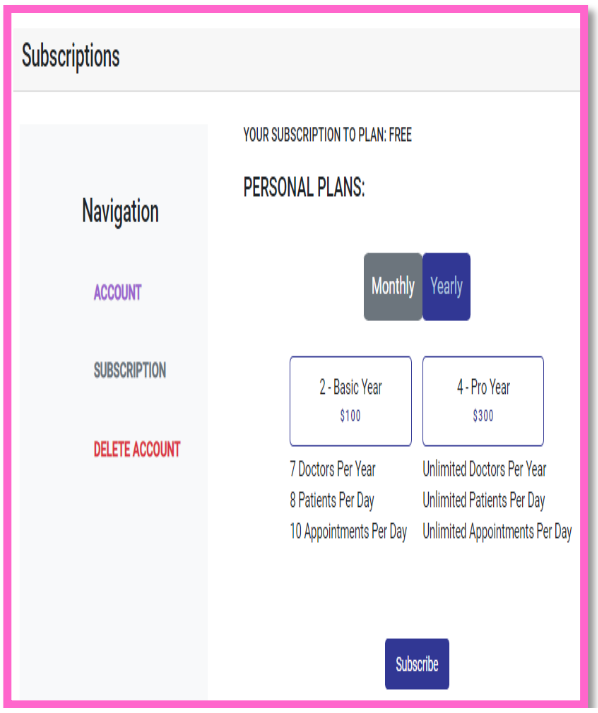
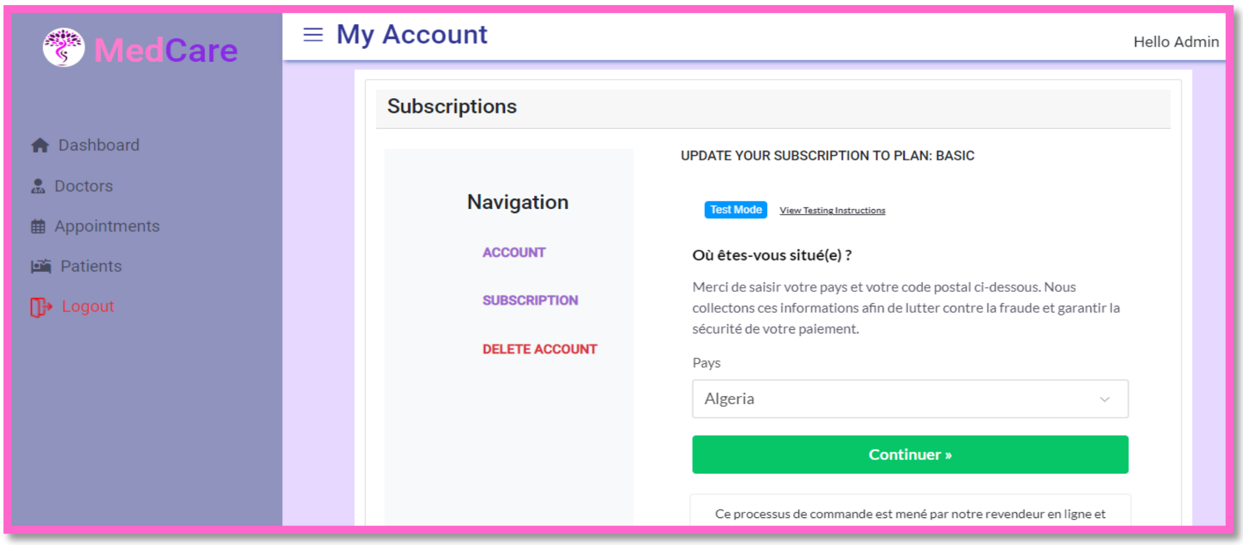


Figure III.9: monthly subscriptions interface **Figure III.10:** yearly subscriptions interface .

Once the admin chooses the type of subscription, he registers the country of residence and chooses the payment method either by card or PayPal as shown in Figure III.11 and III.12.



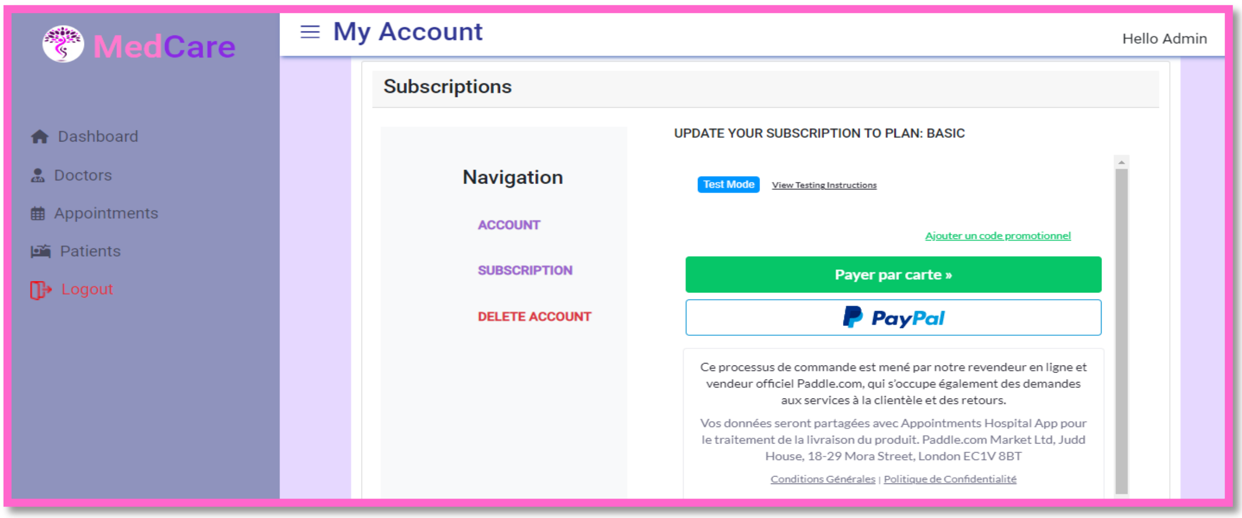
Où êtes-vous situé(e) ?

Merci de saisir votre pays et votre code postal ci-dessous. Nous collectons ces informations afin de lutter contre la fraude et garantir la sécurité de votre paiement.

Pays: Algeria

Ce processus de commande est mené par notre revendeur en ligne et

Figure III.11: Subscriber's country interface.



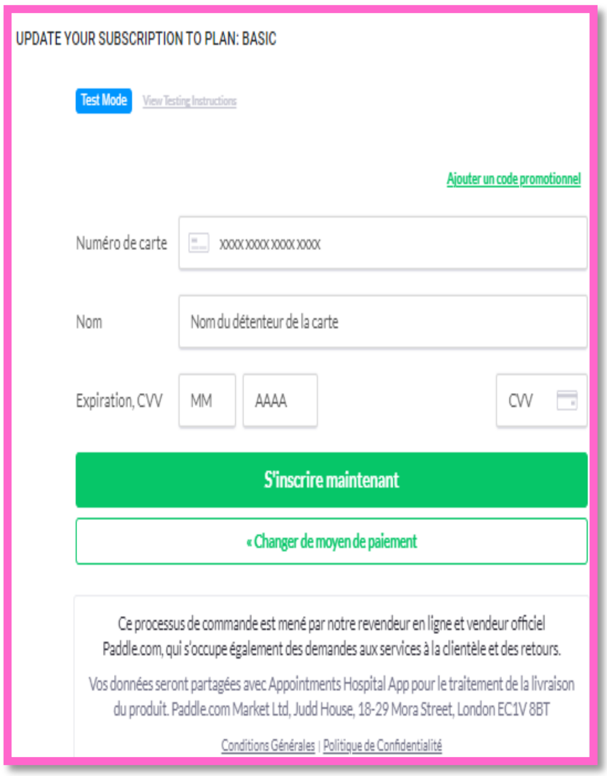
Ce processus de commande est mené par notre revendeur en ligne et vendeur officiel Paddle.com, qui s'occupe également des demandes aux services à la clientèle et des retours.

