REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : [Génie Informatique]

Par :

ELAACHI Rihem Zoulikha **CHERIFI Miloud**

Sur le thème

Élaboration d'un système d'abonnement multi-locataire SaaS avec une base de données Unique.

Soutenu publiquement le ../../2023 à Tiaret devant le jury composé de :

Mr ALEM Abdelkader	MAA	Université tiaret	Président
Mr HATTAB Nourddine	MAA	Université tiaret	Encadreur
Mr ABID Khaled	MAA	Université tiaret	Examinateur

2022-2023

Remercîments

Tout d'abord, nous remercions Dieu de nous avoir donné la force, la volonté et la patience nécessaires pour mener à bien ce travail.

Nous tenons également à exprimer nos sincères remerciements à notre chef de projet, Monsieur [Hattab Noureddine], pour son soutien continu et ses précieux conseils. Ses idées et ses recommandations ont été très utiles au cours de cette entreprise.

Nous tenons également à remercier tout particulièrement les membres du jury qui ont généreusement donné de leur temps pour évaluer ce modeste travail.

Nos remerciements vont également à toute l'équipe pédagogique de [Génie Civil] pour la qualité de leur enseignement et leur accompagnement tout au long de nos études.

Enfin, nous tenons à remercier chaleureusement notre famille et nos amis pour leur soutien et leurs encouragements tout au long de ce projet. Leur présence et leurs encouragements ont été pour nous une précieuse source d'inspiration et de motivation.

Dédicace

Je dédie ce travail :

À ma mère, celle qui a illuminé ma vie avec une touche d'amour, tu es mon héroïne et ma source d'inspiration. Tu es le centre de ma vie, ta présence me réconforte dans les moments difficiles et m'encourage à poursuivre mes rêves. Tu es le pilier solide sur lequel j'ai construit mes succès.

À mon père, tu es mon roc, mon guide bienveillant, ton soutien constant m'a permis de grandir et de prospérer. Je te suis infiniment reconnaissant pour toutes les leçons que tu m'as enseignées.

À ma grand-mère, tu es chaleur et tendresse, que Dieu prolonge ta vie.

À mes sœurs chères et adorées, Rania, DJihane et Soulef.

À mon frère mon héro Houssem.

À ma tante, ma deuxième mère "Boumba", que Dieu préserve ta présence dans ma vie.

À toute ma famille, que Dieu préserve votre présence dans ma vie.

À mon binôme cherifi Miloud, je suis reconnaissant pour l'opportunité qui m'a été donnée de travailler à tes côtés. Merci pour notre collaboration harmonieuse.

À toutes mes amies, Sara, Amina, Cherifa, Feriel, Kheira, et à tous ceux qui m'ont soutenu et encouragé.

Rihem

Dédicace

Je dédie ce travail :
Jexprime mon profond respect et ma sincère gratitude aux personnes qui me sont les plus chères au monde, à mes parents qui ont su me faire confiance et me soutenir en toutes circonstances au cours de ma vie.
C'est avec émotion que je leur exprime toute ma gratitude mon admiration et mon profond respect.
A mes chères frères pour leurs soutiens et leurs encouragements Houssine, Ammar, Nour El houda , Mohamed, et à ma petite sœur Fatima El Zohra et La femme de mon frère Fatiha , Younes , Lokman . . . toute la famille Cherifi .
Je remercie Mon partenaire de travail, Rihem et à tous mes collègues de ma carrière d'études, Sohaib, Marwan, Abdul Samad, Mohammed, Hussein, Khaled, Sabrina , Abdnour , Imade... J'ai passé les plus beaux moments avec eux.
Et à tous ceux que je connais et que je n'ai pas cités, et à tous les étudiants en informatique de l'Université Ibn Khaldoun de Tiaret promo 2022/2023.
Miloud

Résumé

Résumé Avec la progression rapide de la technologie, de nombreuses architectures ont été brusquement remplacées par de nouvelles solutions afin de surmonter les limitations de leurs prédécesseurs. Parmi ces nouvelles perspectives, les architectures multi-locataires se sont avérées essentiels en tant qu'avantages supplémentaires pour les firmes qui utilisent des systèmes virtualisés tels que le Cloud, le Fog, l'Edge, et autres. Elles sont considérées aujourd'hui comme une compétence vitale pour les ingénieurs et les développeurs.

Lorsqu'il s'agit de la multi-location, il existe deux approches distinctes : la multi-location au niveau de la base de données et multi-location au niveau fonctionnel. Cependant, le sujet abordé dans ce mémoire se concentre exclusivement sur la multi-location de type base de données et ne traite pas de l'aspect fonctionnel. Afin de clarifier cette tendance architecturale dans le contexte des bases de données, notre intention initiale pour ce mémoire était d'adopter une architecture multi-locataire avec une solution de base de données multiple. Cependant, en raison de contraintes de programmation et d'un temps très limité pour la réalisation de ce mémoire, nous allons rapidement pris la décision d'opter pour une solution de base de données unique, qui présente moins de complexités sur le plan de la programmation.

Le principal objectif de ce mémoire est de dissiper l'ambiguïté conceptuelle et de relever les défis l'implémentation liés à cette tendance, en utilisant un système d'abonnement simple de facturation comme application cible. Ce mémoire se concentre sur l'utilisation du Framework Laravel pour développer cette application d'abonnement distincte.

Mots-clés :

Système virtualisé, Cloud, SaaS, Multi-tenancy.

Abstract

With the rapid advancement of technology, many structures have been suddenly replaced with new solutions to overcome the limitations of their predecessors. Among these new perspectives, multi-tenant architectures have proven to be necessary as additional advantages for companies using virtual systems such as Cloud, Fog, Edge, and others. Today, it is considered a vital skill for engineers and developers.

When addressing the issue of multitenancy, there are two distinct approaches : database-level multi-tenancy and functional-level multi-tenancy. However, the topic addressed in this thesis exclusively focuses on the type of multi-tenant database and does not deal with the functional aspect. In order to illustrate this architectural trend in the context of databases, our original goal for this thesis was to adopt a multi-tenant structure with a multi-database solution. However, due to programming constraints and very limited time to achieve this thesis, we made a quick decision to choose a single-database solution, which presents fewer programming complexities.

The main objective of this thesis is to dispel conceptual ambiguities and address implementation challenges related to this direction, using a simple subscription system in billing as the targeted application. This thesis focuses on using the Laravel Framework to develop this featured subscription application.

Keywords :

Virtualized system, Cloud, SaaS, Multi-tenancy.

ملخص

مع التقّدم السريع للتكنولوجيا ، تم استبدال العديد من الهياكل فجأه بحلول جديده من أجل التغلب على قيود أسلافها. من بين هذه المنظورات الجديده ، أثبتت البنى متعدده المستأجرين أنها ضرورية كمزايا إضافية للشركات التي تستخدم أنظمة افتراضية مثل Cloud و Fog و Edge وغيرها. تعتبر اليوم مهاره حيوية للمهندسين والمطورين.

عندما يتعلق الأمر بالعديد من الإيجارات ، هناك طريقّتان متميزتان: تعدد الإيجارات على مستوى قاعده البيانات ، والتعددية على المستوى الوظيفي. ومع ذلك ، فإن الموضوع الذي يتم تناوله في هذه الأطروحة يركز حصريًا على نوع قاعده البيانات متعدده الإيجارات ولا يتعامل مع الجانب الوظيفي. من أجل توضيح هذا الاتجاه المعماري في سياق قواعد البيانات ، كان هدفنا الأصلي من هذه الأطروحة هو اعتماد بنية متعدده المستأجرين مع حل متعدد قواعد البيانات. ومع ذلك ، نظرًا لقيود البرمجة والوقت المحدود جدًا لتحقيق هذه الأطروحة ، اتخذنا قرارًا سريعًا باختيار حل قاعده بيانات واحده ، والذي يعرض تعقيدات أقل من حيث البرمجة.

الهدف الرئيسي من هذه الأطروحة هو تبديد الغموض المفاهيمي ومعالجة تحديات التنفيذ المتعلقة بهذا الاتجاه ، وذلك باستخدام نظام اشتراك بسيط في الفواتير كتطبيق مستهدف. تركز هذه الرسالة على استخدام Laravel Framework لتطوير تطبيق الاشتراك المميز هذا.

الكلمات الاساسية

متعدد الايجار - سحاب -SaaS - نظام افتراضي

La rédaction de ce mémoire a été réalisée à l'aide de l'outil LaTeX.

Table des matières

Table des figures

Liste des tableaux

Liste d'abréviation

PaaS Platform as a Service

SaaS Software as a Service

IaaS Infrastructure as a Service

UML Unified Modeling Language

PDF Portable Document Format

NIST National Institute of Standards and Technology

MVC Model-View-Controller

CISCO Commercial & Industrial Security Corporation

Introduction générale

Contexte & Sujet

La virtualisation de l'informatique classique a engendré l'émergence de nouveaux paradigmes tels que le Cloud computing, le Fog computing, le Edge computing, et d'autres encore.
Le présent mémoire se concentre principalement sur le paradigme du cloud, qui connaît une croissance fulgurante grâce à la virtualisation et est largement employé pour offrir des services sur Internet. L'adoption de ce paradigme a profondément transformé notre perception de la science des services. Parmi les trois modèles offerts par le Cloud, le logiciel en tant que service (SaaS) est considéré comme le modèle de prestation de services le plus attrayant. Contrairement aux modèles de licence traditionnels, le SaaS permet aux utilisateurs de souscrire à un abonnement pour accéder à des services à la demande. Des recherches prévoyaient une croissance dominante et rapide du marché des services basés sur le cloud.[1]
Avec l'essor du SaaS et la popularité des paradigmes orientés services, le concept de multi-location a récemment suscité un grand intérêt dans la communauté des applications cloud. La multilocation dans le Cloud représente une voie de recherche prometteuse, qui suscite toujours beaucoup d'intérêt. Cet intérêt vient principalement de sa capacité à partager (selon des niveaux de maturité) des ressources matérielles/logicielles avec plusieurs locataires «consommateurs » du Cloud, c'est-à-dire à augmenter les profits des applications Cloud actuelles.
La multi-location est perçue différemment selon les différents modèles de service. Cependant, la multilocation a des significations différentes dans les différentes couches de service du modèle cloud « IaaS, PaaS et SaaS ». Ce travail se concentre sur la catégorie

de multi-location «SaaS », avec le niveau d'isolement le plus élevé. Plusieurs consommateurs partagent une instance d'application Cloud. A ce stade, deux modèles principaux sont décrits dans : le Multi-instance Multi-tenant (MIMT) et le Single-Instance to Multi-tenant (SIMT). Dans le modèle MIMT, un locataire utilise une instance d'application d'exécution dédiée. D'autre part, dans le modèle SIMT, les locataires partagent simultanément la même instance d'application. Parmi les avantages de ces deux modèles, on peut citer la réduction des coûts de service due au partage des ressources, ce qui conduira à une augmentation de l'utilisation.

Défis, objectif et énoncé de mémoire

Les consommateurs et les entreprises ont tendance à privilégier des coûts d'hébergement ou d'utilisation de logiciels aussi bas que possible. Lorsque les applications sont conçues et personnalisées pour un locataire spécifique, puis hébergées en tant qu'instance individuelle, les coûts d'hébergement augmentent de manière proportionnelle pour chaque locataire. Pour répondre à la différence entre les approches d'hébergement traditionnelles et le cloud, des solutions multi-locataires ont été développées. Ces solutions permettent de servir de multiples utilisateurs d'applications en créant une instance d'application unique. Chaque client peut ainsi apporter des améliorations, des personnalisations ou des modifications à l'application comme s'il était le seul utilisateur. Les solutions multi-locataires offrent une alternative puissante aux entreprises qui souhaitent obtenir des avantages concurrentiels distinctifs.

Dans le contexte d'un système logiciel en tant que service multi-locataire, deux approches conceptuelles ont été identifiées : l'utilisation de multiples bases de données et l'utilisation d'une seule base de données. L'objectif de ce travail est de comprendre le concept de la multilocation et sa pertinence dans la conception de solutions basées sur le cloud. Une étude de cas est utilisée pour fournir un contexte de recherche pertinent à partir duquel la conception est dérivée. Finalement, une application simple est développée en mettant en œuvre la multilocation pour répondre aux exigences spécifiques du cas, en utilisant une approche basée sur une seule base de données. L'objectif final de ce projet est de concevoir et développer une application de facturation simple sous forme de SaaS (Software as a Service), qui repose sur un modèle d'abonnement (basique/avancé) pour accéder à toutes ses fonctionnalités. Initialement, l'approche multi-base de données était envisagée, cependant, à la fin du projet, l'approche à base de données unique a été retenue pour la réalisation de l'application en utilisant le Framework Laravel.

Structure Générale

Ce mémoire se compose de trois chapitres complémentaires qui abordent différents aspects du sujet traité. Dans le premier chapitre, nous examinerons le paradigme le plus couramment utilisé aujourd'hui en matière d'environnement virtualisé, à savoir le cloud computing. Nous effectuerons également une recherche bibliographique succincte sur ce paradigme, en nous concentrant plus particulièrement sur le modèle de service SaaS. Dans le deuxième chapitre,nous allons présenté une vue d'ensemble de l'architecture multi-tenant, en décrivant sadéfinition, ses avantages et ses inconvénients. De plus, nous allons examiné les différents niveaux de maturité SaaS en relation avec la multi-location, mettant en évidence l'importance cruciale de la multi-location pour créer des applications SaaS. Dans le dernier chapitre, nous allons fourni une description détaillée des fonctionnalités et du comportement de l'application proposée en utilisant le langage UML. De plus, nous allons présenté le processus de mise en œuvre qui comprend l'expression du langage de programmation utilisé, l'environnement de développement choisi, et enfin, nous allons illustré les résultats obtenus grâce à notre application. Et pour conclure, nous allons fourni une visualisation des résultats obtenus grâce à notre application basée sur le modèle d'abonnement.

Chapitre I :
L’infonuagique ” Aperçus général ”

Chapitre 1

L'infonuagique " Aperçus général "

1.1 Introduction

Indéniablement, la technologie de l'internet se développe d'une manière exponentielle depuis sa création. Actuellement, une nouvelle "tendance" a fait son apparition dans le monde de l'IT, il s'agit du Cloud Computing, offre des opportunités aux sociétés de réduire les coûts d'exploitation des logiciels par leurs utilisations directement en ligne. Dans ce chapitre nous allons étudier le Cloud Computing en précisant son architecture et ses services.

1.2 Les paradigmes informatiques avant et après virtualisation

1.2.1 La virtualisation

La virtualisation a été l'un des principaux moteurs de transformation du secteur des technologies de l'information au cours des 50 dernières années. Avant l'avènement de la virtualisation, les entreprises et les organisations utilisaient des serveurs physiques distincts pour chaque application ou service. Cette approche était coûteuse, inefficace . Avec l'introduction de la virtualisation, il est devenu possible de créer des machines virtuelles (VM) sur une seule machine physique, en divisant ses ressources informatiques en plusieurs instances isolées. La virtualisation a permis de passer de systèmes non virtuels tels que le grid computing et le cluster computing à la virtualisation, ce qui s'est

traduit par des avantages significatifs tels que la réduction des coûts, l'augmentation de la flexibilité et de la disponibilité des ressources informatiques. Elle a également facilité l'adoption de nouveaux paradigmes tels que l'IOT (internet of thing), Fog Computing, l'Edge Computing et le Cloud Computing.
La Figure 1.1 présente des paradigmes informatiques avant et après la virtualisation

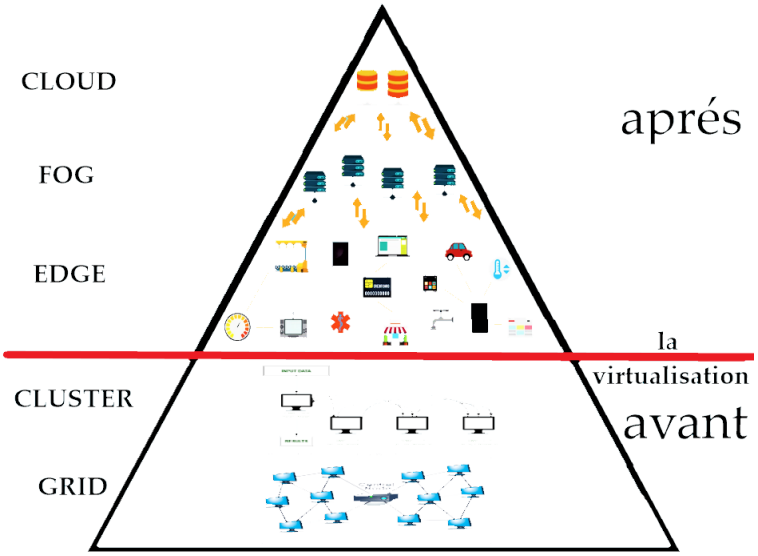


FIGURE 1.1 – Les paradigmes informatiques avant et après La virtualisation . [2]

1.3 Définition

L'informatique en nuage est un modèle de prestation de services de technologie de l'information dans lequel les ressources sont fournies sur Internet selon les besoins, plutôt que d'être fournies à partir d'un serveur local ou d'un appareil personnel. Il permet aux utilisateurs d'accéder et d'utiliser des ressources informatiques partagées, telles que des serveurs, du stockage et des applications, sur Internet.

Dans le cloud computing, les utilisateurs peuvent accéder aux services et aux ressources via un navigateur Web ou une interface de programmation d'application (API). Le fournisseur des services de cloud computing gère l'infrastructure et les ressources, et les utilisateurs ne paient que pour les services qu'ils utilisent. Cela permet aux entre-

prises et aux particuliers d'accéder à des ressources informatiques puissantes et évolutives sans avoir à investir dans et à entretenir leur propre matériel et infrastructure.

Dans l'ensemble, le cloud computing permet aux utilisateurs d'accéder et d'utiliser des ressources informatiques à la demande, plutôt que d'avoir à les acheter, les installer et les entretenir localement. Il est devenu un choix populaire pour les entreprises et les particuliers qui cherchent à réduire leurs coûts, à accroître leur efficacité et à faire évoluer leurs opérations. [3]

FIGURE 1.2 montre une représentation visuelle du cloud computing

FIGURE 1.2 – Le Cloud Computing .
[4]

Selon la définition du National Institute and Technology (National Institute of Standards and Technology (NIST)), définit le Cloud comme un modèle qui permet un accès omniprésent, pratique à la demande à un réseau partagé et à un ensemble de ressources informatiques configurables, à titre d'exemple des réseaux, des serveurs, du stockage, des applications et des services, qui peuvent être provisionnées et libérées avec un minimum d'administration. [5]

Ce modèle est composé de :

• Cinq caractéristiques essentielles : Libre-service à la demande, accès au réseau large, la mise en commun des ressources, élasticité rapide, service mesuré.

