Royaume du Maroc
Haut-Commissariat au Plan

# Development and Deployment of a Web-Based SaaS (Software-as-a-Service) App for Hotels Internal Management

**Thesis for the degree of Master of Engineering**

**KERBAL Ilyas**

**Supervised by:**
**Mrs. Maryem RHANOUI**

**Promotion 2019-2020**

## Dedication

I dedicate this work

To my beloved parents, who have positively shaped the person I am today.

To my dear family, who supports me through the downs of life.

To all my friends who were there for me.

To my professors, who believed in me.

# Acknowledgments

I am endlessly thankful to my parents, who have dedicated all their time and care for me during the preparation of this project.

I want to express my infinite gratitude and respect to my supervisor **Mrs. Maryem RHANOUI** for her valuable help, advice, and for agreeing to guide me through this journey.

Special thanks for my internship director, **Mr. Hicham NAIT ZLAY**, and the whole Digimental team for making me feel welcomed. Their expertise sharing and support were much appreciated and valuable for the success of this project.

I am also grateful to all my professors in ***Ecole des Sciences de l'Information*** without whom I will not get where I am today.

# Abstract

This report summarizes the work done as part of the graduation project within ***DigiMental***. The project aim is to develop and deploy a web-based SaaS (Software-as-a-Service) app for hotels' internal management. Including deployment automation and DevOps best practices.

In 2019, the Morocco Tourism Observatory announced that nearly 7.544 million tourists had visited Morocco in late July 2019 (8% increase compared to the last year). Furthermore, experts forecast post-COVID-19 tourism recovery. To meet the demand, hotels find it challenging to manage critical daily tasks and coordinate between staff members.
Any delays or issues will lead to customer frustration, which will lead to reputation damage on booking platforms.

In particular, developing custom-made software that satisfies the hotels' needs require enormous time and capital investment, notably the forever-ongoing maintenance and infrastructure.

**Software-as-a-service (SaaS)** is a software distribution model of business solutions hosted and managed by the vendor and delivered to the end-user on a subscription basis, commonly via the web.

In recent years, the software-as-a-service (SaaS) model has been remarkably trending and becoming the favorite option for both business and customers. Admittedly, the democratization of cloud platforms made the development of SaaS apps easy and available for companies of any size and budget.

In this context, Digimental saw a significant opportunity in developing a SaaS solution and solving hotels' problems as part of its services.

*Keywords:* SaaS, Software-as-a-service, DevOps, CI/CD, Google Cloud Platform, Firebase, Angular, hospitality, management.

# Résumé

Ce rapport résume le travail effectué dans le cadre du projet fin d'études au sein de DigiMental. L'objectif du projet est de développer et de déployer une application SaaS (logiciel en tant que service 'Software-as-a-Service') basée sur le web pour la gestion interne des hôtels. Y compris l'automatisation du déploiement et les meilleures pratiques DevOps.

En 2019, l'Observatoire du tourisme du Maroc a annoncé que près de 7,544 millions de touristes avaient visité le Maroc fin juillet 2019 (soit une augmentation de 8 % par rapport à l'année précédente). En outre, les experts prévoient une reprise du tourisme après la tenue du COVID-19. Pour répondre à la demande, les hôtels ont du mal à gérer les tâches quotidiennes essentielles et à coordonner entre les membres du personnel.
Tout retard ou problème entraînera la frustration des clients, ce qui portera atteinte à la réputation des plateformes de réservation.

En particulier, le développement de logiciels sur-mesure qui répondent aux besoins des hôtels exige un énorme investissement en temps et en capital, notamment pour la maintenance et l'infrastructure, qui ne cesse de s'étendre.

Le logiciel en tant que service (SaaS) est un modèle de distribution de logiciels de solutions commerciales hébergées et gérées par le fournisseur et fournies à l'utilisateur final sur la base d'un abonnement, généralement via le web.

Ces dernières années, le modèle du logiciel en tant que service (SaaS) a connu une tendance remarquable et est devenu l'option préférée des entreprises et des clients. Il est vrai que la démocratisation des plates-formes dans le cloud a rendu le développement d'applications SaaS facile et disponible pour les entreprises de toute taille et de tout budget.

Dans ce contexte, Digimental a vu une opportunité significative dans le développement d'une solution SaaS et la résolution des problèmes des hôtels dans le cadre de ses services.

Mots-clés : SaaS, logiciel en tant que service, DevOps, CI/CD, Google Cloud Platform, Firebase, Angular, hospitalité, gestion.

## ملخص

يلخص هذا التقرير العمل المُتمم كجزء من مشروع التخرج في .DigiMental ويهدف المشروع إلى تطوير ونشر تطبيق web-based SaaS (Software-as-a-Service) للإدارة الداخلية للفنادق. بما في ذلك آلية التشغيل وأفضل تقنيات DevOps.