Vos données seront partagées avec Appointments Hospital App pour le traitement de la livraison du produit. Paddle.com Market Ltd, Judd House, 18-29 Mora Street, London EC1V 8BT

Conditions Générales | Politique de Confidentialité

Figure III.12: Payment method interface.

After the admin chooses the payment method, he must fill in the data in both cases and press the login button as shown in Figure III.13 and Figure III.14 and an email will be sent to the admin with all the information confirming his subscription .


UPDATE YOUR SUBSCRIPTION TO PLAN: BASIC
Test Mode
View Testing Instructions
Ajouter un code promotionnel
Numéro de carte
XXXX XXXX XXXX XXXX
Nom
Nom du détenteur de la carte
Expiration, CVV
MM
AAAA
CVV
S'inscrire maintenant
« Changer de moyen de paiement

Ce processus de commande est mené par notre revendeur en ligne et vendeur officiel Paddle.com, qui s'occupe également des demandes aux services à la clientèle et des retours.

Vos données seront partagées avec Appointments Hospital App pour le traitement de la livraison du produit. Paddle.com Market Ltd, Judd House, 18-29 Mora Street, London EC1V 8BT

Conditions Générales | Politique de Confidentialité

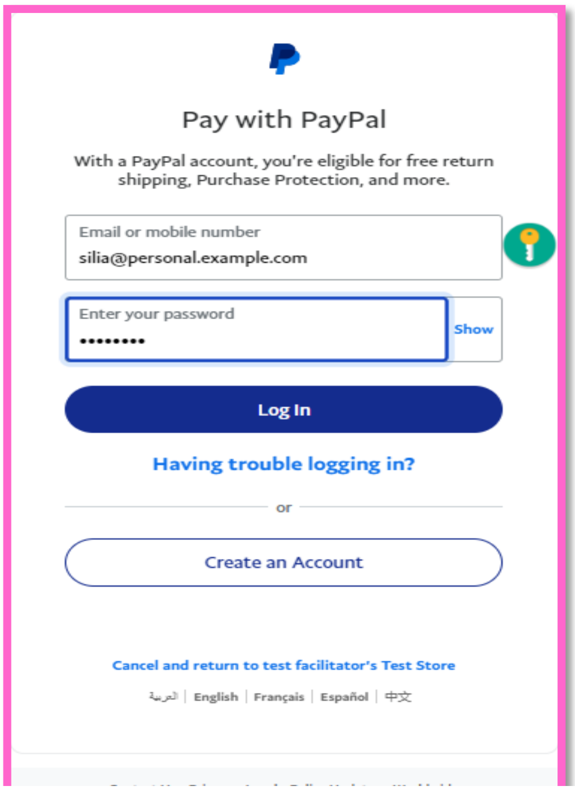
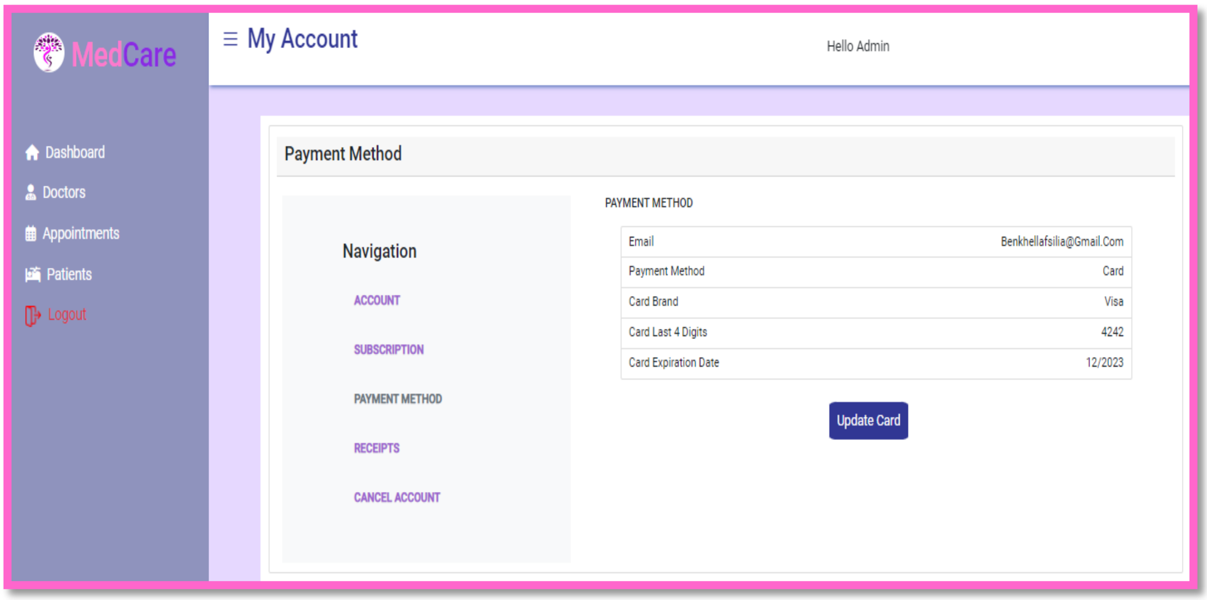

Pay with PayPal
With a PayPal account, you're eligible for free return shipping, Purchase Protection, and more.
Email or mobile number
silia@personal.example.com
Enter your password
••••••••
Show
Log In
Having trouble logging in?
or
Create an Account
Cancel and return to test facilitator's Test Store
العربية | English | Français | Español | 中文

Figure III.13: card payment interface. **Figure III.14:** PayPal payment interface.

After the administrator is registered, all his information will be recorded in his account as the payment method, the amount paid, and the start and end date of the subscription period as shown in Figure III.15 and Figure III.16.

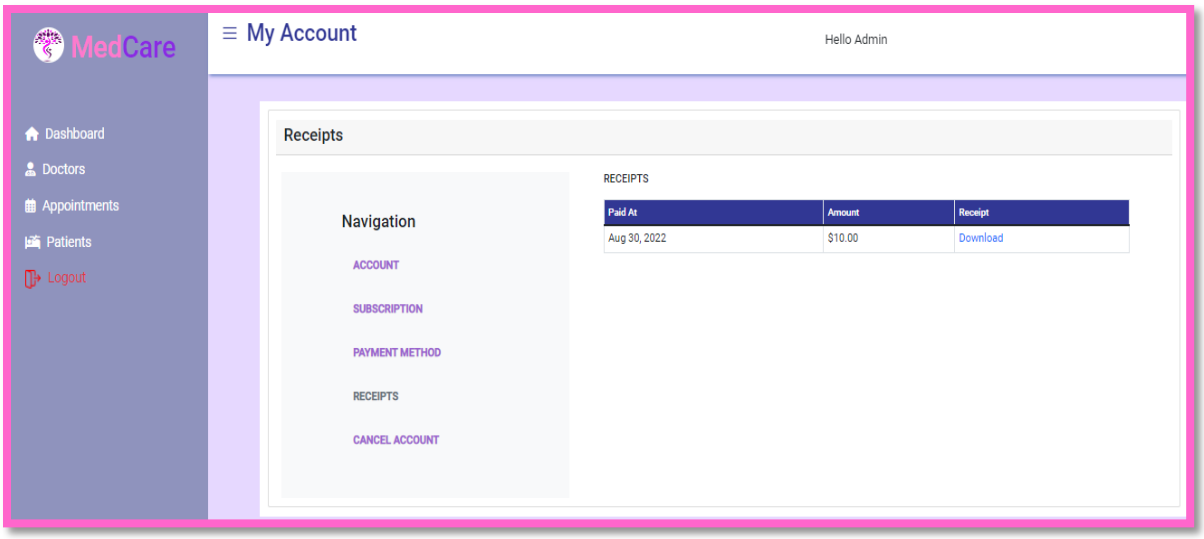


Payment Method

PAYMENT METHOD

Email	Benkhellafsilia@Gmail.Com
Payment Method	Card
Card Brand	Visa
Card Last 4 Digits	4242
Card Expiration Date	12/2023

Figure III.15: Payment Method Interface .