• Trois modèles de services : (Software as a Service (SaaS)), (Platform as a Service (PaaS)), (Infrastructure as a Service (IaaS)).

• Quatre modèles de déploiement (Cloud privé, Cloud communautaire, Cloud public, Cloud hybride.

Une autre définition du NIST [1] : « L'informatique dans le nuage est une nouvelle façon de délivrer les ressources informatiques, et non une nouvelle technologie » [5].

Selon Génération NT [2]« Le Cloud Computing est un concept d'organisation informatique qui place l'Internet au cœur de l'activité des entreprises, il permet d'utiliser des ressources matérielles distantes pour créer des services accessibles en ligne ».

Pour Commercial & Industrial Security Corporation (CISCO) [3] le Cloud Computing est une plateforme de mutualisation informatique fournissant aux entreprises des services à la demande avec l'illusion d'une infinité de ressources. Alors, le Cloud Computing est un concept qui consiste à transférer des fichiers ou des bases de données sur des serveurs à distance, qui étaient auparavant stockés dans la machine du client. Il permet d'accéder sur demande aux mêmes informations par plusieurs personnes .[9]

Selon Rajkumar Buyya [4], le Cloud Computing est un système parallèle et distribué d'un ensemble d'ordinateurs interconnectés et virtualisés, qui sont dynamiquement provisionnés et qui sont présentés comme une ressource ou plusieurs ressources informatique unifiées basées sur des accords SLA (service Level Agreementt) «établis par voie de négociation entre les fournisseurs de services et les consommateurs .[11]

Alors, le Cloud Computing est un concept qui consiste à transférer des fichiers ou des bases de données sur des serveurs à distance, qui étaient auparavant stockés dans les machines du client .Il permet d'accéder sur demande aux mêmes informations par plusieurs personnes.

De manière plus simple, le Cloud Computing est un style d'informatique ou les ressources sont faciles à obtenir et faciles d'accès, simple à utiliser, bon marché et tout simplement fonctionnelles. Le Cloud est donc un ensemble de matériel, de raccordements réseau et de logiciels qui fournit des services sophistiqués que les individus et les collectivités peuvent exploiter à volonté depuis n'importe où dans le monde.

1. Le NIST (National Institute of Standards and Technology) est une agence gouvernementale non réglementaire. Le NIST a été fondé en 1901 et fait maintenant partie du département américain du Commerce, et se consacre à l'avancement de divers secteurs industriels.[6]

2. Le lancement de Génération-NT.com remonte à août 2001, lorsque Bruno et Ray, ses fondateurs, ont décidé de créer une plateforme en ligne dans le but d'informer et d'assister les internautes qui cherchent des conseils et des ressources.les numeriques [7]

3. Cisco est une entreprise multinationale spécialisée dans les technologies de l'information et des communications. Fondée en 1984, Cisco est reconnue comme l'un des principaux fournisseurs mondiaux de solutions réseau et de communication[8].

4. Rajkumar Buyya est un universitaire australien d'origine indienne. En 2022, il occupe le poste de professeur distingué et directeur du laboratoire de Cloud Computing et de systèmes distribués à l'Université de Melbourne [10].

1.4 Principe du Cloud computing

Le cloud computing est un système qui permet de stocker et d'accéder à des données et des applications via internet. Il repose sur un grand groupe d'ordinateurs interconnectés appelé "cloud". Les applications et les données fournies par le cloud sont disponibles à un large groupe d'utilisateurs, d'entreprises et de plates-formes. L'accès se fait via internet et pour l'utilisateur, la technologie et l'infrastructure derrière le cloud est invisibles. Le cloud computing facilite la collaboration de groupe et est utilisé pour partager des photos, coordonner des bénévoles ou gérer des projets dans une grande entreprise .[12]

1.5 Les avantages et les inconvénients du Cloud Computing

Le tableau 1.1 ci-dessus présente les avantages et les inconvénients du cloud computing :

Les avantages	Les inconvénients
• Ordinateurs moins coûteux pour les utilisateurs. • Moins de problèmes de maintenance avec des performances améliorées. • réduction des Coûts de l'infrastructure informatique et des logiciels. • Mises à jour instantanées des logiciels. • Augmenter la Puissance de calcul et la sécurité des données. • Capacité de stockage illimitée. • Compatibilité améliorée entre les systèmes d'exploitation et entre les formats de documents. • Collaboration de groupe plus facile.	• Nécessite une connexion Internet constante. • Ne fonctionne pas bien avec des connexions à faible vitesse Peut être lent. • Les fonctionnalités peuvent être limitées. • Les données stockées peuvent ne pas être sécurisées. • Si le cloud perd vos données, vous êtes foutu.

TABLE 1.1 – Les avantages et les inconvénients du cloud computing .

[12]

1.6 Les déférents modèles du Cloud Computing

Dans le domaine du Cloud Computing, il existe deux types de modèles importants : les modèles de déploiement et les modèles de services. Voici une explication de chacun de ces modèles :

1.6.1 Les modèles de déploiement

Selon la définition du Cloud Computing données par le NIST il existe quatre modèles de déploiement des services du Cloud, à savoir : Cloud privé, Cloud communautaire, Cloud public et Cloud hybride.
La figure 1.3 illustre visuellement ces quatre modèles de déploiement du Cloud Computing.

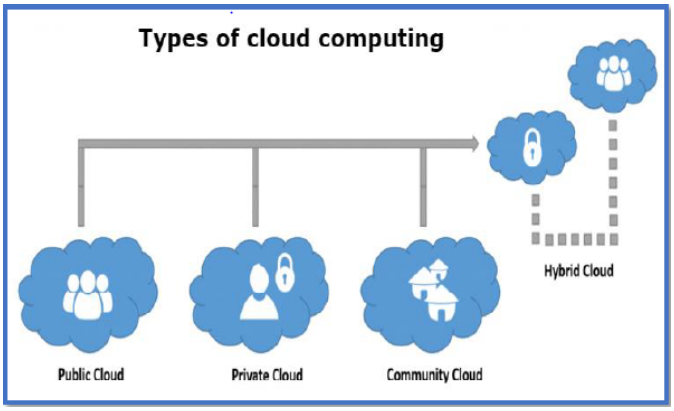


FIGURE 1.3 – Les déférents modèles du déploiement du Cloud .
[13]

1.6.1.1 Cloud public (Public Cloud Computing)

Un cloud public est une plateforme qui utilise le modèle standard de l'informatique en nuage pour rendre les ressources, telles que des machines virtuelles, des applications ou du stockage, disponibles aux utilisateurs à distance. Les services de cloud public peuvent être gratuits ou proposés via une variété de schémas de tarification par abonnement ou à la demande, y compris un modèle de paiement à l'utilisation .[13]
La figure 1.4 représente le concept de Cloud public.

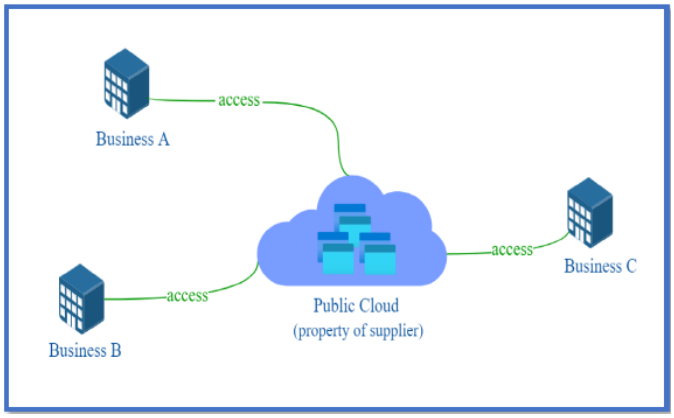


FIGURE 1.4 – Cloud public .
[14]

1.6.1.2 Cloud privé (Private Cloud Computing)

L'exploitation d'un cloud privé se fait au sein du centre de données interne d'une organisation. Le principal avantage est qu'il est plus facile de gérer la sécurité, la maintenance et les mises à jour, et qu'il permet également de contrôler le déploiement et l'utilisation. Contrairement au cloud public où toutes les ressources et les applications sont gérées par les fournisseurs de services, dans le cloud privé, ces services sont regroupés et mis à disposition des utilisateurs au niveau organisationnel. Ces ressources et applications sont gérées par l'organisation elle-même. Comme seuls les utilisateurs de l'organisation ont accès au cloud privé, la sécurité est renforcée .[13]
La figure 1.5 représente le concept de Cloud privé.

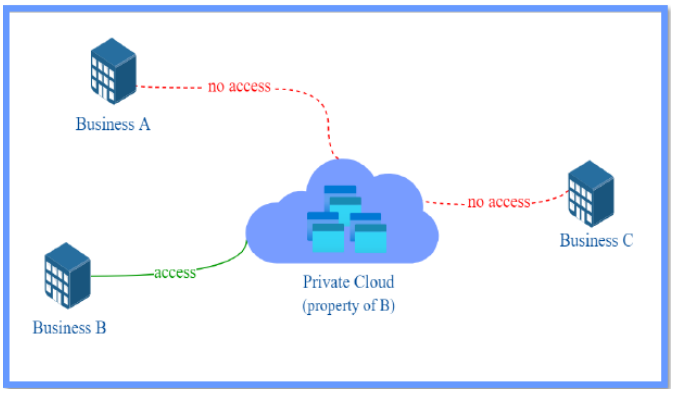


FIGURE 1.5 – Cloud privé .
[14]

1.6.1.3-Cloud communautaire (Community Cloud)

Le modèle de cloud communautaire permet à plusieurs organisations liées de partager et de gérer l'environnement de cloud computing. Lorsque plusieurs organisations construisent et partagent l'infrastructure cloud tout en respectant leurs politiques et exigences, ce modèle est appelé cloud communautaire. L'infrastructure cloud est hébergée par un fournisseur tiers ou au sein de l'une des organisations de la communauté .[13]
La figure 1.6 représente le concept de Cloud communautaire.

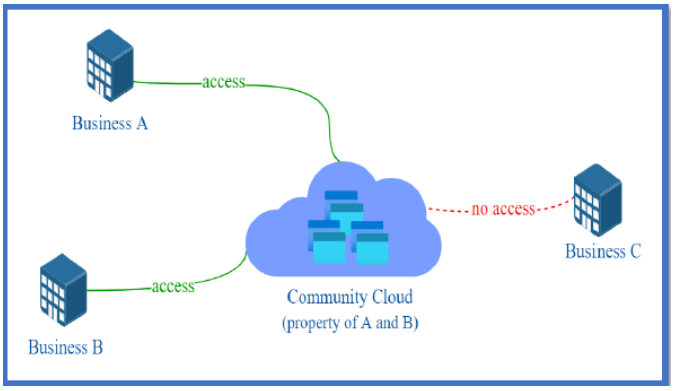


FIGURE 1.6 – Cloud communautaire .
[14]

1.6.1.4 Cloud hybride (Hybrid Cloud)

Ce modèle est composé à la fois de modèles de cloud public et privé, où l'environnement de cloud computing est hébergé et géré par des tiers, mais certaines ressources dédiées sont utilisées exclusivement par une organisation. Dans ce modèle, un cloud privé est lié à un ou plusieurs services cloud externes. C'est une manière plus sûre de contrôler les données et les applications et permet à l'utilisateur d'accéder aux informations via internet. Il permet à l'organisation de répondre à ses besoins dans le cloud privé et si des besoins occasionnels se présentent, elle demande au cloud public des ressources de calcul intensif .[13]
La figure 1.7 représente le concept de Cloud hybride.

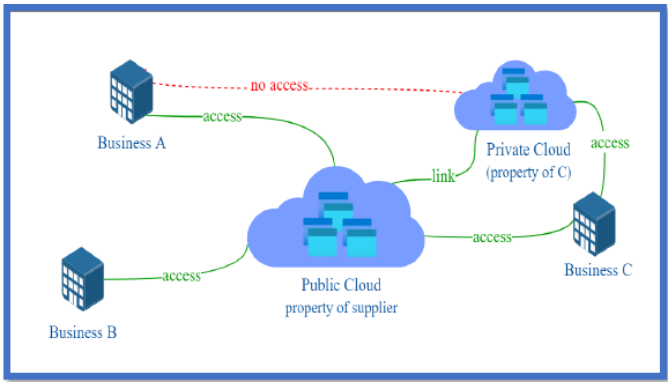


FIGURE 1.7 – Cloud hybride .
[14]

1.6.2 Les modèles de services du Cloud Computing

Le Cloud Computing est composé de plusieurs catégories de services, les trois principales catégories sont :(IaaS), (PaaS) et (SaaS) comme illustré dans la figure 1.8.

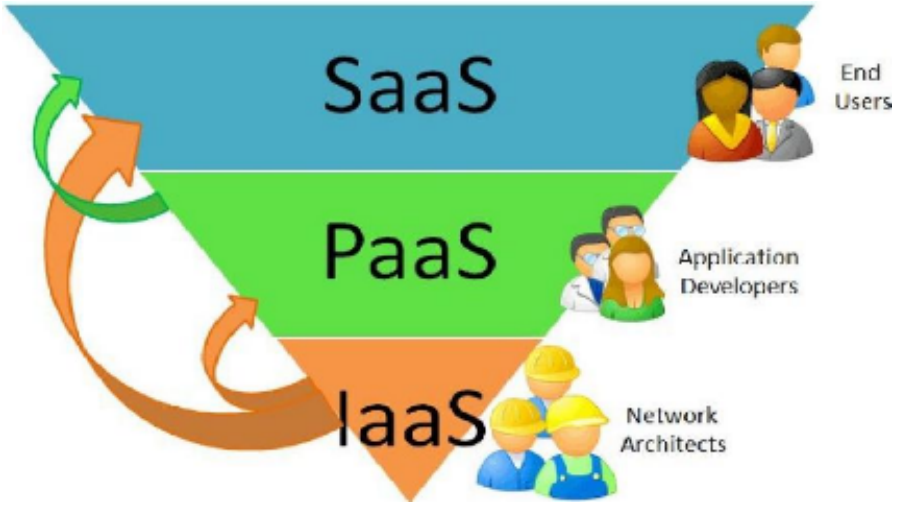


FIGURE 1.8 – Les couches du Cloud Computing .
[15]

1.6.2.1 Software as a Service(SaaS)

Le Software as a Service est un applicatif développé et déployé dans une plateforme en tant que service. C'est une application souple qui est accessible uniquement par le biais d'Internet et qui est le plus souvent facturée à l'utilisation au client final .[15]

Les exemples de SaaS incluent les logiciels de gestion de la relation client, les logiciels de planification des ressources d'entreprise et les outils de productivité basés sur le cloud tels que Google Workspace.

1.6.2.2 Platform as a Service(PaaS)

La plateforme as a Service est un ensemble de composants de base qui reposent sur l'architecture souple offerte par l'IaaS. Elle permet aux développeurs d'application d'avoir une plateforme de travail souple, distribuée et virtualisée dans laquelle ils n'ont besoin d'aucune compétence dans les domaines sous-jacents (réseau, matériel, système d'exploitation, etc.). On peut la voir comme une abstraction de la technologie utilisée pour distribuer les calculs sur plusieurs machines au sein d'un réseau.[15]

Quelques exemples de fournisseurs PaaS incluent AWS Elastic Beanstalk, Google App Engine et Microsoft Azure.

1.6.2.3 Infrastructure as a Service(IaaS)

L'Infrastructure as a Service offre une base matérielle (hardware) aux plateformes en tant que service. Ces infrastructures sont mises en place et gérées exclusivement par des architectes réseau, ce qui permet de garder un bon niveau d'expertise et déviter de mélanger les domaines de compétence. Elle se compose d'équipements réseau et de serveurs, la plupart du temps complètement virtualisés .[15]

Exemples des fournisseurs laaS : Amazon EC2, VPC, IBM Blue Cloud, Eucalyptus, FlexiScale, Joyent, Rackspace Cloud, etc.