Receipts

RECEIPTS

Paid At	Amount	Receipt
Aug 30, 2022	$10.00	Download

Figure III.16: Receipts interface.

If the admin decides to pause their account or permanently delete it, they must press the button cancel account as shown in Figure III.17

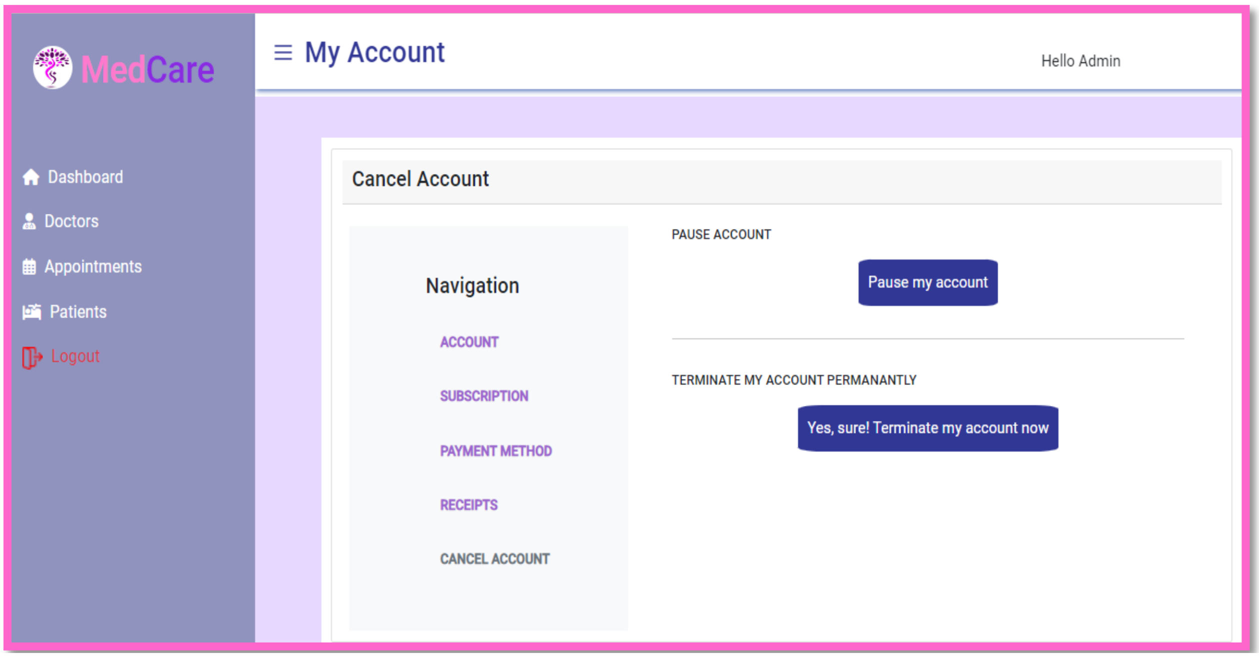


Figure III.17 : Cancel Account interface.

Dashboard:

After identification, the system offers the admin the interface as shown in Figure III.18.

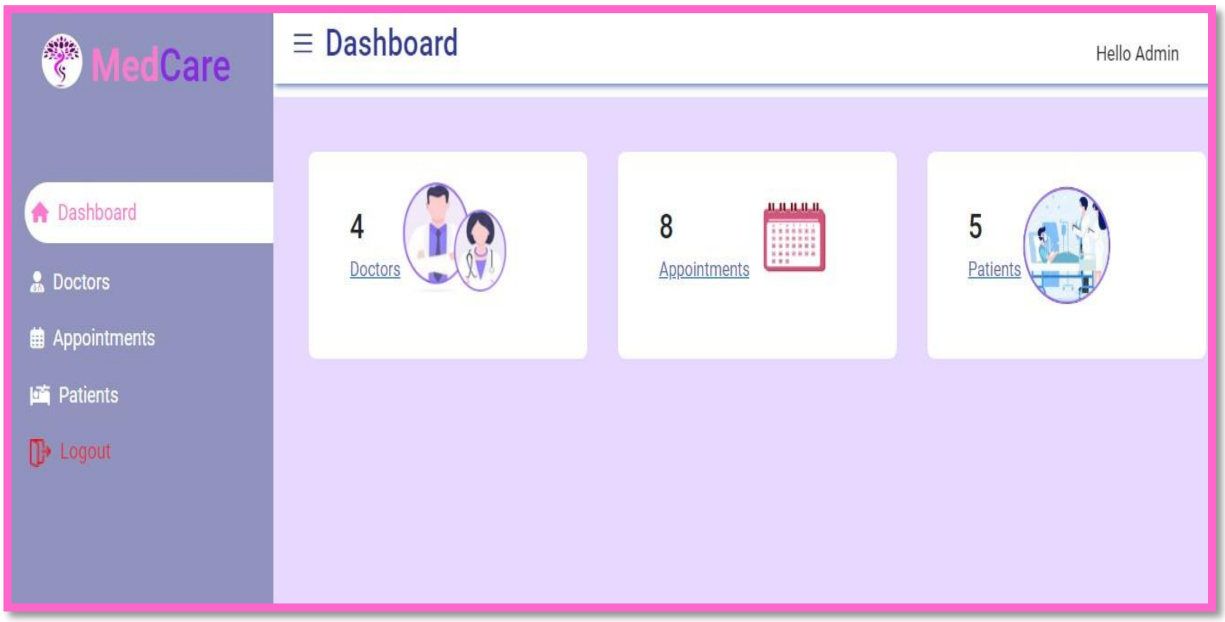
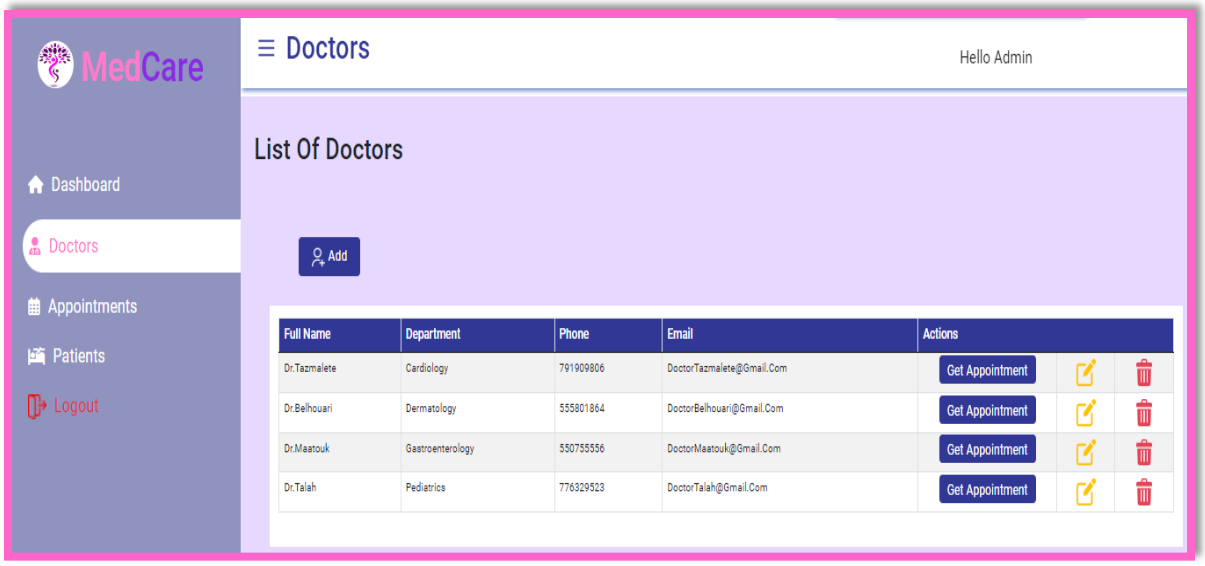


Figure III.18: Dashboard interface.

The latter allows the admin to do the following actions:

➢ Consult the list of doctors.



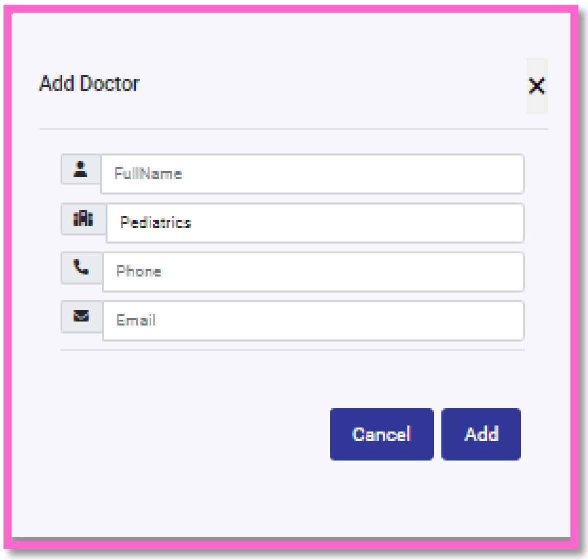
List Of Doctors

Full Name	Department	Phone	Email	Actions		
Dr.Tazmalete	Cardiology	791909806	DoctorTazmalete@Gmail.Com	Get Appointment		
Dr.Belhouari	Dermatology	555801864	DoctorBelhouari@Gmail.Com	Get Appointment		
Dr.Maatouk	Gastroenterology	550755556	DoctorMaatouk@Gmail.Com	Get Appointment		
Dr.Talah	Pediatrics	776329523	DoctorTalah@Gmail.Com	Get Appointment		

Figure III.19: List of Doctors (admin Interface)

➢ Add and delete a doctor with the ability to modify his dataas shown in Figure III.20 and Figure III.21.

➢ Add, edit and delete appointments with the possibility of consulting the list of appointments as shown in Figure III.21, III.22 and III.23,



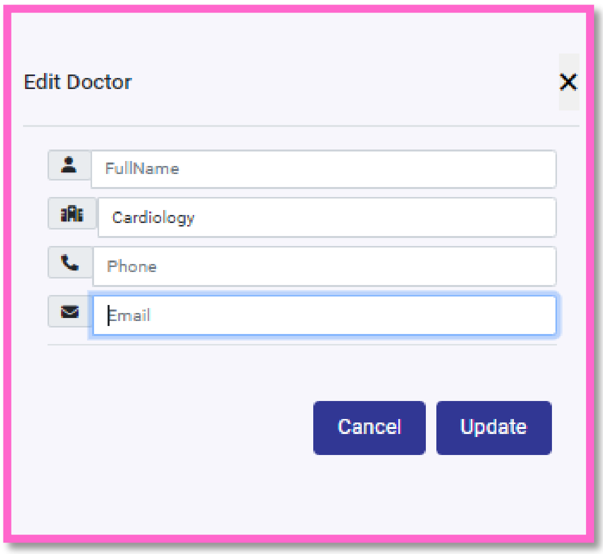
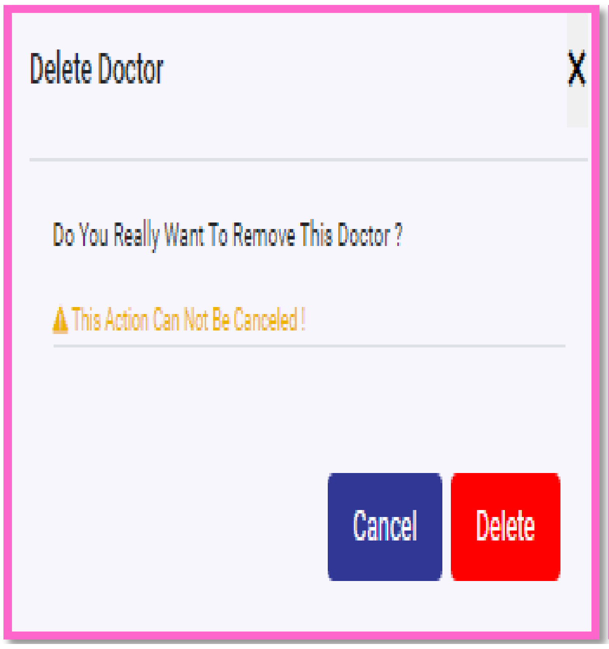


Figure III.20: Add Doctor interface.

Figure III.21: Edit Doctor interface.



Delete Doctor
X
Do You Really Want To Remove This Doctor ?
⚠ This Action Can Not Be Canceled !
Cancel
Delete