1.6.2.4- Les avantages et les inconvénients des services Cloud

Le Tableau 1.2 présente les avantages et les inconvénients du Cloud Computing :

	Avantage	Inconvénient
SaaS	-pas d'installation -plus de licence -migration	-logiciel limité -sécurité -dépendance des prestataires
PaaS	-pas d'infrastructure nécessaire -pas d'installation -environnement hétérogène	-limitation des langages -pas de personnalisation dans la configuration des machines virtuelles
IaaS	-administration -personnalisation -flexibilité d'utilisation	-sécurité -besoin d'un administrateur système

TABLE 1.2 – Les avantages et les inconvénients des services Cloud . [16]

1.6.2.5 la différence entre SaaS et PaaS et IaaS

Les termes SaaS, PaaS et IaaS font référence à différents modèles de services cloud offerts par les fournisseurs de services cloud.
•IaaS est là pour vous fournir une flexibilité maximale en matière d'hébergement d'applications personnalisées, ainsi qu'un data center pour le stockage de données.
•PaaS est le plus souvent basé sur une plate-forme IaaS afin de réduire les besoins en administration système. Cela vous permet de vous concentrer sur le développement d'applications plutôt que sur la gestion d'infrastructure.
•SaaS offre des solutions prêtes à l'emploi et qui répondent à un besoin commercial particulier (tel qu'un site Web ou un courrier électronique). La plupart des plateformes SaaS modernes sont construites sur des plateformes IaaS ou PaaS .[17]
La figure 1.9 représente la différence entre ces trois modèles :

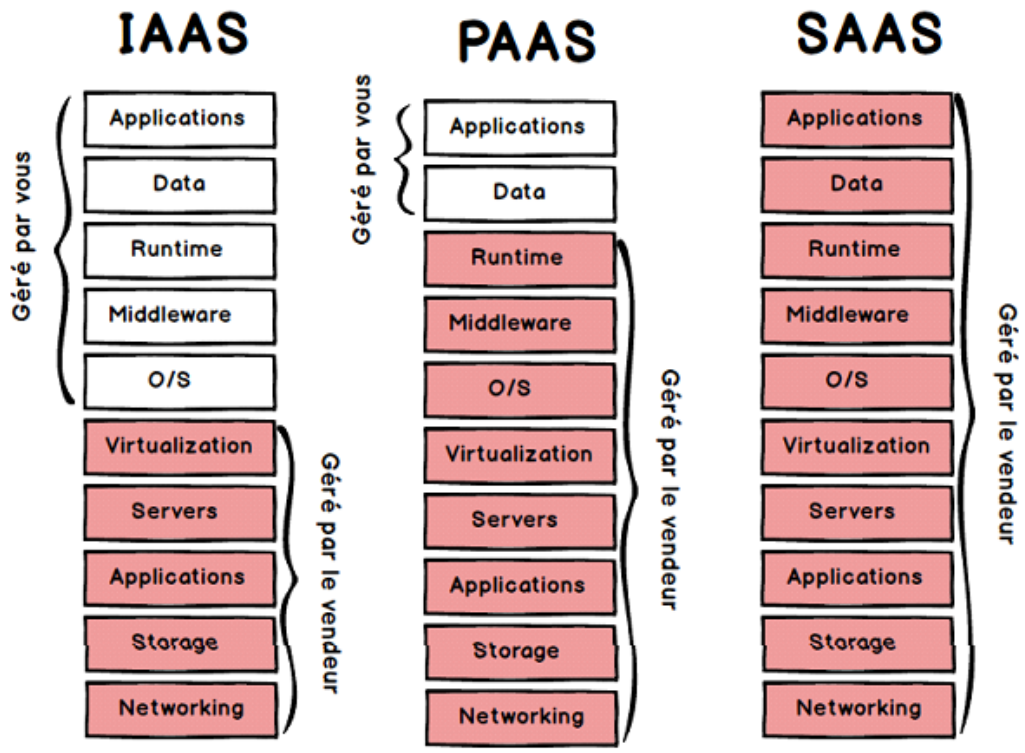


FIGURE 1.9 – La différence entre les différents modèles de services . [17]

1.6.3 Les fournisseurs du cloud Computing

Les fournisseurs de services cloud sont des entreprises qui proposent des solutions d'hébergement, de stockage et de gestion des données sur des serveurs distants accessibles via Internet.
La Figure 1.10 illustre les principaux fournisseurs de services cloud.

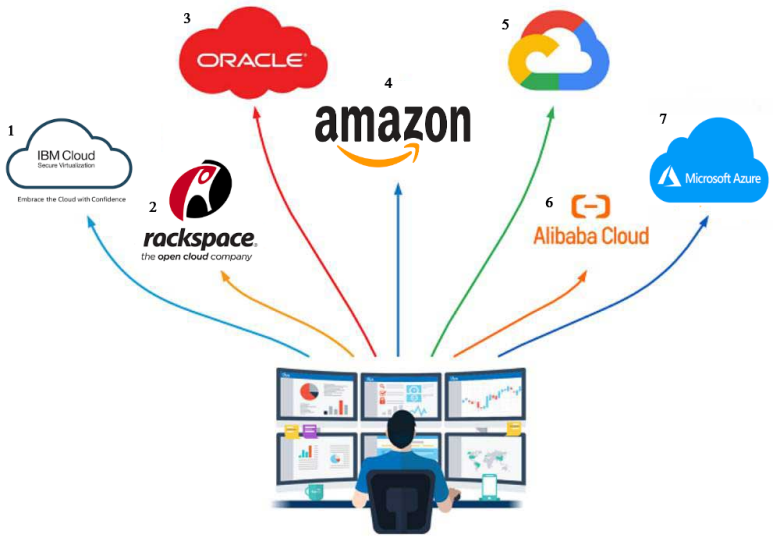


FIGURE 1.10 – Les fournisseurs du cloud Computing .
[18]

1.7 Les acteurs du Cloud Computing

Le marché du Cloud Computing est dominé par plusieurs acteurs majeurs qui proposents des services de Cloud public,privé et hybride.La figure FIGURE 1.11 présente quelques-uns des principaux acteurs du Cloud :

FIGURE 1.11 – Les acteurs du cloud Computing.

1.8 Conclusion

Depuis l'apparition de l'informatique classique, plusieurs systèmes ont vu le jour. Avec l'avènement de la technologie de stockage, cette dernière a donné un nouvel aspect à l'informatique moderne, celui du cloud computing. Parmi les modèles importants du cloud, on peut citer les modèles de service et les modèles de déploiement. Dans le modèle de service, nous nous intéressons particulièrement au modèle SaaS qui représente une part importante des services offerts dans le cloud.

Dans le prochain chapitre, nous nous concentrons sur le modèle de service SaaS, en mettant particulièrement l'accent sur l'une de ses caractéristiques : la multi location. Nous aborderons également les frameworks qui supportent cette caractéristique, ainsi que quelques articles traitant de ce sujet.

Chapitre II :
La Multilocation dans les modèles de service ”SaaS” : Étude Approfondie

Chapitre 2

La Multilocation dans les modèles de service ”SaaS” : Étude Approfondie

2.1 Introduction

L'architecture multi-tenant est une piste de recherche prometteuse dans le domaine du cloud computing, suscitant un intérêt constant. Cette approche permet le partage, selon différents niveaux de maturité, de ressources matérielles et logicielles entre plusieurs locataires ou utilisateurs du cloud, ce qui se traduit par une optimisation des bénéfices des applications cloud existantes.

Ce chapitre se concentre sur la présentation des concepts, des idées et de l'approche de l'architecture multi-tenant, ainsi que sur l'explication des principales solutions qui abordent cette problématique dynamique qui a récemment suscité beaucoup d'engouement.

2.2 Le modèle SaaS et ses caractéristiques

Le modèle SaaS (Software as a Service) représente une méthode de déploiement de logiciel offrant de nombreux avantages aux entreprises. Au lieu d'installer et d'exécuter des logiciels sur chaque ordinateur utilisateur, les applications sont hébergées en tant que service sur Internet. Les clients peuvent y accéder facilement via une connexion Internet, sans avoir à se soucier de la complexité liée au logiciel et au matériel. Les fournisseurs SaaS gèrent l'application et la rendent disponible à plusieurs clients grâce

à une architecture multilocation.

Cela permet aux fournisseurs de réaliser des économies de maintenance, tandis que les clients peuvent réduire leurs investissements initiaux. Le SaaS est considéré comme une évolution majeure dans la manière dont les logiciels sont conçus, vendus, achetés et utilisés . [19]

Parmi les caractéristiques essentielles du modèle SaaS, on peut identifier les trois caractéristiques présentées dans la figure 2.1 et qui sont expliquées en détail par la suite.

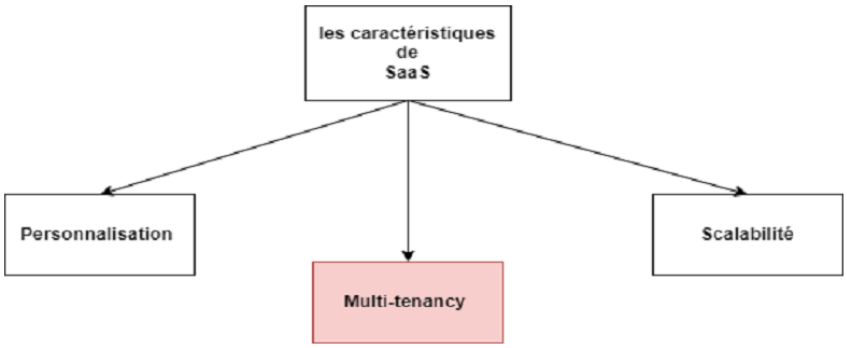


FIGURE 2.1 – Les caractéristiques de SaaS

a- La Personnalisation

La personnalisation, dans le cadre des applications SaaS, offre aux locataires (utilisateurs ou consommateurs) la possibilité d'adapter l'application SaaS à leurs besoins spécifiques. Cela peut impliquer la personnalisation de l'interface utilisateur, l'ajout ou la suppression des fonctionnalités, ainsi que la configuration de règles métier spécifiques.[20]

2.2.1.2 La Scalabilité

Elle représente une caractéristique essentielle pour les applications SaaS, cela signifie qu'elles sont capables de s'adapter à la croissance des besoins des utilisateurs [21].

2.2.1.3 La Multilocation

La multi-location est une caractéristique fondamentale du modèle SaaS, qui permet à plusieurs clients (ou locataires) d'utiliser une seule instance de l'application. Chaque client dispose d'un environnement isolé et sécurisé, avec ses propres données et paramètres de configuration[22].

2.3 La multilocation versus La location Unique :

Dans cette section, nous présenterons une brève comparaison entre deux approches de location : la location unique et la location multiple.

2.3.1 La location unique

L'approche de location unique est une architecture où une seule instance d'une application logicielle et de son infrastructure de support est dédiée à un seul client. Cette approche est couramment utilisée dans les modèles de fourniture de logiciels en tant que service (SaaS) et les services cloud. Les locataires bénéficient d'un niveau élevé de contrôle, de personnalisation, de fiabilité, de sécurité et de capacité de sauvegarde. Les clients potentiels choisissent cette architecture pour avoir un contrôle et une flexibilité accrus dans leur environnement, afin de répondre à des exigences spécifiques. La Figure 2.2 illustre l'architecture de la location unique. [23]
La Figure 2.2 illustre l'architecture de la location unique.

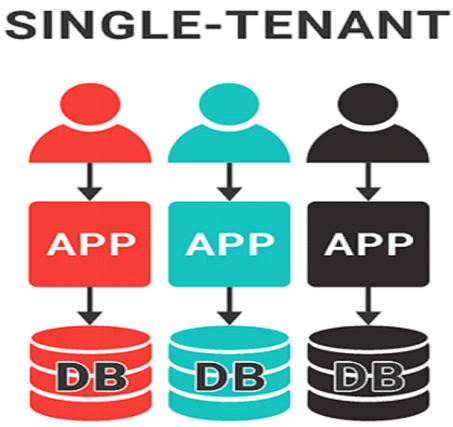


FIGURE 2.2 – Illustration de la location unique.
[24]

2.3.1.1 Les avantages et les inconvénients de l’architecture de location unique

L’approche mono-locataire ou de location unique présente plusieurs avantages qui en font une option à prendre en compte lors du choix de l’architecture d’un service. Le tableau 2.1 ci-dessous présente quelques-uns de ces avantages.

Les avantages	D’etails les avantages
Portabilité :	Les instances mono-locataires peuvent être facilement déployées sur différents services cloud ou SaaS. De plus, l’approche mono-locataire facilite la migration vers un autre environnement d’hébergement, le cas échéant.
Sécurité des données :	En cas de violation de données d’un locataire chez un fournisseur de services, les autres locataires sont protégés car leurs données sont stockées dans des instances distinctes. Cela garantit une meilleure sécurité des données pour chaque locataire.
Personnalisation des données :	Grâce à la séparation des données de chaque client, il est possible de personnaliser les instances de logiciels et de matériel à un degré élevé. Cela permet d’ ;adapter les fonctionnalités et les paramètres spécifiques à chaque client, offrant ainsi une plus grande flexibilité et une meilleure adaptation aux besoins individuels.
Restauration et récupération après sinistre :	La présence de sauvegardes isolées permet aux utilisateurs de restaurer rapidement leurs données en cas de sinistre ou de perte. Cette fonctionnalité garantit une récupération efficace des données et contribue à minimiser les pertes potentielles lors d’événements imprévus.
Fiabilité :	Les instances mono-locataires sont réputées pour leur fiabilité, car les performances sont basées sur une seule instance dédiée à un seul locataire, plutôt que sur plusieurs instances partagées entre différents locataires. Cela permet d’assurer une stabilité et une cohérence des performances pour chaque client, sans être impacté par les activités ou les besoins des autres locataires.

TABLE 2.1 – Les avantages de l’approche Mono-locataire.

[23]

Malgré les avantages potentiels de l’approche mono-locataire, il est important de noter qu’elle est moins couramment utilisée que d’autres architectures concurrentes. Cela peut être attribué à certains inconvénients spécifiques associés à cette approche. Le tableau 2.2 présente les inconvénients de l’approche mono-locataire :

Les inconvénients	D’etails Les inconvénients
Les coûts élevés :	sont l’un des inconvénients de l’approche mono-locataire. Cela est dû au temps de configuration nécessaire, à l’allocation de ressources, à la personnalisation et à la gestion de l’hébergement d’une seule instance SaaS par client, ce qui peut entraîner des dépenses significatives.
La configuration complexe :	un autre inconvénient de l’approche mono-locataire. Étant donné que le locataire est généralement responsable de la gestion d’un système locataire unique, les mises à jour, les ajustements et la gestion en général peuvent prendre plus de temps et nécessiter des compétences techniques avancées.
La gestion complexe :	Grâce à la séparation des données de chaque client, il est possible de personnaliser les instances de logiciels et de matériel à un degré élevé. Cela permet d’ ;adapter les fonctionnalités et les paramètres spécifiques à chaque client, offrant ainsi une plus grande flexibilité et une meilleure adaptation aux besoins individuels.

TABLE 2.2 – Les inconvénients de l’approche mono-locataire.

[23]

2.3.2 L’approche multi-locataire

Une architecture de partage d’une instance d’application entre plusieurs locataires, en leur fournissant chacun une partie dédiée de l’instance. Cette approche garantit que chaque locataire dispose d’un environnement isolé, offrant des performances optimales et une confidentialité des données. Elle est particulièrement adaptée lorsque différents groupes d’utilisateurs partagent la même application, mais nécessitent des restrictions de sécurité et de confidentialité spécifiques à leurs données. Il est important de ne pas confondre la multi-locataire avec la notion d’espace locataire, où les clients louent un espace prédéfini de ressources pour exécuter différentes instances d’application. De même, la multi-locataire ne doit pas être confondue avec le déploiement de plusieurs applications sur une même instance d’environnement d’exécution, appelé également multi-locataire par certains fournisseurs de PaaS. Certaines approches telles que le partage de centre de données, de virtualisation et de middleware peuvent être considérées comme des moyens d’atteindre un comportement multi-locataire, mais elles peuvent

manquer de partage efficace des ressources.[23]

La Figure 2.3 illustre l’architecture de la location multiple.

FIGURE 2.3 – Illustration de l’approche multi-locataire
[24]

A Les caractéristiques de la location multiple

Dans cette section, nous allons mentionner trois caractéristiques clés de l’approche multi-locataire.

- Partage des ressources matérielles : Dans un environnement de développement logiciel traditionnel avec une approche mono-locataire, chaque locataire dispose généralement de son propre serveur virtuel dédié. Cependant, en regroupant plusieurs locataires sur le même serveur, il est possible d’optimiser l’utilisation des ressources matérielles, ce qui entraîne une réduction des coûts globaux de l’application. La mise en œuvre de l’approche multi-locataire peut varier, mais la variante "pure multi-locataire" est préférée car elle permet d’accueillir un plus grand nombre de locataires sur le système sans compromettre les performances.

- Haut degré de configurabilité : Étant donné que tous les locataires partagent la même instance d’application, il est essentiel de pouvoir configurer et personnaliser l’application en fonction des besoins spécifiques de chaque locataire. Les options de configuration doivent être intégrées dès la conception du produit, afin de permettre des ajustements flexibles sans nécessiter de modifications profondes de l’architecture sous-

jacente. Cela permet aux locataires de bénéficier d'une application adaptée à leurs exigences individuelles, tout en maintenant l'intégrité de l'instance partagée.

- Partage d'instances d'application et de bases de données : Dans un environnement mono-locataire, chaque locataire dispose généralement de ses propres instances d'application et de bases de données dédiées. En revanche, dans un environnement multi-locataire, moins d'instances d'application et de bases de données sont nécessaires, ce qui facilite le déploiement et la gestion des ressources. Ce partage permet une utilisation plus efficace des ressources, réduit les coûts d'infrastructure et simplifie la collecte et l'analyse des données pour améliorer l'expérience utilisateur globale. [25]

B Les avantages

Voici quelques-uns des principaux avantages de l'approche de la location multiple [25] :
- Utilisation plus efficace des ressources matérielles.
- Maintenance de l'application plus facile et moins coûteuse.
- Réduction des coûts globaux, ce qui permet d'offrir un service à un prix inférieur à celui des concurrents.
- Création de nouvelles opportunités d'agrégation de données.

B Les Défis

Parmi les défis importants de l'approche multi-locataire[25], on peut mentionner les suivants :
-Sécurité : Les données des différents clients peuvent être compromises ou mélangées si les mesures de sécurité ne sont pas suffisamment strictes.
-Performances : Le partage des ressources peut entraîner des problèmes de performance pour les utilisateurs lorsque les charges de travail sont élevées.
-Coûts : Les coûts associés aux solutions multi-locataires peuvent être plus élevés que ceux des solutions dédiées.
-Maintenance : La gestion d'un système multi-locataire peut représenter un défi tout au long du cycle de vie d'un logiciel. Cela implique notamment l'adaptation du système aux exigences changeantes et son déploiement ultérieur.
-Évolutivité : Il peut être difficile de faire évoluer les solutions multi-locataires pour répondre aux besoins croissants des clients.

2.4 Les approches de réalisation de l’architecture multi-Locataire

Avec la croissance de la popularité de l’architecture multi-locataire dans les applications SaaS fournies via le Cloud, la gestion de ce type d’application devient de plus en plus cruciale. Deux approches principales de gestion de la multi-locataire dans le Cloud ont été identifiées, à savoir l’approche "Fonctionnelle" et l’approche "Base de données". Selon l’auteur[26], Oracle se concentre principalement sur l’approche de la multi-locataire du point de vue de la base de données, tandis que Microsoft adopte davantage une approche fonctionnelle. Une classification des approches multi-locataires de SaaS est présentée dans la Figure 2.4.