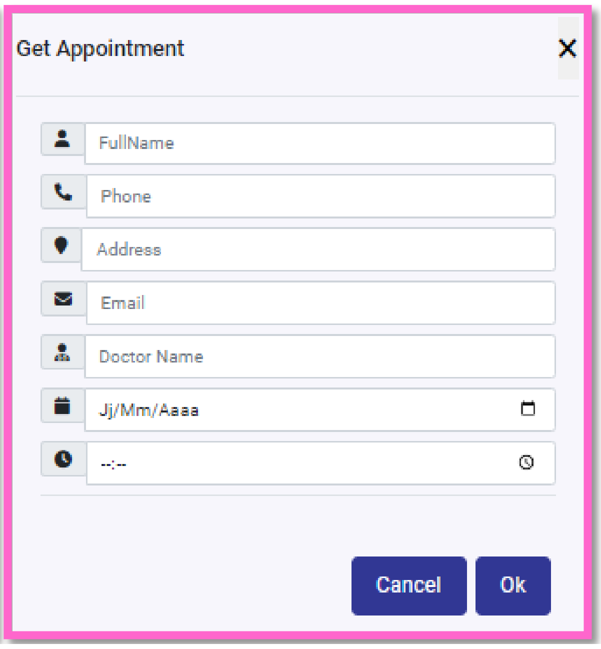
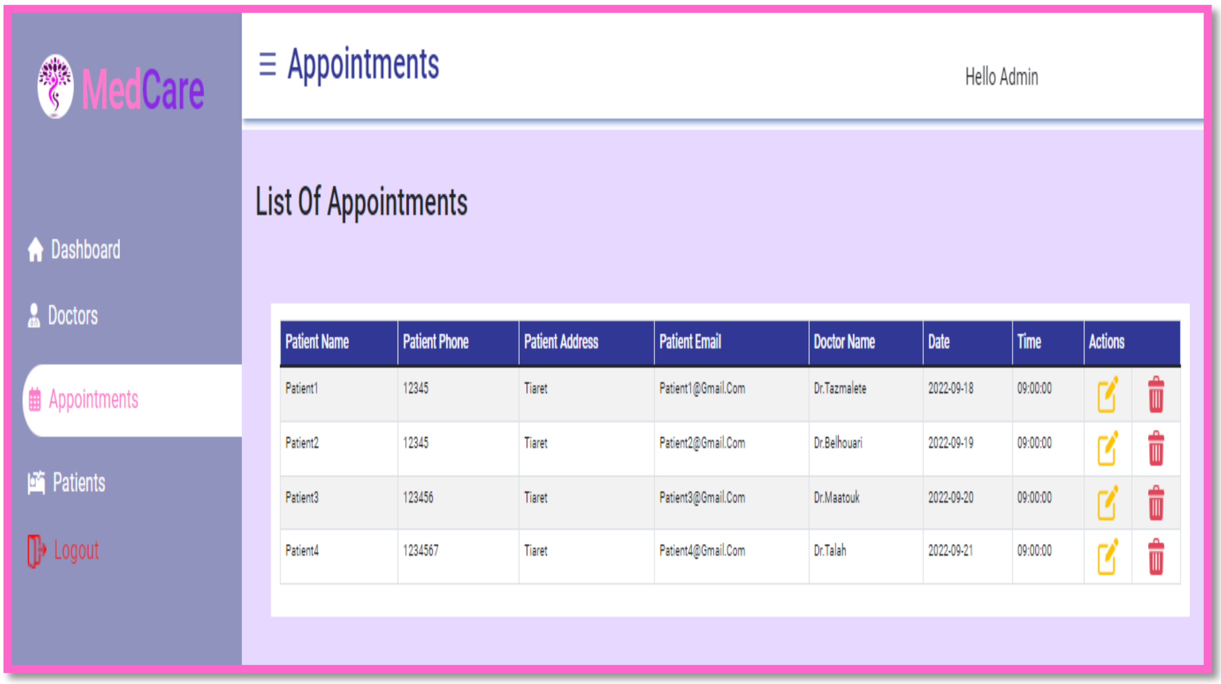


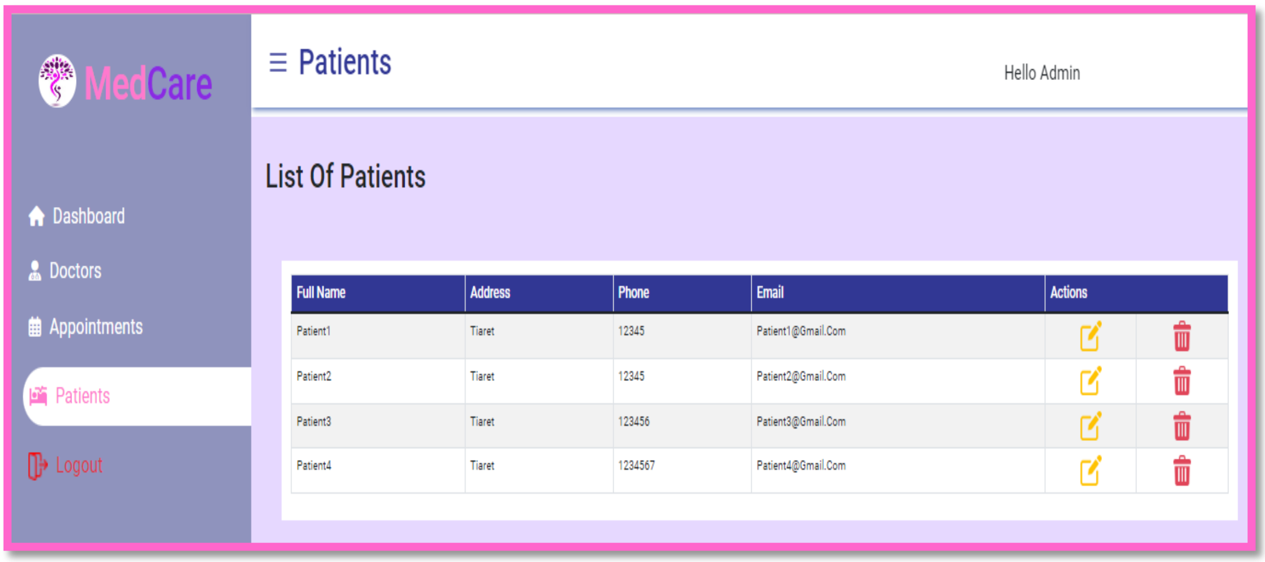
Figure III.22: Delete Doctor interface. **Figure III.23:** Get Appointment interface.



Patient Name	Patient Phone	Patient Address	Patient Email	Doctor Name	Date	Time	Actions
Patient1	12345	Tiaret	Patient1@Gmail.Com	Dr.Tazmalete	2022-09-18	09:00:00	
Patient2	12345	Tiaret	Patient2@Gmail.Com	Dr.Belhouari	2022-09-19	09:00:00	
Patient3	123456	Tiaret	Patient3@Gmail.Com	Dr.Maatouk	2022-09-20	09:00:00	
Patient4	1234567	Tiaret	Patient4@Gmail.Com	Dr.Talah	2022-09-21	09:00:00	

Figure III.24: List of Appointments (admin Interface) .

➢ Add and delete a patient , with the ability to modify his data as shown in Figures III.25



List Of Patients

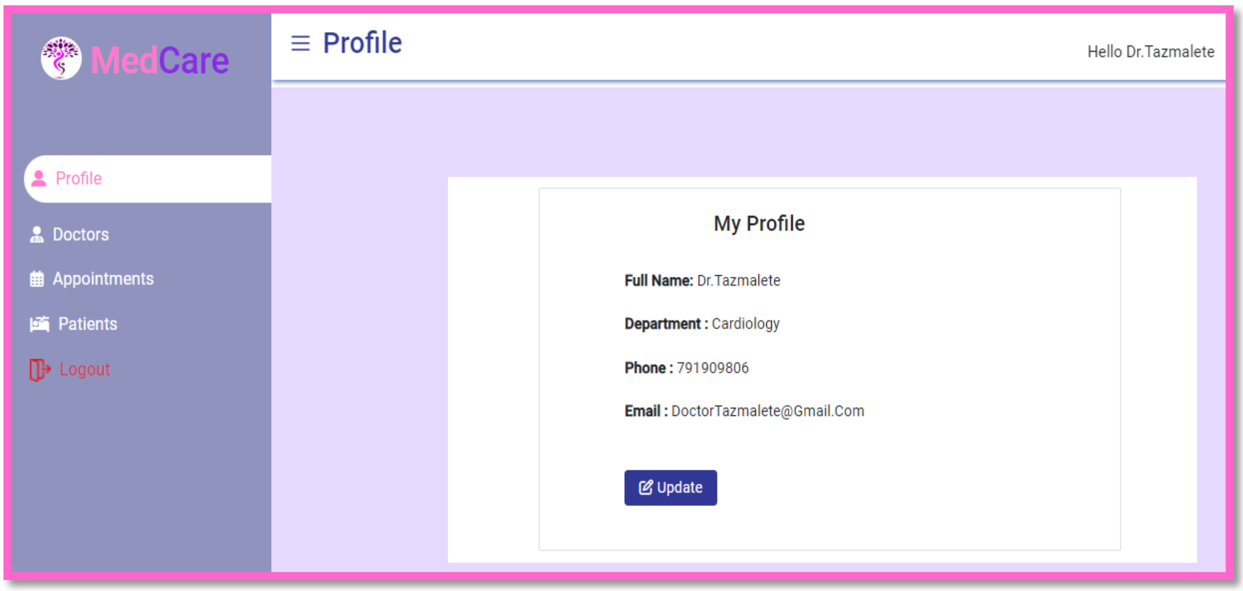
Full Name	Address	Phone	Email	Actions	
Patient1	Tiaret	12345	Patient1@Gmail.Com		
Patient2	Tiaret	12345	Patient2@Gmail.Com		
Patient3	Tiaret	123456	Patient3@Gmail.Com		
Patient4	Tiaret	1234567	Patient4@Gmail.Com		

Figure III.25: List of Patients (admin Interface)

3.7.2 Doctor interfaces:

After identification, the doctor can do the following actions:

➢ Consultation and update of his profile with the possibility of changing his information.



My Profile

Full Name: Dr.Tazmalete

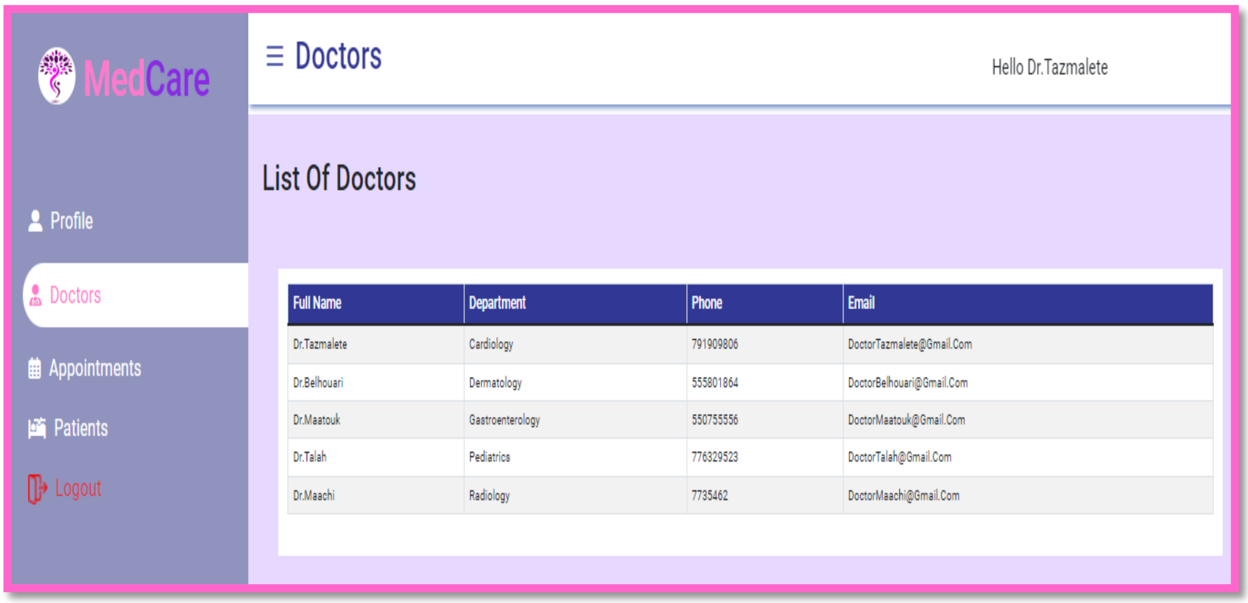
Department : Cardiology

Phone : 791909806

Email : DoctorTazmalete@Gmail.Com

Figure III.26: Doctor Profile interface .

➢ Consult the list of doctors.

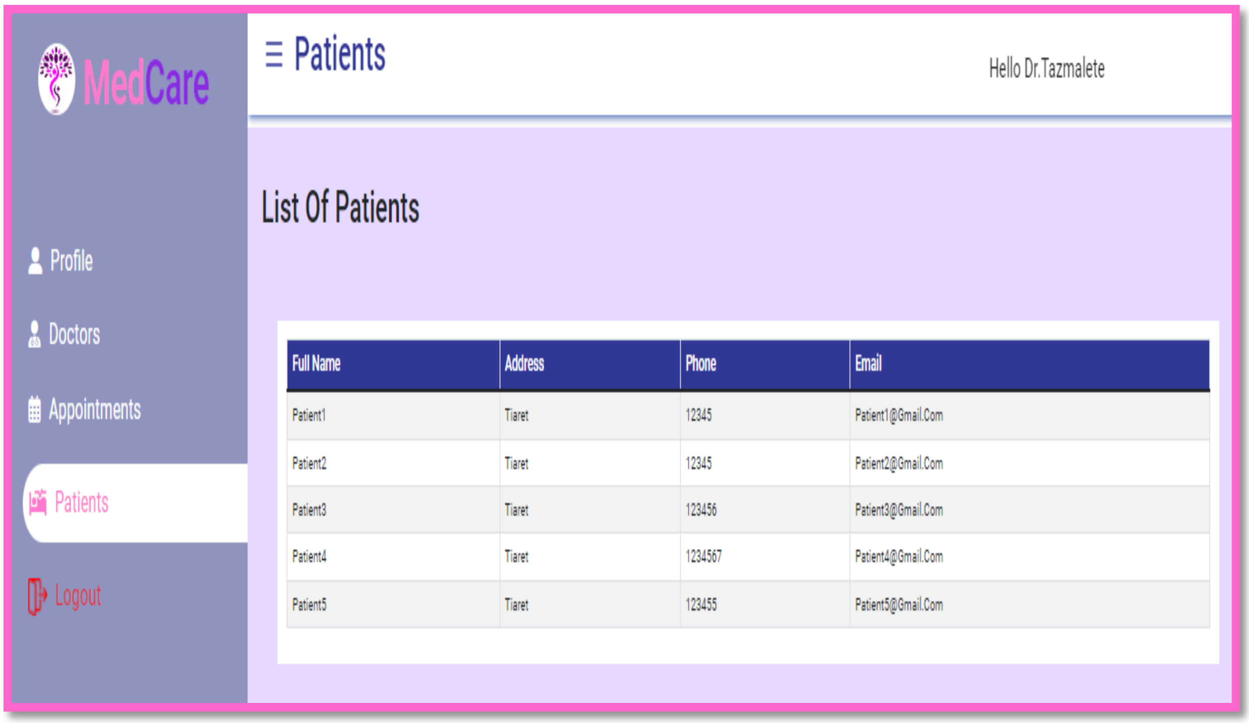


List Of Doctors

Full Name	Department	Phone	Email
Dr.Tazmalete	Cardiology	791909806	DoctorTazmalete@Gmail.Com
Dr.Belhouari	Dermatology	555801864	DoctorBelhouari@Gmail.Com
Dr.Maatouk	Gastroenterology	550755556	DoctorMaatouk@Gmail.Com
Dr.Talah	Pediatrics	776329523	DoctorTalah@Gmail.Com
Dr.Maachi	Radiology	7735462	DoctorMaachi@Gmail.Com

Figure III.27: List of Doctors (Doctor Interface)

➢ Consult the list of patients.