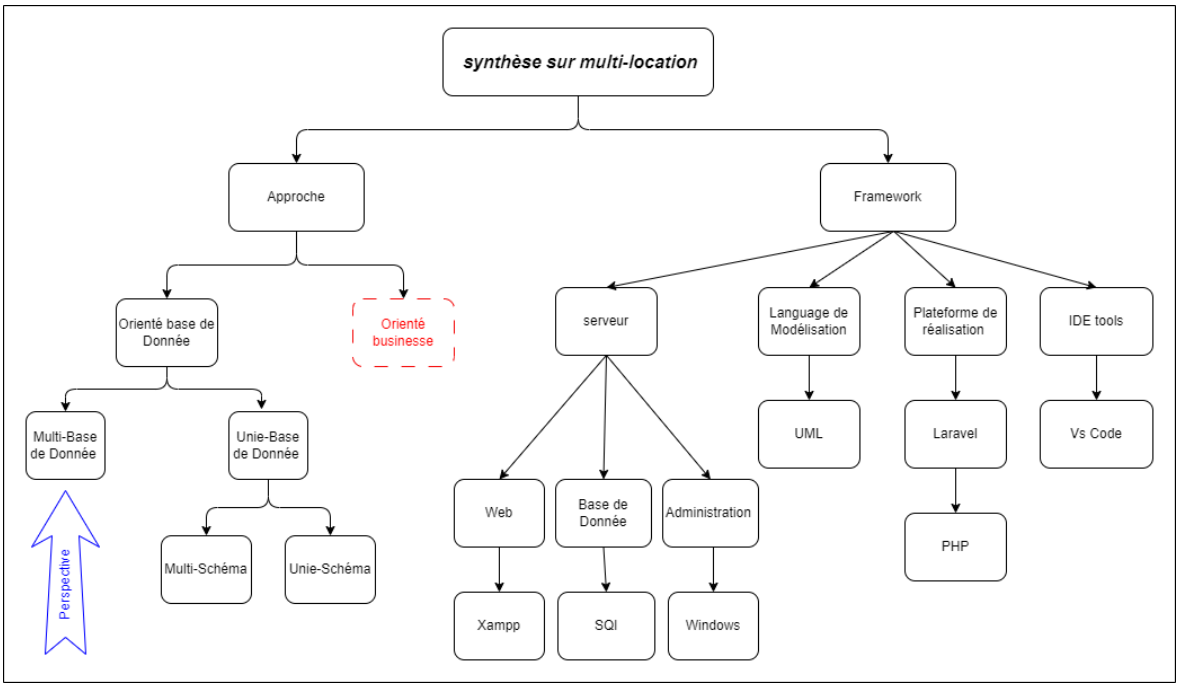


FIGURE 2.4 – Classification des approches multi-locataires de SaaS.

Ce mémoire porte sur la perspective orientée bases de données dans le domaine de la recherche et du développement/gestion d’applications Cloud multi-tenant pour les applications SaaS. Cette approche a suscité un intérêt considérable chez de nombreux chercheurs dans ce domaine dynamique. La perspective nommée base de données multi-locataires met l’accent sur la fonctionnalité que la base de données fournirait à plusieurs

locataires, leur permettant de créer, de stocker et d'accéder à leurs bases de données via Internet. Intérieurement, il est crucial que la base de données soit hautement paramétrable et sécurisée afin de répondre aux attentes des locataires et à leurs différents besoins métiers. Il est important de noter que la perspective fonctionnelle diffère de la perspective centrée sur les données. La perspective fonctionnelle met l'accent sur la gestion et le contrôle des processus métier qui produisent des services [27] . Nous adoptons dans ce travail la distinction trouvée dans [27] lors de l'étude des approches multi-locataire avec une vue base de données. Dans cette vision, on distingue deux grandes approches :

2.4.1 L'approche orienté base de données

Dans cette tendance, on distingue deux grandes approches :

2.4.1.1 Base de données unique

a - Base de données unique schéma unique

Dans cette approche, les locataires utilisent la même base de données et le même ensemble de tables. Une table peut contenir des enregistrements provenant de plusieurs locataires identifiés par l'ID du locataire. Parmi les trois approches, celle-ci est la moins coûteuse, mais elle est également la moins sécurisée et présente le même problème de sauvegarde des données que la deuxième approche. [28]
La figure 2.5 présente une architecture de base de données unique, schéma unique.

Tenant ID	Cust Name	Address
4711	Tenant ID	Produc ID	Product Name
132	4711	Tenant ID	Shipment	Date
680	132	4711	324365	2006-02-21
4711	680	132	115468	2006-04-06
	4711	680	654109	2006-03-27
		4711	324956	2006-02-23

FIGURE 2.5 – Base de données unique, schéma unique.
[29]

Avantages :

Le coût de maintenance et d'achat du troisième schéma est le plus bas, et il permet à chaque base de données de prendre en charge le plus grand nombre de locataires .[30]

Inconvénients :

Le niveau d'isolation le plus bas, la sécurité minimale, la conception et le développement sont nécessaires pour augmenter le niveau de sécurité ; il est le plus difficile de sauvegarder et de restaurer les données, car cela doit être fait table par table. Si vous souhaitez utiliser le moins de serveurs possible pour fournir des services à un maximum de locataires, et si les locataires acceptent une moindre isolation en échange d'une réduction des coûts, alors ce schéma est le plus adapté .[30]

b - Base de données unique, schéma multiple

Plusieurs locataires partagent la même base de données, mais chacun a son propre schéma qui est un ensemble de tables. Cette approche est facile à mettre en œuvre mais présente les mêmes avantages que l'approche précédente. Cependant, l'inconvénient est que les données des locataires ne peuvent pas être récupérées en cas de défaillance car la sauvegarde est effectuée au niveau de la base de données et la récupération à partir de la sauvegarde implique d'écraser toutes les données de tous les locataires alors qu'au départ seul un locataire était concerné [28].

La figure 2.6 présente une architecture de base de donnée unique schéma multiple.

Avantages :

Chaque base de données peut prendre en charge un plus grand nombre de locataires [30].

Inconvénients :

En cas de défaillance, la récupération des données est plus difficile car la base de données implique les données d'autres locataires. Par conséquent, il est nécessaire de prendre en compte les statistiques entre locataires, ce qui peut entraîner certaines difficultés .[30]

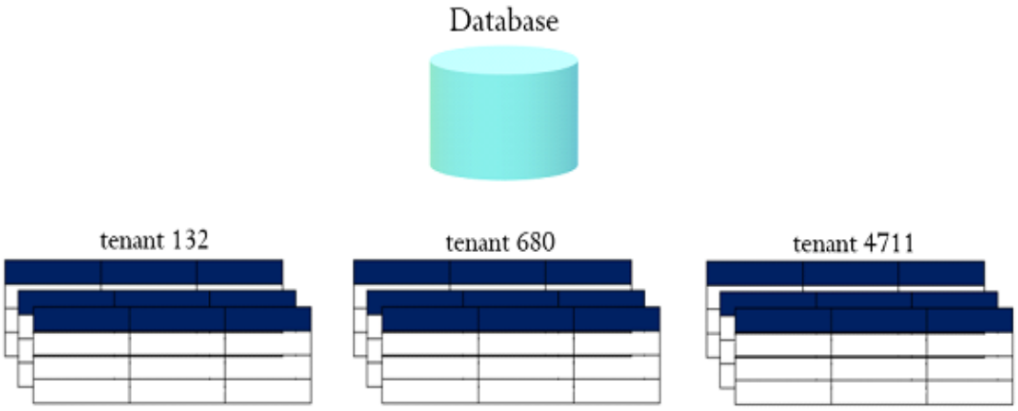


FIGURE 2.6 – Base de données unique schéma multiple.
[29]

2.4.1.2 Base de données multiple

Dans cette approche, chaque locataire dispose de sa propre base de données dans laquelle ses données sont stockées, mais il partage les ressources de calcul et le code d'application avec d'autres locataires.[28]

La figure 2.7 présente l'architecture de multi base dec données.

FIGURE 2.7 – Architecture Multi base de données .
[**?**]

Avantages

Chaque locataire dispose d'une base de données indépendante, ce qui facilite l'expansion du modèle de données et répond aux besoins uniques des différents locataires. En cas de problème, il est relativement simple de restaurer les données .[30]

Inconvénients

Avec l'augmentation du nombre de bases de données d'installation, les coûts de maince et d'achat sont plus élevés. Ce schéma ressemble à celui traditionnel où un client, un ensemble de données et un ensemble de déploiements sont utilisés, la seule différence étant que le logiciel est uniformément géré par les opérateurs.[30]

2.5 Les Framework qui supporte la multilocation

Dans cette partie, nous allons citer quelques cadres utilisés pour implémenter la location multiple.

Django

Django est un Framework web open-source en Python qui permet aux développeurs de créer rapidement des applications web complètes Il offre la possibilité de supporter l'abonnement grâce aux packages Django payments , Djstripe et Django-oscar.[31]

Express

Express est un Framework léger pour les applications web basées sur Node.js . Il offre la possibilité de supporter l'abonnement grâce aux packages Stripe , Braintree et PayPal .[32]

Spring

Spring est un cadre de développement Java conçu pour les applications d'entreprise .Il offre la possibilité de supporter l'abonnement grâce aux packages SpringPayflow Pro et Spring Commerce Tools.[33]

Ruby on Rails

Ruby on Rails est un Framework open-source utilisant le langage de programmation Ruby pour le développement d'applications Web . Il offre la possibilité de supporter l'abonnement grâce aux packages Stripe , Braintree et PayPal.[34]

ASP.NET

ASP.NET est un Framework de développement web développé par Microsoft. Il permet aux développeurs de créer des applications web interactives et riches en utilisant .NET et des langages de programmation comme C ou VB.NET. ASP.NET offre un modèle de programmation puissant. Il offre la possibilité de supporter l'abonnement grâce aux packages Stripe.NET , PayPal SDK pour.Net et Braintree SDK pour .NET.[35]

Laravel

Laravel est un Framework PHP open-source conçu pour le développement d'applications web. Il offre la possibilité de supporter l'abonnement grâce aux packages Paddle, Stripe et PayPal.[36]

Ce tableau 2.3 présente des informations sur différents frameworks, Ces informations peuvent être utiles pour choisir le Framework le mieux adapté aux besoins d'une application multi-locataire :

Framework	Langage	Support SaaS	Outils de multi-locataire	Unie base de donnée ou multiple	outils facturation	Packages qui support payment de l'abonnement
Django	Python	Oui	Oui	Unique et multiple	Oui	-Django payments -Djstripe -Django-oscar
Express	Node.js	Oui	Oui	Unique et multiple	Oui	-Stripe -Braintree -Paypal
Spring	Java	Oui	Oui	Unique et multiple	Oui	-Spring Payflow Pro -Spring Commerce Tools
Ruby on Rails	Ruby	Oui	Oui	Unique et multiple	Oui	-Stripe -Braintree -Paypal
ASP.NET	C#	Oui	Oui	Unique et multiple	Oui	-Stripe.NET -Paypal SDK pour.NET -Braintree SDK pour.NET
Laravel	Php	Oui	Oui	Unique et multiple	Oui	-Laravel cashier -Laravel stripe -Laravel paypal

TABLE 2.3 – Tableau des frameworks pour applications multi-locataires

2.6 Travaux Connexes Migration vers le modèle SaaS Multilocation

Des recherches antérieures ont étudié un cadre permettant de guider la conversion d'applications Web traditionnelles vers le paradigme SaaS multi-tenant. Dans ce mémoire, nous nous appuyons sur deux publications majeures dans le domaine du Cloud SaaS, ayant reçu un grand nombre de citations.

Saleh et al.[37]ont proposé un cadre de migration qui intègre des éléments tels que la personnalisation de l'interface utilisateur et du flux de travail, la logique métier et la configuration de la base de données. Bezemer et al.[38]ont développé des modèles

de migration d'applications prenant en compte le partage de ressources matérielles, un haut degré de configurabilité et une instance de base de données partagée. Ces cadres comprennent trois composants essentiels :

1. Un module d'authentification pour associer les informations d'identification des utilisateurs finaux aux locataires,

2. Un module de configuration pour gérer les paramètres spécifiques aux locataires.

3. Un module de base de données permettant d'adapter, d'insérer, de modifier et d'interroger les données propres à chaque locataire.

2.7 Conclusion

Ce chapitre examine en détail l'architecture multi-locataire, en mettant en évidence les différentes solutions et approches de cette vision architecturale prometteuse, ainsi que les différents niveaux de maturité associés. Il souligne également l'importance cruciale de la multi-location pour la création de systèmes de qualité. L'objectif du chapitre suivant est de maximiser les efforts pour concevoir et réaliser un système multi-locataire basé sur l'abonnement en utilisant l'approche orientée vers une seule base de données.acement les données des utilisateurs et leur offrir une expérience utilisateur uniforme.

Chapitre III :
Le Système d'Abonnement Multi-locataire " Conception et réalisation "

Chapitre 3

Le Système d'Abonnement Multi-locataire : Conception et réalisation

3.1 Introduction

La croissance de l'utilisation du cloud computing et du modèle SaaS dans le domaine informatique est indéniable. Cependant, la mise en place d'une architecture multi-location avec une base de données unique/multiple est un défi complexe, tant sur le plan de la conception que de la programmation Pour relever ce défi, différents Framework et bibliothèques ont été proposés pour faciliter la réalisation de modèles SaaS multi-locataires avec la fonctionnalité d'abonnement. Dans ce chapitre, nous examinerons en détail l'approche adoptée pour rendre notre système typique compatible avec la multi-location et l'abonnement. Nous mettrons également l'accent sur les outils matériels et logiciels utilisés, en mettant en avant les différentes interfaces de notre système final appelé Mt-Fact.

3.2 Les outils d'implémentation et de réalisation

3.2.1 Environnement matériel

Nous avons mis en œuvre le system de ce mémoire en utilisant un ordinateur "Toshiba" avec une configuration détaillée dans le tableau suivant 3.1 :

Processeur	i3-4005U CPU @ 1.70GHz 1.70 GHz
Disque dur	500 GB HDD
Mémoire vive	6,00 Go
Système opérateur	Windows 10
Carte graphique	Intel HD 4000 – 1 go

TABLE 3.1 – Les caractéristiques de l'ordinateur

3.2.2 Environnement logiciel

Le développement d'un système peut être complexe en raison de la multitude de langages de programmation et de Framework disponibles. Le choix de la technologie à utiliser peut être coûteux en termes de temps et d'argent. Nous avons également été confrontés à ce défi, car l'accès aux plateformes payantes telles qu'Azure, Google Cloud ou SAP nous était limité en raison de l'absence d'une carte de payement électronique. "Carte Visa" . Cependant, après des recherches approfondies, nous avons découvert le Framework Laravel, qui est gratuit, et il nous a permis de développer notre système.

• Laravel

Développé par Taylor Otwell en 2011, Laravel est un Framework PHP largement reconnu pour son statut open source sous licence MIT (Massachusetts Institute of Technology). Il a gagné en popularité et est désormais considéré comme le Framework PHP le plus populaire. Cette popularité est en grande partie attribuée à sa communauté d'utilisateurs active et à sa documentation de qualité. Pour le projet présenté dans ce mémoire, nous avons utilisé la version 10.10.1 de Laravel. [39]

Pourquoi avons-nous choisi Laravel ?

Selon son fondateur, Laravel est considéré comme le Framework le plus puissant de l'écosystème PHP, en raison de son intégration des fonctionnalités nécessaires à la construction de systèmes web modernes. Il se distingue par son élégance, sa propreté et sa syntaxe raffinée, ce qui facilite la création de systèmes de grande envergure. Parmi les avantages de Laravel, on peut citer :
a-L'utilisation des dernières fonctionnalités de PHP, ce qui permet de bénéficier des améliorations et des performances de la dernière version du langage.
b-Une excellente documentation, avec des explications détaillées sur le style de codage,

les méthodes et les classes, ce qui facilite l'apprentissage et le développement.
c-Une intégration transparente avec des services de messagerie, offrant ainsi des fonctionnalités étendues pour la gestion des communications et des notifications.
d-Présence d'un outil en ligne de commande intégré appelé Artisan : cet outil facilite la création de code de base et la gestion efficace du système de base de données. Artisan permet de générer les fichiers MVC de base et de gérer les ressources, y compris leurs configurations respectives.
e- Prise en charge des caches populaires : dès l'installation, Laravel prend en charge des caches tels que BootStrap et Redis. De plus, vous pouvez configurer plusieurs configurations de cache.
f-Support de l'architecture MVC : une recherche sur Google vous confirmera que Laravel suit une architecture Modèle-Vue-Contrôleur (MVC). C'est ce qui fait de Laravel un "excellent" Framework pour le développement de vos systèmes web, améliorant les performances, offrant une clarté et favorisant une meilleure documentation .
g-Offrir un super package appelé 'Padel' qui support le paiement par abonnement.

En résumé, nous avons choisi Laravel en raison de sa puissance, de sa convivialité, de sa documentation complète et de ses fonctionnalités avancées qui correspondent parfaitement aux exigences de notre projet.[40]

• PHP

Abréviation de "Hypertexte Pre-processor", est un langage de script largement utilisé par les développeurs web pour créer des sites web dynamiques. Pour notre projet, nous avons spécifiquement utilisé la version 8.1.10 de PHP .[41]

FIGURE 3.1 – PHP .
[42]

• **MySQL**

Nous avons choisi d'installer le serveur web Apache en conjonction avec MySQL en utilisant XAMPP, qui est l'environnement de développement PHP le plus largement utilisé. XAMPP est une distribution gratuite et simple à installer d'Apache, qui inclut également MySQL, PHP et Perl. Dans notre cas, nous avons utilisé la version 8.1.10 de XAMPP.

FIGURE 3.2 – Mysql .
[43]

• **Visual studio code**

Il s'agit d'un éditeur de traitement de texte open source, gratuit et compatible avec plusieurs plates-formes (Windows, Mac et Linux), développé par Microsoft. Il est principalement conçu pour le développement de systèmes utilisant JavaScript, TypeScript et Node.js, mais il peut également être adapté à d'autres langages grâce à son système d'extensions complet.[44]

FIGURE 3.3 – Visual studio code .
[45]

• **Mailtrap**

Un service qui offre une solution sécurisée pour tester les emails envoyés depuis des environnements de développement et de mise en scène. Il crée une boîte de réception virtuelle où tous les emails sont capturés, vous permettant ainsi de tester et d'améliorer vos campagnes d'email avant de les envoyer aux utilisateurs réels. Cela garantit que vos emails sont bien conçus et fonctionnent correctement avant d'être diffusés à grande échelle.[46]

FIGURE 3.4 – Mailtrap .
[47]

• **Expose dev**

Une application tunnel qui offre la possibilité de partager vos sites Web locaux et vos applications avec d'autres utilisateurs sur Internet. Cette application utilise PHP comme base et est également open source, ce qui signifie que son code source est accessible et modifiable par la communauté.

FIGURE 3.5 – Expose dev .
[48]

3.3 Méthodologie de développement du système

Dans cette section, nous allons commencer par décrire le système final souhaité, puis nous aborderons le processus utilisé pour le développer, ainsi que l'approche et les étapes suivies pour mettre en œuvre cette vision.

3.3.1 Description du système

Le système de facturation est un programme informatique très reconnu dans les différents établissements, conçu pour simplifier la gestion commerciale des factures de vente. Il vise à simplifier diverses tâches telles que l'édition des factures, l'envoi, l'archivage et l'intégration dans le système comptable. L'objectif de ce mémoire est de transformer une application de facturation uni-locataire de type bureau ou service web en un système d'abonnement multi-locataire appelé "Mt-Fact". Ce système offrira à ses locataires un modèle de facture préconçu. Cette solution permettra non seulement de gagner du temps, mais aussi de personnaliser les factures en fonction du modèle d'abonnement choisi par les locataires du système. Avec le system d'abonnement multi-locataire "Mt-Fact", la facturation en ligne devient plus simple que jamais, offrant la possibilité de créer et d'envoyer facilement des factures. Le modèle de facture convertit instantanément les informations fournies en factures au format Portable Document Format (PDF). Les factures sont enregistrées en tant que fichiers PDF téléchargeables, vous permettant de les envoyer directement par e-mail aux clients ou de les imprimer pour une remise en main propre.
Dans les sections suivants, nous décrirons la méthodologie adoptée, ainsi que la conception du modèle finalisé et la mise en œuvre du système cible appelé Mt-Fact, ainsi que ses différentes interfaces.

3.3.2 Processus d'élaboration du système

La Figure 3.6 ci-dessous représente le processus de développement d'un système ambitieux. L'objectif principal est de migrer d'un système uni-locataire vers un système multi-locataire, avec un accès d'abonnement basé sur la souscription.

FIGURE 3.6 – Processus de système.

3.3.3 L'approche adoptée

Dans le travail de Nugraheni [49] et la formation Udemy [1] , une approche est proposée pour le processus d'élaboration vers une système SaaS multi-locataire, détaillée comme suit :

a- Création d'un nouveau module pour gérer les procédures d'enregistrement des locataires et de leurs utilisateurs.

b- Construction du système de base de données pour permettre la création d'une base de données pour chaque locataire d'un groupe enregistré.

c- Ajout d'une couche de personnalisation supplémentaire lorsque les utilisateurs entrent dans le système : un modèle d'authentification est utilisé pour mapper les utilisateurs aux locataires et à leurs données (y compris l'identifiant du locataire ainsi que d'autres informations pertinentes). Le module d'authentification est utilisé comme mécanisme d'identification de chaque locataire. Chaque locataire qui se connecte avec succès au système aura un identifiant de locataire qui peut être utilisé pour la configuration et la personnalisation de système.

d- Fourniture à chaque locataire d'une page d'initialisation pour configurer et personnaliser les composants de l'interface, tels que le logo, la couleur, l'apparence, etc., qui seront enregistrés et transmis au serveur de système.

e- Les données de configuration de la base de données seront transmises aux serveurs de base de données pour la transformation des requêtes.

f- Développement d'un nouveau tableau de bord pour l'administrateur du système afin de surveiller les performances et l'état de chaque système locataire. Les principales

1. https ://www.udemy.com/course/laravel-saas/

étapes de l'approche utilisé sont inspirées du travail référencé [49] et d'une formation Udemy exceptionnelle spécifiquement axée sur le Framework Laravel.c'est étapes sont détaillées dans le tableau 3.2 :

Étape	Description
Étape 0	Installer et configurer le Framework "Laravel" ainsi que la base de données associée et les bibliothèques (par exemple, Laravel UI). Créer les modèles principaux du système SaaS Mt-Fact.
Étape 1	Installer et configurer le package Paddle-Cachier avec la commande "composer require laravel/cachier paddle". Introduire les tables : client (locataire), abonnement et reçu. Configurer les plans d'abonnemen en tenant compte de leurs caractéristiques. Extraire les attributs de chaque plan à partir du site web des fournisseurs sandbox-paddles et les réintégrer dans le modèle de plan. Ajouter le modèle PlanFeature.
Étape 2	Créez soigneusement des vues pour toutes les interfaces utilisées, en utilisant des modèles fournis par diverses bibliothèques déjà installées (comme Bootstrap).
Étape 3	Créer tous les contrôleurs pour vérifier les différents modèles développés dans les phases précédentes.

TABLE 3.2 – Étapes du développement du système SaaS Mt-Fact.

3.4 Architecture et modéle du Système

Dans cette section, nous souhaitons fournir une vue d'ensemble de l'architecture générale du système Mt-Fact, ainsi que de la modélisation de ce système (la modélisation adopté dand ce mémoire et appuyer sur la démarche Unified Modeling Language (UML).

3.4.1 Architecture générale du système

Comme représenté dans la Figure 3.7, l'architecture du system développé sous "Mt-Fact" est basée principalement sur l'architecture MVC (Modèle-Vue-Contrôleur), qui est largement connue et se présente comme suit :
Le Tenant représente le locataire abonné du système

Le navigateur :

représente le moyen d'interaction du locataire avec le fournisseur du système.

Les contrôleurs :

un fichier qui regroupe toutes les classes qui contrôlent les modèles et les vues (par exemple, le contrôleur locataire).

Les modèles :

un ensemble de fichiers (classes) qui constituent la couche d'interaction avec la base de données de système.

Les vues :

Tout dossier ou classes qui représentent les modèles de programmation en CSS/HTML/JS, par exemple, dans le Framework BootStrap.

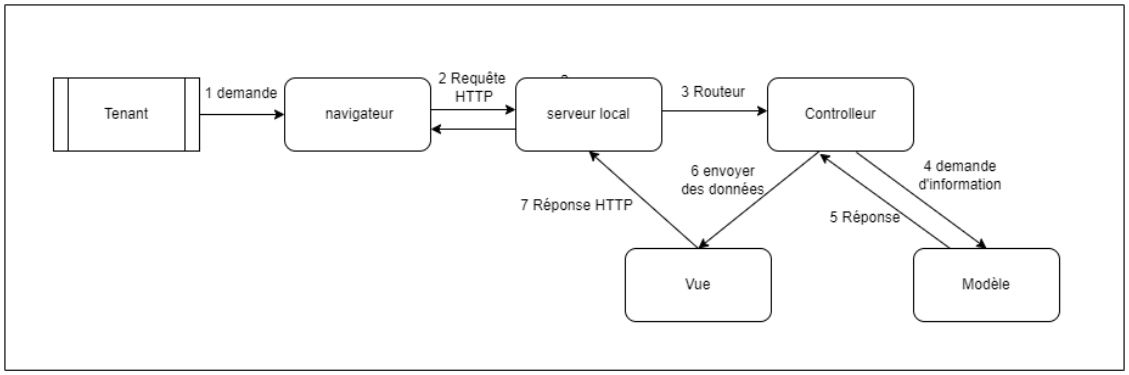


FIGURE 3.7 – Mt-Fact SaaS Architecture

3.4.2 La Modélisation du Système

La modélisation de notre système repose principalement sur l'approche UML, en particulier sur le diagramme de cas d'utilisation. Ce diagramme exprime de manière précise les différents besoins fonctionnels de notre système d'abonnement multi-locataire. L'UML (Unified Modeling Language) est une méthode de modélisation visuelle utilisée pour concevoir et développer des programmes orientés objet. Grâce à l'UML, il est possible de modéliser une grande variété d'applications logicielles qui peuvent fonctionner sur différentes plateformes, systèmes d'exploitation et réseaux, et qui peuvent être écrites dans divers langages de programmation.

Diagramme de cas d'utilisation

La Figure 3.8 présente une vue d'ensemble des différentes fonctionnalités du système étudié, représentées par un diagramme de cas d'utilisation. Le tableau 3.3 fournit une description détaillée de l'ensemble de ces besoins.

Gestion de l'authentification	Permet à l'utilisateur de créer un compte, de se connecter et de gérer les informations d'identification pour accéder aux fonctionnalités du système.
Abonnement	Permet à l'utilisateur de souscrire à un plan d'abonnement, de consulter et modifier les informations d'abonnement, y compris le mode de paiement et le plan de souscription .
Création et gestion des factures	Permet à l'utilisateur de créer, modifier et gérer les factures dans le système .
Options d'exportation	Offre à l'utilisateur la possibilité de télécharger, envoyer par e-mail ou exporter les facturesau format PDF.

TABLE 3.3 – Les déférents besoins fonctionnels de système

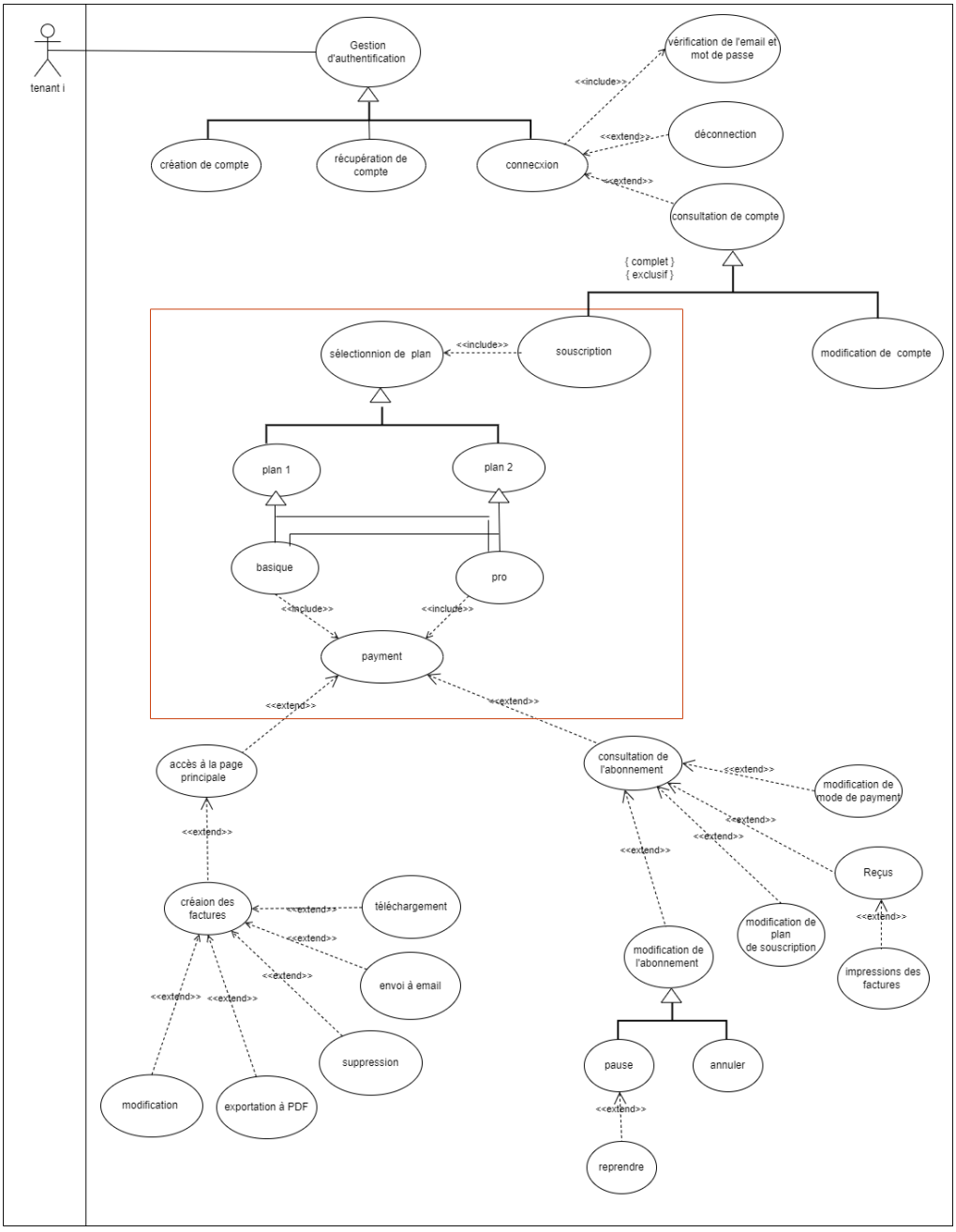


FIGURE 3.8 – Diagramme de cas d'utilisation

3.4.3 Réalisation du système

3.4.3.1 La Mise en œuvre :

Le système est développé en utilisant le langage PHP sur le Framework Laravel avec l'éditeur visuel Studio. Nous avons adopté l'architecture Model-View-Controller (MVC) (Modèle-Vue-Contrôleur). La Figure 3.9 montre une capture d'écran illustrant quelques modèles et contrôleurs.

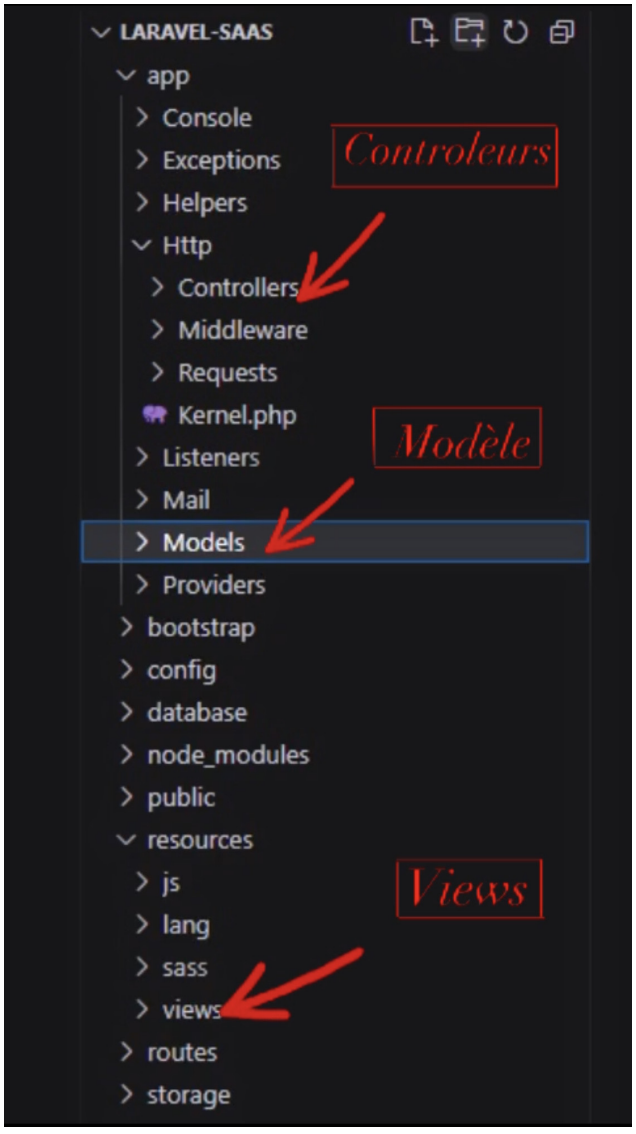


FIGURE 3.9 – l'architecture MVC de Mt-Fact

Pour la mise en ouvre de notre systeme, nous avons choisi d'utiliser une base de données SQL fournie par XAMPP. La Figure 3.10 affiche une capture d'écran montrant la structure de la base de données utilisée dans le système.

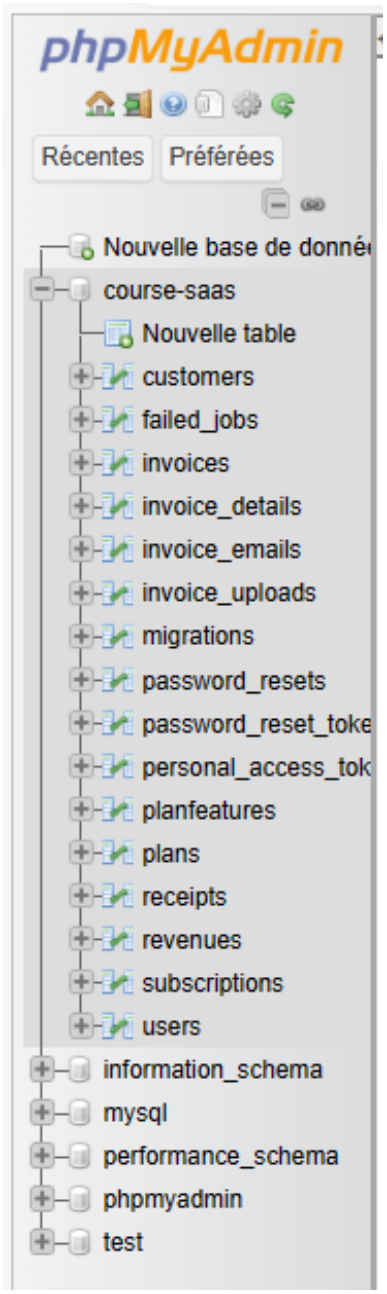


FIGURE 3.10 – Structure de la base de données de Mt-Fact

Pour l'hébergement de notre système, nous avons utiliser le package Laravel Valet en collaboration avec Exposé Dev. La Figure 3.11 ci-dessous présente une capture d'écran illustrant l'utilisation de Laravel Valet avec Exposé Dev. Cette combinaison nous offre une solution d'hébergement pratique et performante pour notre système.