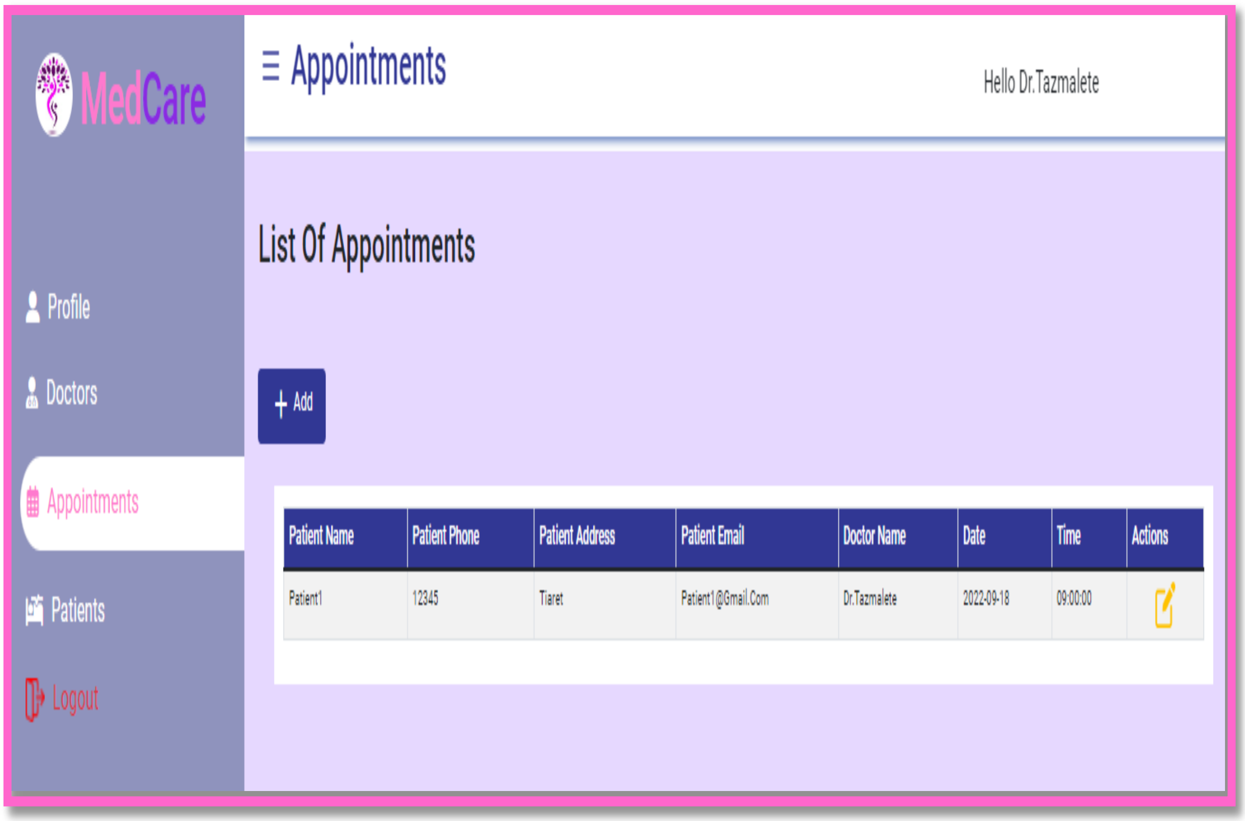


List Of Patients

Full Name	Address	Phone	Email
Patient1	Tiaret	12345	Patient1@Gmail.Com
Patient2	Tiaret	12345	Patient2@Gmail.Com
Patient3	Tiaret	123456	Patient3@Gmail.Com
Patient4	Tiaret	1234567	Patient4@Gmail.Com
Patient5	Tiaret	123455	Patient5@Gmail.Com

Figure III.28: List of Patients (Doctor Interface)

- Add and edit appointment with the possibility of consulting the list of appointments.



☰ Appointments

Hello Dr.Tazmalete

List Of Appointments

Patient Name	Patient Phone	Patient Address	Patient Email	Doctor Name	Date	Time	Actions
Patient1	12345	Tiaret	Patient1@Gmail.Com	Dr.Tazmalete	2022-09-18	09:00:00	

Figure III.29: List of Appointments (Doctor Interface).

3.8 Discussion

Because of the difficulties encountered especially during the programming phase of our MedCareSaaS application, there is still a lot to be done about the functionality offered by this application, for example the possibility of printing the prescription for each appointment/consultation, and why not the possibility of sending this prescription to another actor such as the pharmacist etc.

3.9 Conclusion

In this last chapter, we have described in detail the implementation and realization part. Indeed, we have shown the hardware and software environment used during the implementation as well as the description of the traditional application targeted by our approach. Moreover, we have highlighted the development process followed and the inspired methodology adopted and the explanation of the different interfaces of our successful SaaS applications.

GENERAL CONCLUSION AND FUTURE WORK

CONCLUSION AND FUTURE WORK

The impact of integrating the SaaS service model and multi-tenant architecture into cloud-hosted applications is easily underestimated but can raise major specification and implementation challenges.
We have written the work carried out in this thesis, in the context of the creation and implementation of a multi-tenant application named "MedCare" in the cloud SaaS with the framework named Laravel.
The overall objective of this work was to distinguish the steps adopted for the migration of the Traditional MedCare application to a multi-tenant SaaS application.

Much of this thesis has focused on:

- The concise establishment of a state-of-the-art on virtualization-based paradigms, including cloud computing, as well as their deployment and service models.
- The description, the explanation, and the distinction between some models of maturity and realization of multi-tenant architecture (Uni/Multi database) in cloud computing.
- The presentation and learning of a framework named Laravel, which provides superb libraries for the SaaS model and the Multitenancy architecture.
- The implementation of a typical SaaS application named MedCare with the Laravel framework. Though these features are quite restricted (limited) in terms of the functionality or multi-tenant architecture achieved by this manuscript.

This limitation comes mainly from the programming difficulties encountered especially during the integration of the multitenancy laravel libraries (Spatie)[48] used during the implementation, these limitations negatively and considerably influence the credibility of our work(application) which remains with basic SaaS architecture a domain and not multidomain based primarily on the LAREVL framework. i.e the proposed realization must be tested with other SaaS and multitenancy libraries, and on other platforms, such as symphony[49]. Codnighter[50], Django[51]….etc.

In the short term, the most important works to be carried out concern the integration of the laravel library named Spatie in order to finalize, refine the Multitenancy itself, with separate URLs (multidomain) in our application.The extensibility of this application, in terms of server and database typing, and not only with the SQL databases (mysql) carried out in this thesis. We plan to extend this study by adopting other databases (OracleDB, PostgresSQL) and go further with other types of NoSql databases (MongoDB, Cassandra, GoogleDatasore).

In the long term, the virtualized environment targeted by the multi-tenant architecture in this thesis, is based on cloud computing as a research paradigm, we envisage as a perspective, to propose an implementation with other virtualized environments, that is- ie, the implementation of multitenancy in fog computing, in edge computing, etc.

The work present in this thesis is a first stone concerning an understanding, a holistic survey of multitenancy solutions in all virtualized environments, cloud, fog edge..etc. It can be broken down into different extensions, whether its type of administration server (windows/linux), the type of web server (appach, Nginx, Tomcate), the type of database (SQL/NoSql) that the duration of a memory did not allow to approach, and which constitute as many avenues of research to be explored.

References

References

[1] Gavlov, N.: 25 Must-Know Cloud Computing Statistics in 2022. https://webtribunal.net/blog/cloud-computing-statistics/#gref. Accessed 26 Apr 2022.

[2] R.Buyya, C.Vecchiola and S. ThamaraiSelvi, " Mastering Cloud Computing: Foundations and Applications Programming ", Elsevier Inc,USA , pp. 23, (2013).

[3] A. Zerrouki and I. Zerrouki, " Ordonnancement des workflows scientifiquesdans un environnement Cloud Computing ", Master thesis, Dr.TaharMoulayUniversity, Saida, pp. 15-18, (2020).

[4] S. Benkouider Sahraoui and Y. Mansouri, " Etude de sécurité d’une extension Cloud ", Master thesis, A/Mira University, Bejaia, pp. 2-3, (June 2017).