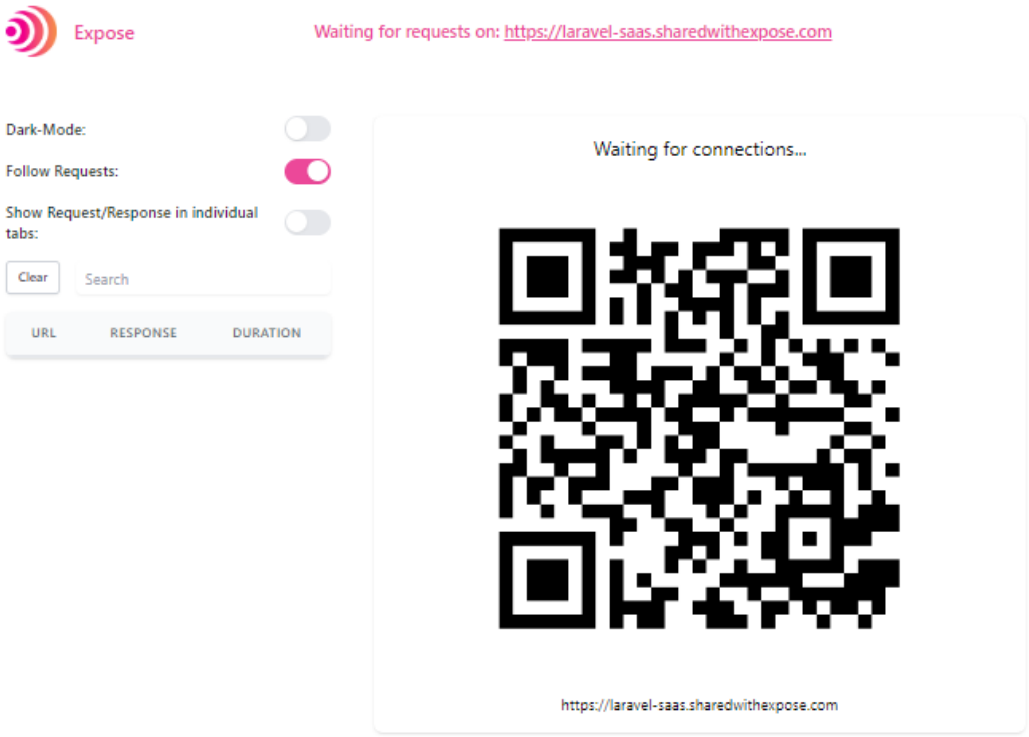


FIGURE 3.11 – l'hébergement de Mt-Fact.

3.4.3.2 Les interface du système

Lorsque les locataires accèdent au système pour la première fois, ils sont accueillis par la page d'accueil qui comprend la gestion de l'authentification (connexion, inscription). Cette fonctionnalité est illustrée dans la Figure 3.12.

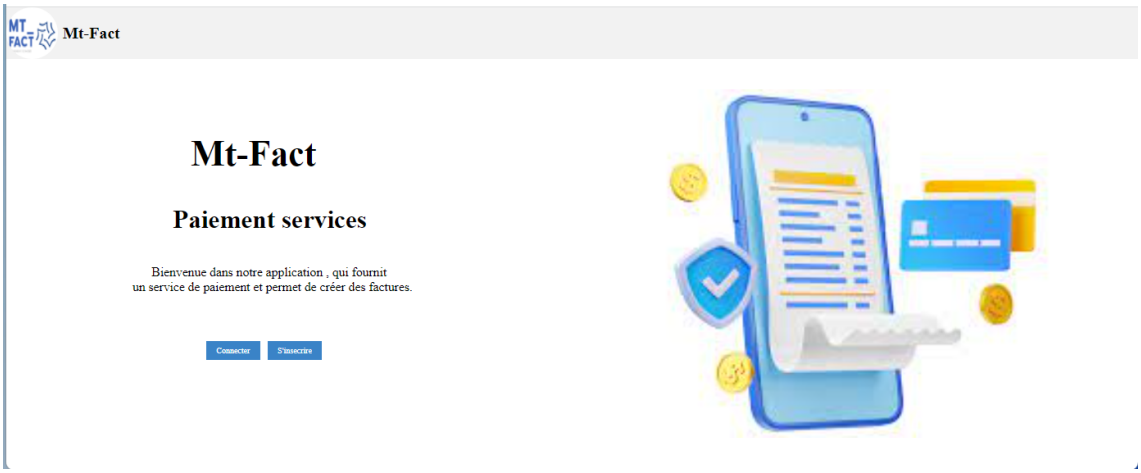


FIGURE 3.12 – Interface principale de Mt-Fact

Une fois que l'interface principale est affichée et que le locataire appuie sur le bouton de connexion (ou inscription), il doit d'abord s'inscrire en fournissant son nom, son adresse e- mail et son mot de passe, et en les confirmant comme illustré dans la Figure 3.13. Alternativement, il peut simplement saisir son adresse e-mail et son mot de passe comme indiqué dans la Figure 3.14. Si les informations fournies sont incorrectes, le système affiche un message d'erreur et le locataire peut réessayer.

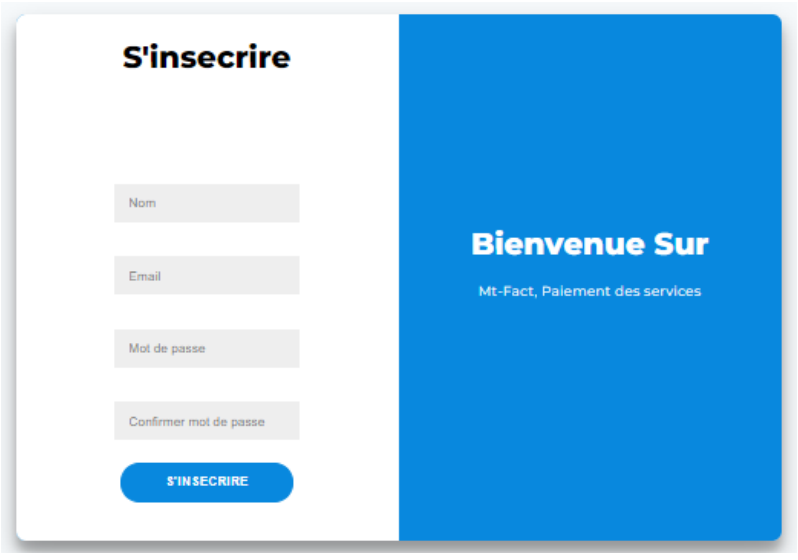


FIGURE 3.13 – Interface d'inscription

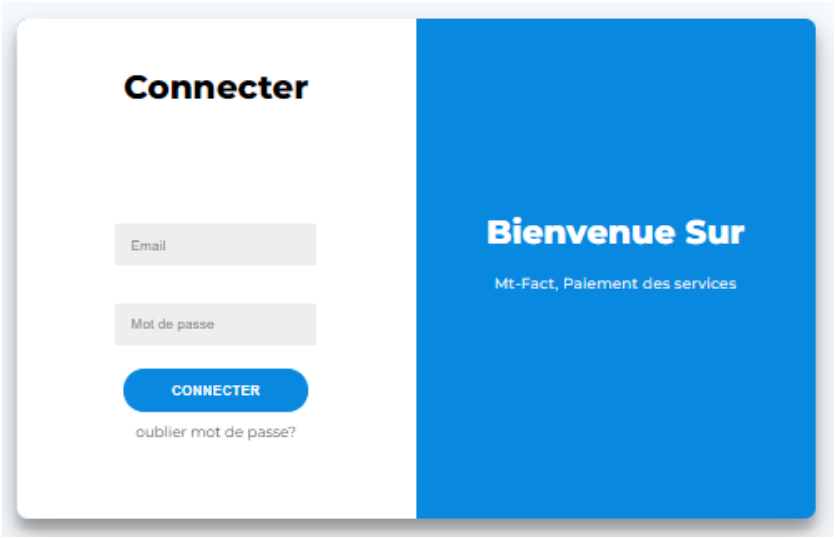


FIGURE 3.14 – Interface de Connexion

-Si l'utilisateur oublie son mot de passe, il a la possibilité de le récupérer en cliquant sur le bouton "Envoyer le lien de réinitialisation du mot de passe". Le système enverra alors un nouveau mot de passe à son adresse e-mail, qu'il pourra utiliser pour accéder à nouveau à son compte, comme illustré dans la Figure 3.15.

FIGURE 3.15 – Interface de récupération de mot de passe

Une fois connecté, le nom de compte de l'utilisateur s'affiche. En cliquant dessus, il a la possibilité d'accéder à ses informations de compte en sélectionnant "Compte" ou de se déconnecter en choisissant "Déconnecter". Cela est illustré dans la Figure 3.16.

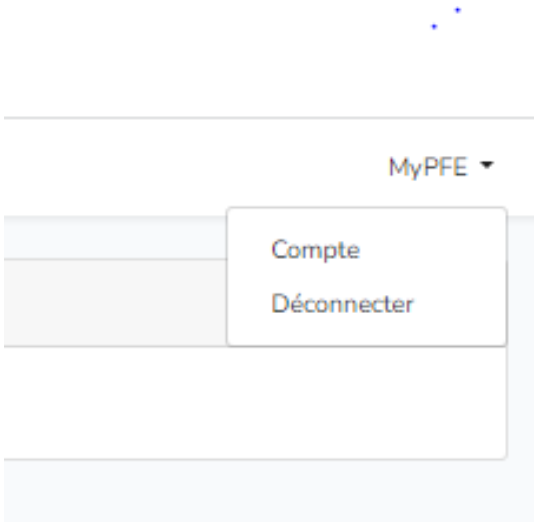


FIGURE 3.16 – Interface d'accueil

Lorsque l'utilisateur clique sur "Compte", il peut mettre à jour ses informations, comme illustré dans la Figure 3.17.

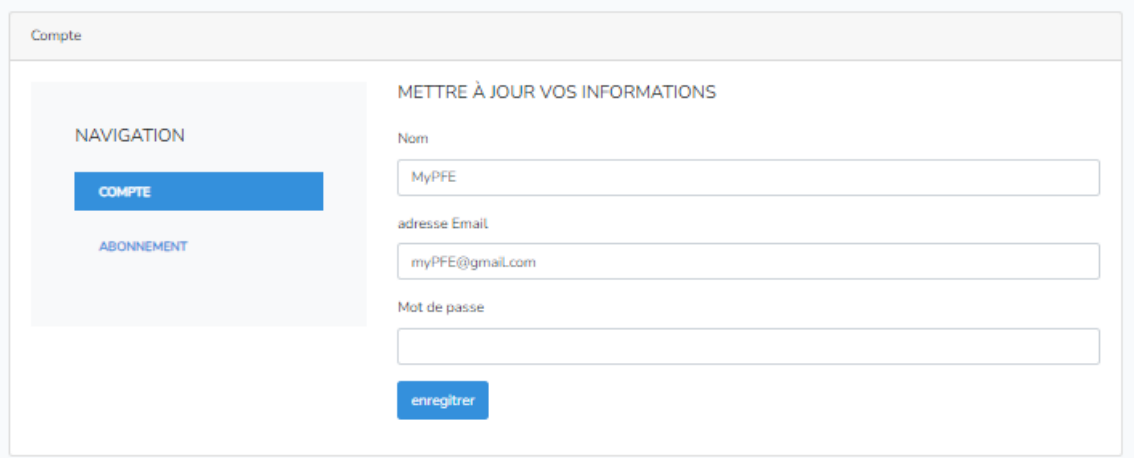


FIGURE 3.17 – Interface de compte

Étant donné que ce système n'est pas gratuite, il existe des abonnements mensuels et annuels qui varient selon la demande du client, comme indiqué dans la Figure 3.18 et 3.19.

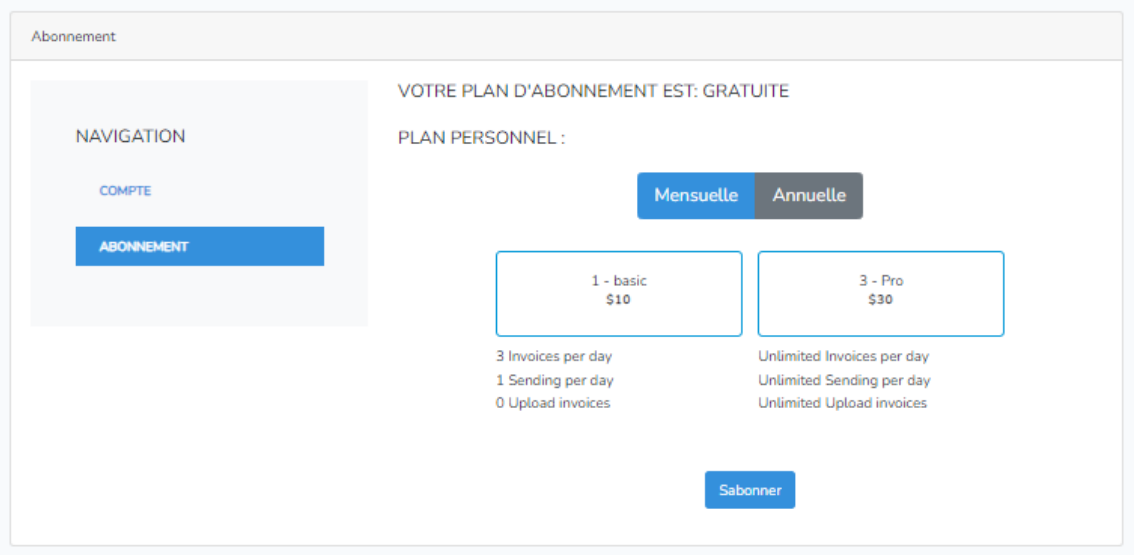


FIGURE 3.18 – L'interface de l'Abonnement mensuel.

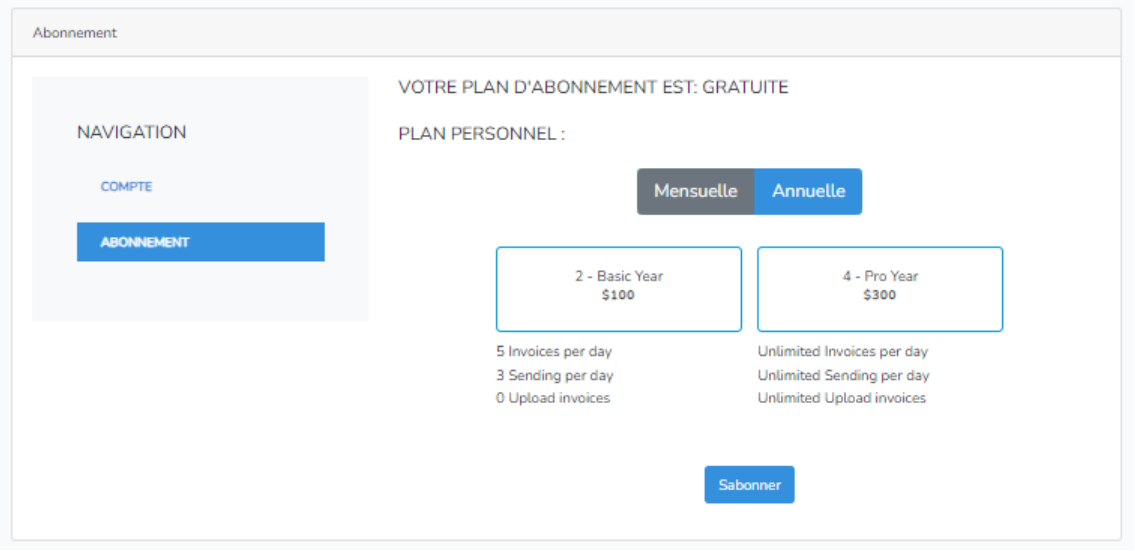


FIGURE 3.19 – L'interface de l'Abonnement annuel.

Une fois que le locataire a sélectionné le type d'abonnement, il peut choisir le pays et le mode de paiement, soit par carte (Visa-Card, PayPal) comme illustré dans la Figure 3.20.

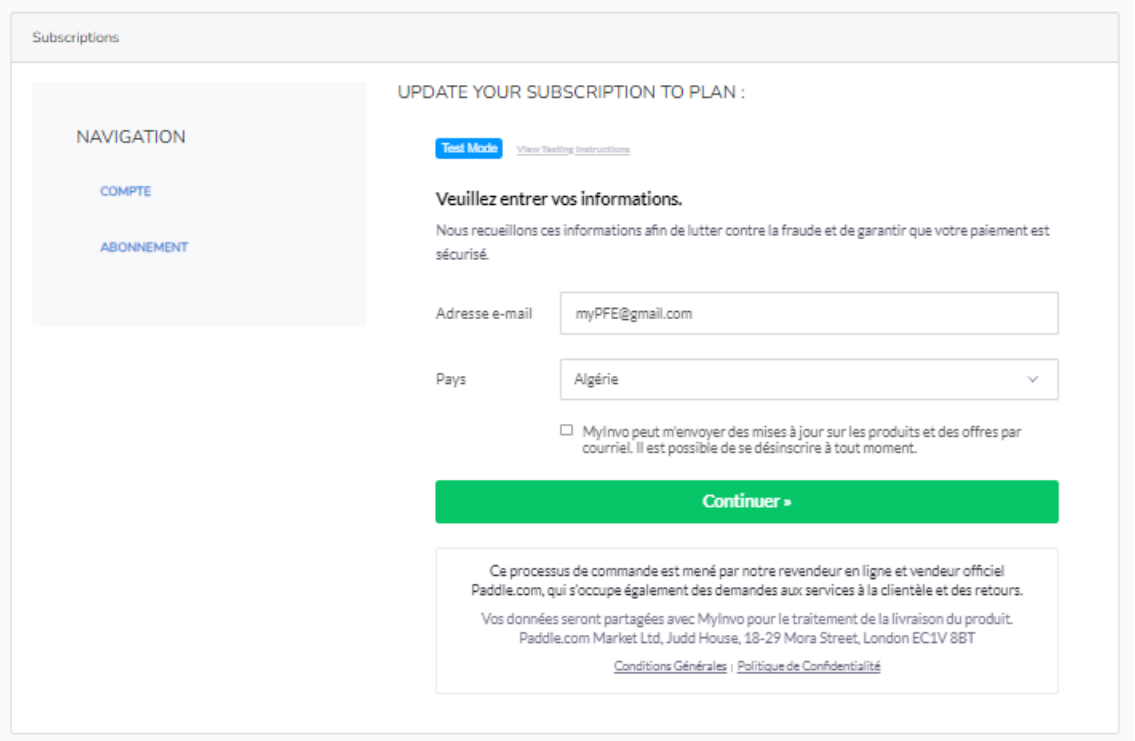


FIGURE 3.20 – Interface du pays d'abonnement.

Une fois que le mode de paiement est sélectionné, l'utilisateur doit fournir les informations requises dans les deux cas et cliquer sur le bouton "S'inscrire maintenant", tel qu'illustré dans la Figure 3.21.

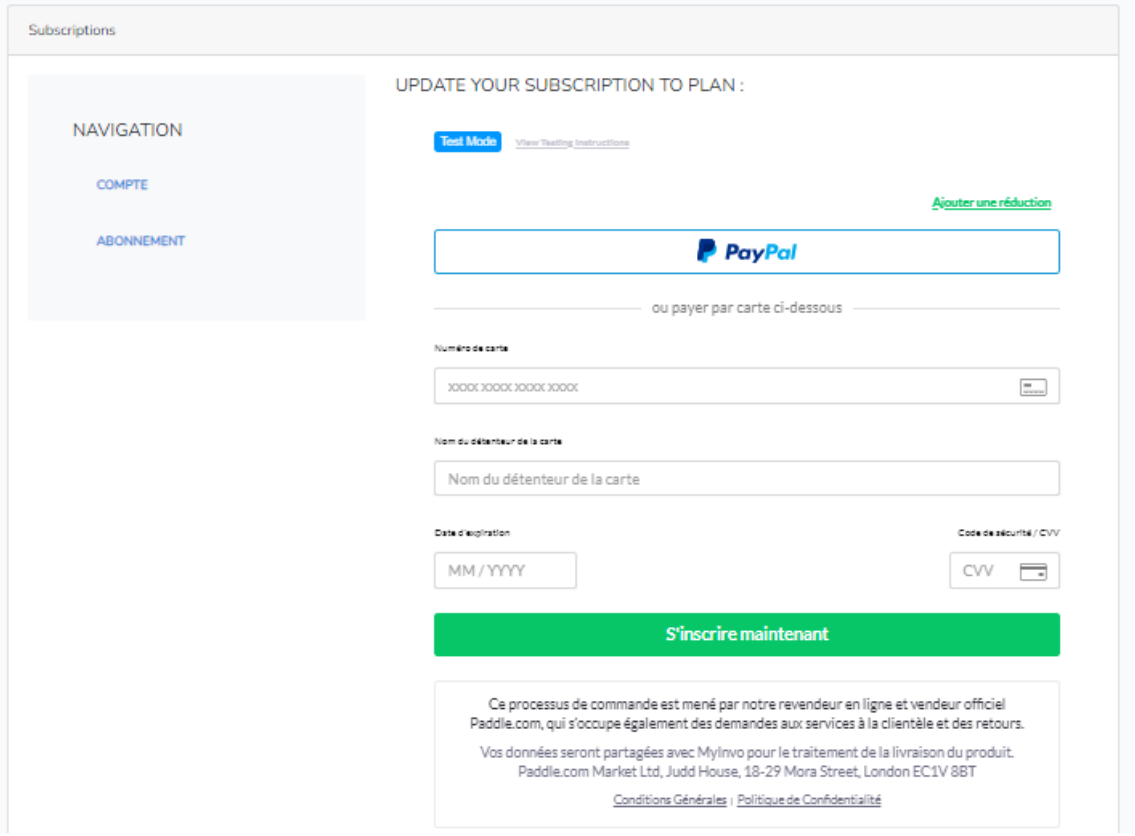


FIGURE 3.21 – Interface de paiement.

Une fois que le locataire a souscrit au systeme, toutes ses informations, telles que le mode de paiement, le montant payé et la date de fin de la période d'abonnement, seront enregistrées dans son compte, comme illustré dans la Figure 3.22.

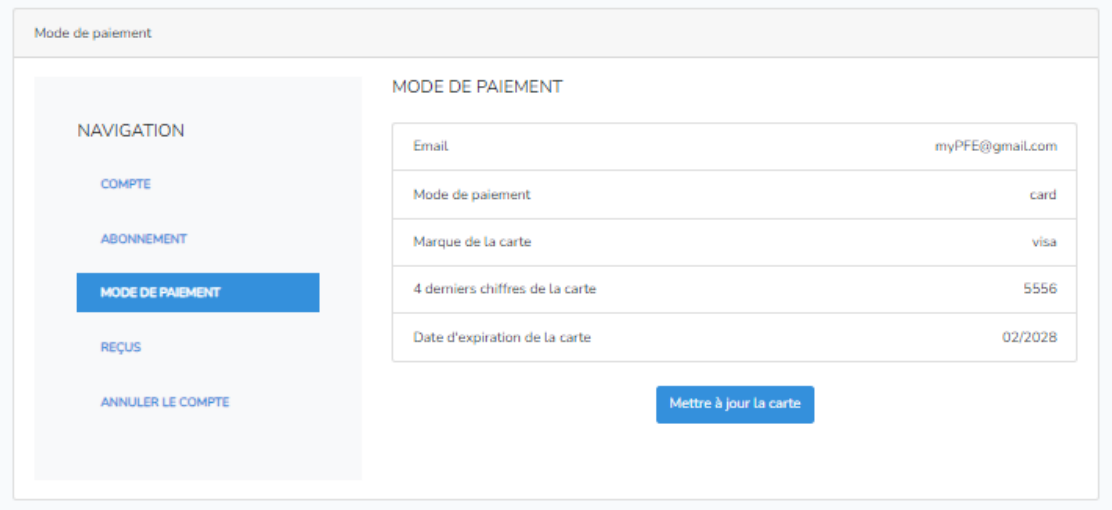


FIGURE 3.22 – Interface de méthode de paiement.

Dans cette section, l'utilisateur a la possibilité de modifier le mode de paiement, comme illustré dans la Figure 3.23.

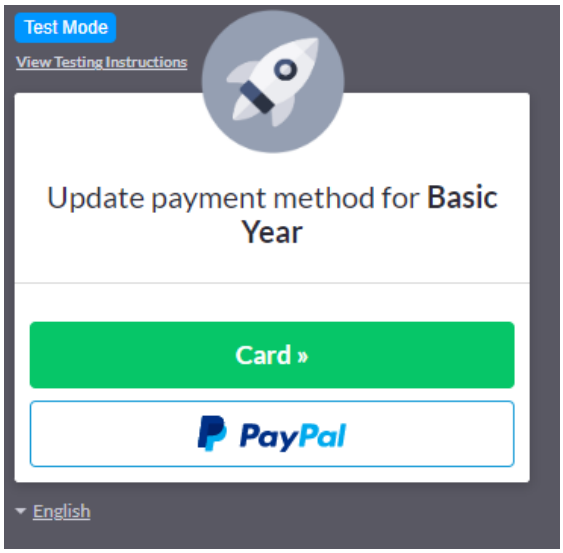


FIGURE 3.23 – Interface de changement de paiement.
[4]

Le Locataire Abonné à également la possibilité de consulter tous ses abonnements précédents, y compris les dates auxquelles ils étaient actifs, comme illustré dans la Figure 3.24. De plus, il peut les télécharger, comme indiqué dans la Figure 3.25.

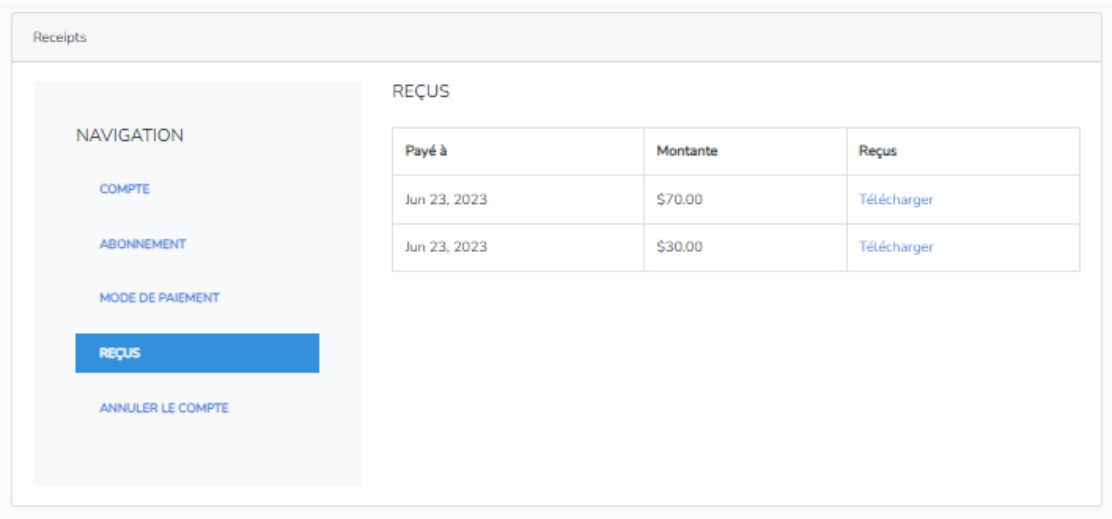
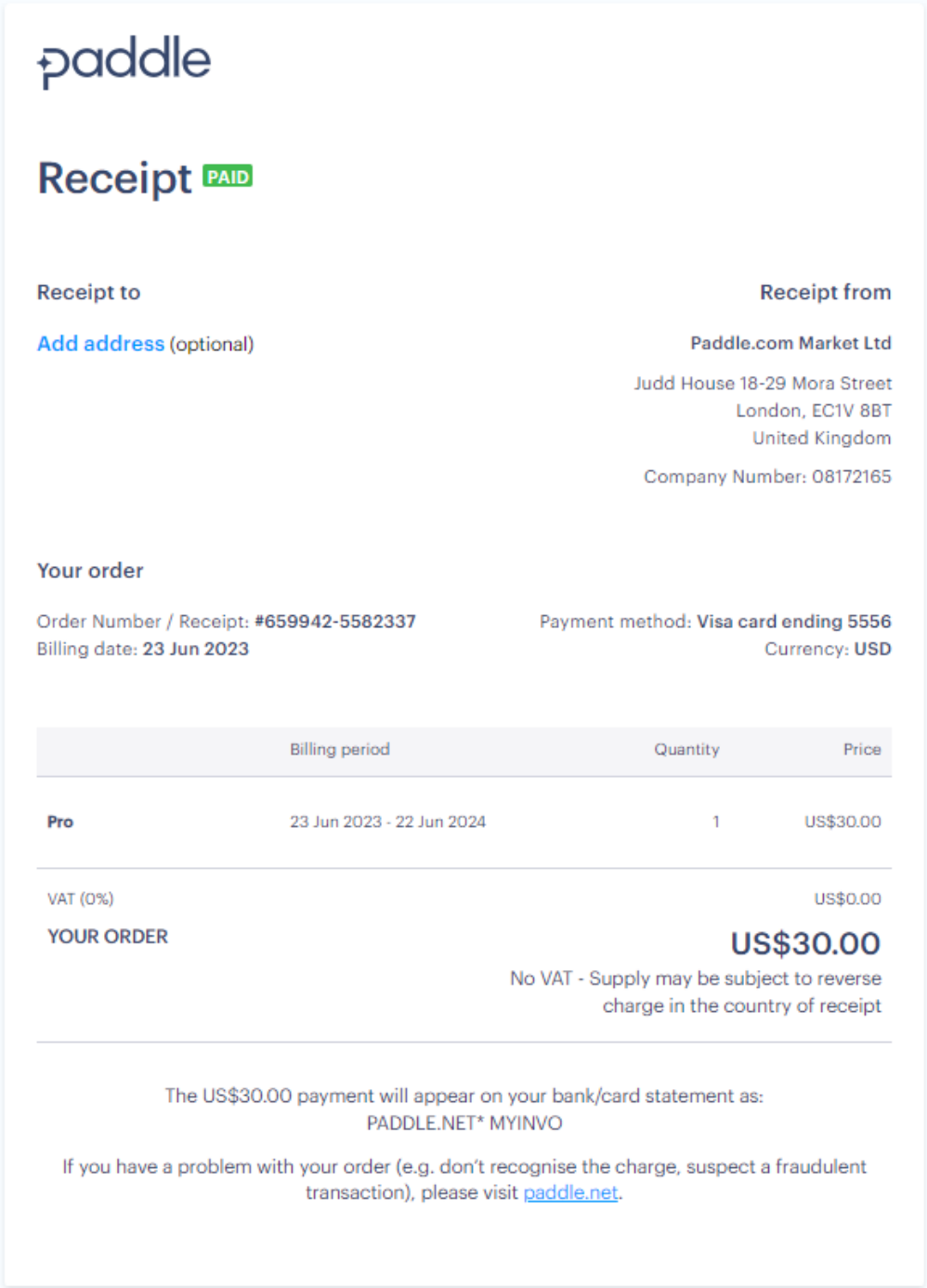


FIGURE 3.24 – Interface de consultation



paddle

Receipt PAID

Receipt to

Add address (optional)

Receipt from

Paddle.com Market Ltd

Judd House 18-29 Mora Street
London, EC1V 8BT
United Kingdom

Company Number: 08172165

Your order

Order Number / Receipt: **#659942-5582337**
Billing date: **23 Jun 2023**

Payment method: **Visa card ending 5556**
Currency: **USD**

	Billing period	Quantity	Price
Pro	23 Jun 2023 - 22 Jun 2024	1	US$30.00

VAT (0%) US$0.00

YOUR ORDER **US$30.00**

No VAT - Supply may be subject to reverse charge in the country of receipt

The US$30.00 payment will appear on your bank/card statement as:
PADDLE.NET* MYINVO

If you have a problem with your order (e.g. don't recognise the charge, suspect a fraudulent transaction), please visit paddle.net.

FIGURE 3.25 – Téléchargement de la facture via Paddle.

Si le souscripteur locataire souhaite suspendre temporairement son compte ou le supprimer définitivement, il peut le faire en cliquant sur les boutons spécifiés dans la Figure 3.26.

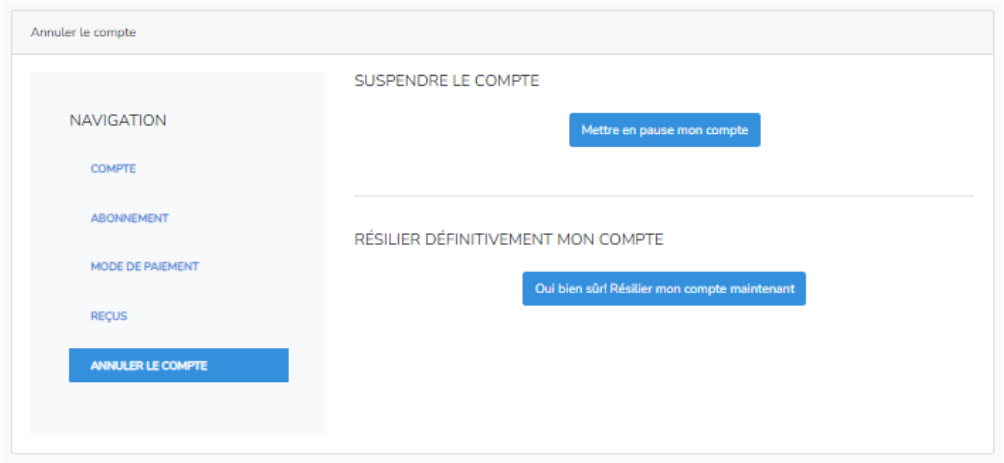


FIGURE 3.26 – Interface de suspension et de suppression du compte.

Si le locataire souscripteur souhaite annuler la suspension de votre compte, il possède la possibilité de le faire en cliquant simplement sur le bouton "Continuer mon compte" indiqué dans la Figure 3.27.

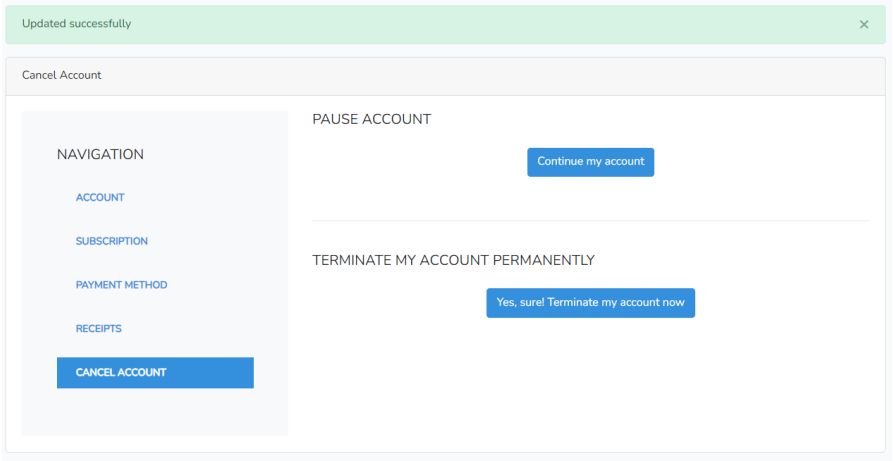


FIGURE 3.27 – Interface de l'annulation de suspension du compte.

Une fois que le paiement de ses abonnements a été vérifié, l'utilisateur peut accéder à l'interface de facturation où il peut ajouter des factures, comme illustré dans la Figure 3.28.

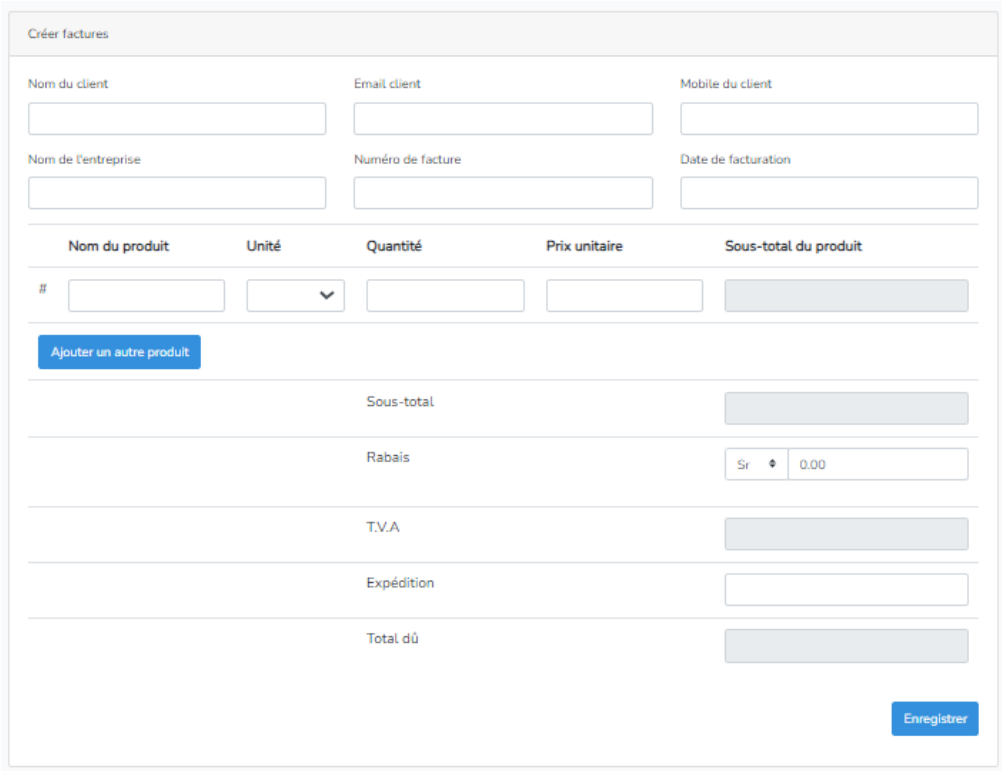


FIGURE 3.28 – Interface de création des factures

Des modifications et des suppressions de factures peuvent également être effectuées, comme illustré dans la Figure 3.29.