[5] www.pioneer-search.com/blog/2021/04/the-history-of-cloud-computing-2021, Accessed February 16, 2022.

[6] A. Bensalah, " On the Performance Evaluation of Cloud Computing Environments Metrics using PRISM Model Checker", Master thesis, Mohamed BoudiafUniversity, M’sila, p. 2, (2017).

[7] I. Strumberger, N. Bacanin, M. Tuba and E. Tuba, "Resource Scheduling in Cloud Computing Based on a Hybridized Whale Optimization Algorithm", Singidunum University, Belgrade,Serbia, pp. 4,(November 2019).

[8] K. Akbi and M. Zehri, " Etude et mise en vert d’une solution Cloud computing privé au sein de l'université d’Ouargla", Master thesis, KaskiMerbahUniversity, Ouargla, pp. 2, (2013).

[9] D. Gallagher and P. Pritzker, " NIST Cloud Computing Standards Roadmap ", National Institute of Standards and Technology, USA, pp. 08, (July 2013).

[10] T. Saaoui and F. Touahria , " Etude et simulation d’une solution Cloud Computing au sein d’une entreprise privée Cas d’étude : Cevital", Master thesis, A/Mira University, Bejaia, pp. 43-44, (2020).

[11] M. Ailam and M. Bafdel, "Ordonnancement de tâches dans le Cloud ", Master thesis, Mouloud MammeriUniversity, Tizi-Ouzou, pp. 3-8, (2019).

[12] K. Chandrasekaran, "Essentials of Cloud Computing",Chapman and Hall/CRC, New York, pp. 46-64, (December 2014).

[13] https://www.plesk.com/blog/various/iaas-vs-paas-vs-saas-various-cloud-service-models-compared/ Accessed February 24, 2022.

[14] Clouds etc.https://www.everwin.fr/la-mediatheque/514-cloud-computing-saas-iaas-paas

[15] https://iamondemand.com Accessed February 22, 2022.

[16] https://www.thousandeyes.comAccessedFebruary 26, 2022.

References

[17] https://www.comptia.org/content/articles/what-is-paas Accessed February 20, 2022.

[18] https://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies Accessed February 28, 2022.

[19] R. Krebs, C. Momm and S. Kounev, "Architectural ConcernsIn Multi-tenant SaaS Applications", CLOSER 2012 - Proceedingsof the 2nd International Conference on Cloud Computing and Services Science,Setúbal, Portugal, (2012).

[20] O. Schiller, "Supporting Multi-tenancy in Relational Database Management Systems for OLTP-style Software as a Service Applications", Stuttgrat University, Germany, pp. 22, (2015).

[21] A. Simitos, "Multi-tenant Databases for SaaS: Security and Privacy issues", School of Science and Technology, Thessaloniki, pp. 4-9, (January 2016).

[22] I. Nikolaou and L. Anthopoulos, " Multi-Tenancy in Smart City Platforms", In Companion Proceedings of the Web Conference 2022 (WWW ’22 Companion), Virtual Event, Lyon, France, (April 2022).

[23] D. Betts, A. Homer, A. Jezierski, M. Narumoto and H. Zhang, "Developing Multi-tenant Applications for the Cloud on Microsoft Windows Azure", 3rd ed, Microsoft patterns & practices, (2013).

[24] R. Mietzner, T. Unger, R. Titze and F. Leymann, "Combining Different Multi-tenancy Patterns in Service-Oriented Applications" in International Conference on Enterprise Distributed Object Computing, pp. 131-140, (2009).

[25] C.-P. Bezemer and A. Zaidman. "Multi-tenant SaaS applications: maintenance dream or nightmare?" In Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution, Antwerp, Belgium ,
pp. 1-14, (September 2010).

[26] S. Kanade and R. Manza , "A Comprehensive Study on Multi Tenancy in SAAS Applications", International Journal of Computer Applications (0975 – 8887) ,Volume 181 – No. 44, pp. 25-27, (March 2019).

[27] G. Karataş, F. Can, G. Doğan, C. Konca andA. Akbulut, " Multi-tenant architectures in the cloud: A systematic mapping study", 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, pp. 2, (September 2017).

[28] J. Kabbedijk, C-P. Bezemer, S. Jansen and A. Zaidman, "Defining multi-tenancy: A systematic mapping study on the academic and the industrial perspective", Journal of Systems and SoftwareVolume 100Issue C. pp 139–148, February (2015).

References

[29] H. Yaish and M. Goyal, "A Multi-tenant Database Architecture Design for Software Applications", IEEE 16th International Conference on Computational Science and Engineering, pp. 933-940, (2013). doi: 10.1109/CSE.2013.139.

[30]https://networkinterview.com/approaches-of-multi-tenancy-in-cloud/, Accessed June 08, 2022.

[31] F. Doudou, G. Blanc, T. Okuda, Y. Kadobayashi, and S. Yamaguchi. "Toward quantified risk-adaptive access control for multi-tenant cloud computing", In The 6th Joint Workshop on Information Security, pp. 1-14, (2011).

[32] V.V. bandgar, G.A. Fattepurkar, C.M.Jadhav, S.M. Kawale, "Development of Multitenant Application as SAAS", In International Journal of Engineering Research & Technology, pp. 2, (2015).

[33] M. Yang and H. Zhou, "New Solution for Isolation of Multi-tenant in cloud computing", 3rd International Conference on Mechatronics, Robotics and Automation, Beijing, China, pp. 334-335, (2015).

[34] R.O.Antonio, M. Noguera, J.L. Garrido, K. Benghazi, and L. Chung. "Multi-Tenancy Multi-Target (MT 2): A SaaS Architecture for the Cloud." In International Conference on Advanced Information Systems Engineering,. Springer, Berlin, Heidelberg, pp. 217, (2012).

[35] E.Saleh, N. Shaabani and C. Meinel, “A Framework for Migrating Traditional Web Applications into MultiTenantSaaS”, INFOCOMP 2012 : The Second International Conference on Advanced Communications and Computation, ISBN : 978-1- 61208-226-4. pp 100-105, (2012).

[36] C-P.Bezemer, A.Zaidman, B. Platzbeecker, T. Hurkmans, and A. Hart: “Enabling multi-tenancy: An industrial experience report”. Proc. 26th IEEE Int. Conf. on Software Maintenance Timioara, Romania, pp. 1-8, (2010).

[37] www.memoireonline.com / Licence Professionnelle en Statistique et Informatique Décisionnelle/2018-2019 Accessed March 2, 2022.

[38] php.developpez.com] Accessed March 5, 2022.

[39] Laravel.sillo.org/cours-laravel-8-les-bases-presentation-generaleAccessedJune 18,2022.

[40] https://techterms.com Accessed May 5, 2022.

[41] www.apachefriends.org Accessed February 27, 2022.

[42] https://help.mailtrap.io/article/12-getting-started-guide Accessed July 28, 2022.

[43] https://github.com/Amel-Abr/Hospital-Appoint-App.git

[44] https://github.com/Amel-Abr/Hospital-Appoint-SAAS-MT.git

[45] E Nugraheni , "Migration of web application SIMA into multi-tenant SaaS", International Conference on ICT for Smart Society, pp.1-4, (2014).
[46] https://www.udemy.com/course/laravel-saas/
[47] L. Bechroune and Z. Aghouiles, "Proposition d’un système multi-agents pour lagestion du cloudcomputing et évaluation deses performances", Master thesis, A/Mira University, Bejaia, pp. 35, (septembre 2020).
[48] https://spatie.be/docs/laravel-permission/
[49] https://symfony.com/
[50] https://codeigniter.com/
[51] https://djangopackages.org/grids/g/multi-tenancy/