FIGURE 3.29 – Interface des factures

Les factures peuvent être exportées au format PDF, comme le montre la Figure 3.30.

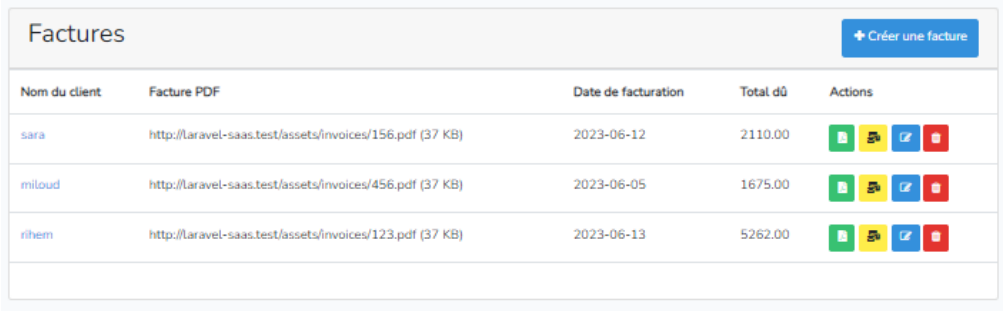


FIGURE 3.30 – Interface d'exportation en format PDF.

Les factures peuvent être envoyées par e-mail, comme illustré dans la Figure 3.31.

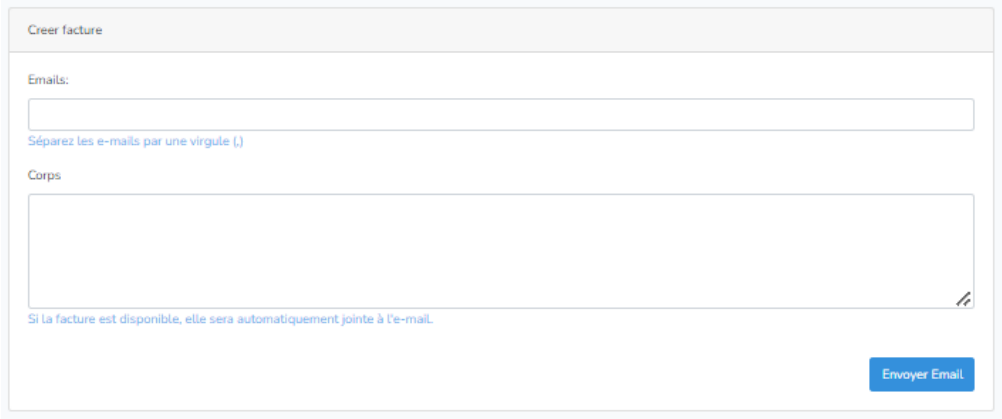


FIGURE 3.31 – Interface d'envoi par e-mail.

En outre, les factures peuvent être téléchargées, comme le montre la Figure 3.32.

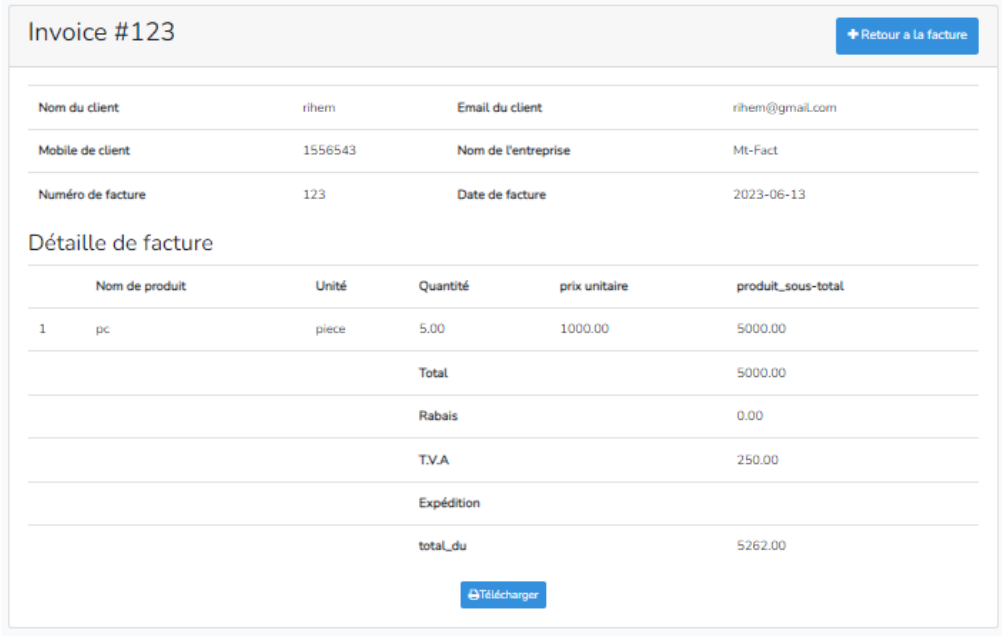


FIGURE 3.32 – Interface de téléchargement.

3.5 Discussion

Le principal objectif de ce mémoire est de développer un système de facturation multi-locataire basé sur un modèle d'abonnement, utilisant une approche de base de données multiple. Cependant, en raison de difficultés rencontrées lors de l'implémentation et du raffinement de cette solution, notamment avec les bibliothèques Tenancy for Laravel et Spatie fournies par Laravel, nous avons dû ajuster notre approche vers une base de données unique. Cette modification a été nécessaire pour éviter de bloquer l'avancement du projet. Néanmoins,l'approche initiale de ce mémoire sera considérée comme une perspective importante pour les études d'implémentation futures.

3.6 Conclusion

Dans ce chapitre final, nous avons fourni une description détaillée de l'implémentation et de la réalisation de notre système. Nous avons présenté l'environnement matériel et logiciel utilisé pendant l'implémentation, ainsi que la description du système d'abonnement ciblé par notre approche. De plus, nous avons exposé le processus de développement suivi et la méthodologie adoptée, en nous inspirant des meilleures pratiques. Enfin, nous avons expliqué les différentes interfaces de notre système multi-locataire basé sur l'abonnement.

Conclusion Générale & Perspectives

L'impact de l'intégration du modèle de service SaaS et de l'architecture multi-tenant sur les applications hébergées par le cloud est facilement sous-estimé, mais peut soulever des défis majeurs en matière de spécification et de mise en œuvre. Cependant, cela peut entraîner des défis majeurs en termes de spécification et de mise en œuvre.
Dans le cadre de ce mémoire, nous avons documenté les efforts entrepris pour créer et mettre en place une application d'abonnement multi-locataire en utilisant le Framework Laravel, avec une approche de base de données unique. L'objectif principal de ce travail était de décrire les étapes suivies pour la conception et la réalisation du système d'abonnement destiné.
Une partie significative de ce mémoire s'est concentrée sur :
• Une revue concise de l'état de l'art des paradigmes basés sur la virtualisation, y compris le cloud computing, ainsi que de leurs modèles de déploiement et de service.
• La description, l'explication et la distinction de plusieurs modèles de maturité et de réalisation d'architectures multi-locataires (à base de données unique ou multiple) dans le cloud computing.
• La présentation et l'apprentissage du Framework Laravel, qui offre d'excellentes bibliothèques pour le modèle SaaS et l'architecture multi-locataire.
• La conception et mise en place d'un système d'abonnement SaaS typique nommé "MT-Fact" avec le Framework Laravel. Cependant, il convient de noter que les fonctionnalités et l'architecture multi-locataire mises en œuvre dans ce manuscrit sont relativement limitées. Ces limitation vient principalement des difficultés de programmation rencontrées notamment lors de l'intégration des librairies multitenancy laravel (Spatie)[50] utilisées lors de l'implémentation, ces limitations influencent négativement et considérablement la crédibilité de notre travail.
Le travail présenté dans ce mémoire ouvre de nombreuses perspectives prometteuses à explorer. À court terme, il est prévu de mettre en œuvre le système "MT-Fact" sur différentes plates-formes telles que Symphony , Codnighter , et Django , entre autres. De plus, il est envisagé d'intégrer d'autres bibliothèques prenant en charge la multi-localisation (par exemple "Tenancy for Laravel"). En ce qui concerne l'expansion de ce système, il est prévu de prendre en compte non seulement l'approche d'une base de données unique, mais aussi celle d'une base de données multiple. De plus, il est important de ne pas se limiter uniquement aux bases de données SQL (comme MySQL) mentionnées dans ce mémoire, mais également de considérer d'autres types de bases de données. Notre étude prévoit d'élargir cette perspective en incluant des bases de données telles qu'OracleDB, PostgresSQL, ainsi que d'explorer d'autres types de bases de

données NoSQL, tels que MongoDB, Cassandra et Google Datastore.
À long terme, l'architecture multi-tenant présentée dans ce mémoire vise à s'intégrer dans un environnement virtualisé basé sur le cloud computing en tant que paradigme de recherche. Cependant, dans une perspective future, nous envisageons de proposer des implémentations similaires dans d'autres environnements virtualisés. Cela signifie que nous pourrions explorer l'implémentation du multitenancy dans des domaines tels que le Fog computing, Edge computing, et d'autres encore.
Le travail présenté dans ce mémoire représente une première étape dans la compréhension et l'étude globale des solutions de multitenancy dans tous les environnements virtualisés, tels que le Cloud, le Fog, l'Edge, et bien d'autres. Cependant, il y a plusieurs extensions qui n'ont pas pu être abordées en raison des contraintes de temps de rédaction de ce mémoire. Ces extensions englobent des aspects cruciaux tels que la sécurité, la tolérance aux pannes, l'équilibrage de charge, la gestion des ressources, et bien d'autres encore, liés à la multilocation. En outre, il convient également de prendre en compte des éléments tels que le type de serveur d'administration (Windows/Linux), le type de serveur web (Apache, Nginx,Tomcat) et le type de base de données (SQL/NoSQL). Chacun de ces aspects ouvre de nouvelles pistes de recherche à explorer.

Bibliographie

[1] N. Gavlov. 25 must-know cloud computing statistics in 2022. https ://webtribunal.net/blog/cloud-computing-statistics/gref. Accessed 26 Apr 2022.

[2] RF Wireless World. Difference between edge, fog, and cloud computing. https ://www.rfwireless-world.com/images/Difference-between-Edge-Fog-and-Cloud-Computing.jpg, Accessed June 5, 2023.

[3] Yousri Kouki, Thomas Ledoux, Damián Serrano, Sara Bouchenak, Jonathan Lejeune, Luciana Arantes, Julien Sopena, Pierre Sens, and Paris LIP6-INRIA. Sla et qualité de service pour le cloud computing. In *Conférence d'informatique en Parallélisme, Architecture et Système, ComPAS*, volume 2013, 2013.

[4] Title of the postcloud computing. https ://fr.linkedin.com/pulse/les-5-avantages-de-linformatique-en-nuage-denis-paradis, Accessed on June 5, 2023.

[5] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[6] Nist - national institute of standards and technology.

[7] Site génération-nt - les numériques.

[8] Cisco systems - wikipedia.

[9] Cisco systems — Wikipedia, the free encyclopedia. [Online ; accessed 29-June-2023].

[10] Rajkumar buyya - wikipedia.

[11] Rajkumar buyya — Wikipedia, the free encyclopedia. [Online ; accessed 29-June-2023].

[12] Shivaji P Mirashe and Namdeo V Kalyankar. Cloud computing. *arXiv preprint arXiv :1003.4074*, 2010.

[13] Allianz Cloud. Cloud Deployment Models. https ://allianzcloud.com/cloud-deployment-models-2/, Date Inconnue.

[14] Silia BENKHALLEF and Amel ABER. *Design and Realization of Cloud SaaS Multi-Tenant Application*. PhD thesis, Université Ibn Khaldoun-Tiaret-, 2022.

[15] Nicolas Grevet. Le cloudcomputing : évolution ou révolution. *Pourquoi, quand, comment et surtout faut-il prendre le risque*, 2009.

[16] Allan Lefort Kuhn. Cloud computing.

[17] WayToLearnX. Difference entre iaas, saas et paas. https ://waytolearnx.com/2019/03/difference-entre-iaas-saas-et-paas.html, March 2019.

[18] Jawaher Al-Khuzaei. Content is king. https ://www.linkedin.com/pulse/content-king-jawaher-al-khuzaei, Date Inconnue.

[19] Ahmed Hadi Ali AL-Jumaili, Yousif I Al Mashhadany, Rossilawati Sulaiman, and Zaid Abdi Alkareem Alyasseri. A conceptual and systematics for intelligent power management system-based cloud computing : Prospects, and challenges. *Applied Sciences*, 11(21) :9820, 2021.

[20] Xuxu Zheng, Qingzhong Li, and Lanju Kong. A data storage architecture supporting multi-level customization for saas. In *2010 Seventh Web Information Systems and Applications Conference*, pages 106–109. IEEE, 2010.

[21] Wei-Tek Tsai, Yu Huang, and Qihong Shao. Testing the scalability of saas applications. In *2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 1–4. IEEE, 2011.

[22] GB Pallavi and Dr P Jayarekha. Multi-tenancy in saas–a comprehensive survey. *International journal of scientific and engineering research*, 7(7) :680, 2014.

[23] Margaret Cochrane Storrie. *Landholdings, land utilisation and settlement in certain isolated areas of Western Scotland, with special reference to two areas of the West Highland seaboard in the Island of Islay in the Inner Hebrides, and the Peninsula of Ardnamurchan and Sunart in northern mainland Argyll.* University of Glasgow (United Kingdom), 1962.

[24] Kartaca. Single tenant vs multi-tenant cloud. https ://kartaca.com/en/single-tenant-vs-multi-tenant-cloud/, Year of publication (if available).

[25] Cor-Paul Bezemer and Andy 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 (iwpse)*, pages 88–92, 2010.

[26] Jaap Kabbedijk, Cor-Paul Bezemer, Slinger Jansen, and Andy Zaidman. Defining multi-tenancy : A systematic mapping study on the academic and the industrial perspective. *Journal of Systems and Software*, 100 :139–148, 2015.

[27] Frederick Chong, Gianpaolo Carraro, and Roger Wolter. Multi-tenant data architecture. *MSDN Library, Microsoft Corporation*, pages 14–30, 2006.

[28] Doudou Fall, Gregory Blanc, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Toward quantified risk-adaptive access control for multi-tenant cloud computing. In *The 6th Joint Workshop on Information Security*, pages 1–14, 2011.

[29] Unknown. Multi-tenant data architecture. https ://renatoargh.files.wordpress.com/2018/01/article-multi-tenant-data-architecture-2006.pdf, 2006.

[30] Manzhi Yang and Huixiang Zhou. New solution for isolation of multi-tenant in cloud computing. In *3rd International Conference on Mechatronics, Robotics and Automation*, pages 334–337. Atlantis Press, 2015.

[31] William S Vincent. *Django for Beginners : Build websites with Python and Django.* WelcomeToCode, 2022.

[32] Azat Mardan. Practical node.js, Year. Accessed on June 29, 2023.

[33] Thomas Risberg Mark Pollack Michael Hunger Jonathan L. Brisbin, Oliver Gierke. Spring data, Year. Accessed on June 29, 2023.

[34] Michael Hartl. *Ruby on rails tutorial : learn Web development with rails*. Addison-Wesley Professional, 2015.

[35] Priyanka Agarwal. .net framework and programming in asp.net, 2019. Consulté le 29 juin 2023.

[36] Matt Stauffer. *Laravel : Up & running : A framework for building modern php apps*. O'Reilly Media, 2019.

[37] Eyad Saleh, Nuhad Shaabani, and Christoph Meinel. A framework for migrating traditional web applications into multi-tenant saas. In *The second International Conference on Advanced Communications and Computation (INFOCOMP)*, pages 100–104, 2012.

[38] Cor-Paul Bezemer, Andy Zaidman, Bart Platzbeecker, Toine Hurkmans, et al. Enabling multi-tenancy : An industrial experience report. In *2010 IEEE International Conference on Software Maintenance*, pages 1–8. IEEE, 2010.

[39] Wikipedia contributors. Laravel, Year. Accessed on June 29, 2023.

[40] Andri Sunardi et al. Mvc architecture : A comparative study between laravel framework and slim framework in freelancer project monitoring system web based. *Procedia Computer Science*, 157 :134–141, 2019.

[41] Danial Kafi Ahmad, Mariam Farida Ahmad, Mohd Nazmi Ahmad, and Abdullah Syafiq Ahmad. An experiment of animation development in hypertext preprocessor (php) and hypertext markup language (html). *Int. J. Sci. Res. in Computer Science and Engineering Vol*, 8(2), 2020.

[42] Wikipédia. PHP. https ://fr.wikipedia.org/wiki/PHP.

[43] 1000logos.net. MySQL Logo. https ://1000logos.net/mysql-logo/, Date Inconnue.

[44] Wikipedia contributors. Visual studio code, Year. Accessed on June 29, 2023.

[45] Google image - vscode. Accessed : July 1, 2023.

[46] Mailtrap. What is an email api ? https ://mailtrap.io/blog/what-is-an-email-api/, 2023.

[47] Saasworthy. Mailtrap.io. https ://www.saasworthy.com/product/mailtrap-io, Date Inconnue.

[48] Expose.dev. https ://expose.dev/.

[49] E. Nugraheni. Migration of web application sima into multi-tenant saas. In *International Conference on ICT for Smart Society*, pages 1–4, 2014.

[50] Laravel multitenancy documentation.