أعلن مرصد السياحة المغربي في عام 2019 أن هناك حوالي 7.544 مليون سائح قام بزيارة المغرب في أواخر يوليو 2019 (يوجد زيادة بنسبة 8٪ مقارنةً بالعام الماضي). علاوةً على ذلك، يتوقع الخبراء انتعاش السياحة بعد فترة فيروس كورونا. تجد الفنادق صعوبة في إدارة المهام اليومية الحرجة والتنسيق بين الموظفين لتلبية الطلب.

سيؤدي أي تأخير أو مشاكل إلى إحباط العملاء، مما سيؤدي إلى الإساءة بالسمعة على منصات الحجز.
على وجه الخصوص، يتطلب تطوير البرامج المُصممة خصيصًا لتلبية احتياجات الفنادق وقتًا طويلاً واستثمارًا رأس مال، وخصوصًا الصيانة والبنية التحتية المستمرة إلى الأبد.

Software-as-a-service (SaaS) هو نموذج توزيع برمجيات لحلول الأعمال التي يستضيفها ويديرها المورد ويتم تسليمها إلى المستخدم النهائي على أساس الاشتراك وعادة عبر الويب.

كان النموذج software-as-a-service (SaaS) رائجًا بشكل ملحوظ في السنوات الأخيرة، وأصبح الخيار المفضل لكل من الشركات والعملاء. من المُعترف به أن ديموقراطية الأنظمة السحابية جعلت تطوير تطبيقات SaaS سهلًا ومتاحًا للشركات بمختلف الأحجام والميزانيات.

رأت Digiment في هذا السياق فرصة كبيرة في تطوير حل SaaS وحل مشكلات الفنادق كجزء من خدماتها.

الكلمات المفتاحية: SaaS و Software-as-a-service و DevOps ونظام التشغيل السحابي من Google و Firebase و Angular و حسن الضيافة و الإدارة

## List of abbreviations

| Abbreviation | Designation |
|---|---|
| **API** | Application Programming Interface |
| **B2B** | Business-to-Business |
| **B2C** | Business-to-Consumer |
| **CI / CD** | Continuous integration / Continuous delivery |
| **CSS** | Cascading Style Sheets |
| **HTML** | Hypertext Markup Language |
| **HTTP** | HyperText Transfer Protocol |
| **IaaS** | Infrastructure as a Service |
| **JSON** | JavaScript Object Notation |
| **MVC** | Model-View-Controller |
| **NoSQL** | Not only SQL |
| **OTP** | One-time password |
| **PaaS** | Platform as a Service |
| **REST** | REpresentational State Transfer |
| **SaaS** | Software-as-a-Service |
| **SDK** | Software Development Kit |
| **SEA** | Search Engine Advertising |
| **SMS** | Short Message Service |
| **SMTP** | Simple Mail Transfer Protocol |
| **SOA** | Service-oriented architecture |
| **URL** | Uniform Resource Locator |
| **XML** | Extensible Markup Language |

## List of Tables

## List of figures

# Tables of contents

# Introduction

Based on research conducted on numerous online sources and white papers, this report tries to offer an insight into the current and future of software as a service (SaaS) business models in the industry.

Development of SaaS technology is presented regarding its definitions and models of deployment, outlining SaaS business models, and special attention given to service-oriented architecture (SOA) as a natural enhancement and extension of services provisioning. Additionally, providing insights into the different **SaaS development stages**, usage, and application.

The major benefits of the SaaS business model can be summarized based on end-users and service providers concerning financial value, enhanced operations, and new technologies. Software as a Service (SaaS) is software that is delivered, owned, and managed remotely by one or more individuals. The popularity of the on-demand deployment technique has and is still increasing significantly over the last decade.

Initially, concerns over **security**, service availability, and response time have been diminishing for many enterprises as SaaS computing models and businesses have matured, and its adoption widely accepted in various industries. In a SaaS model, the service provider is responsible for creating, updating, and maintaining the software. Subscribers subscribe to access the services, that include separate licensing or slot for each software user. A SaaS model adds **efficiency** and **cost-saving** for both the vendor and subscriber.

Customers save time and resources since they do not have to develop (in most cases), upgrade, or maintain programs. Additionally, they do not have to hire external staff to handle the maintenance or buy new hardware, thus, allowing them to focus more resources on growing the business. This **business model** shifts the burden of software development and deployment to the service provider and speeds up time customers take to obtain a return on investment on the same.

Service providers only have to maintain and update the software in a network infrastructure than focusing on updating the various software instances on various computer systems. This allows service providers to offer the latest updates and technology to every customer simultaneously. Currently, there is a growing spread of SaaS service providers with the key plays being Microsoft, Amazon, Google, IBM, Oracle, Apple, among others.

Software as a Service (SaaS) is a technique of **software delivery** that allows access to the software platform, and it features from remote locations over a **web-based** service. Therefore, SaaS can be viewed as a software distribution model where vendors or service providers host software instances over a network, usually the internet.

SaaS provides reprieve to organizations as they eliminate the **deployment**, installation, upgrade, and *regular maintenance* of software. Moreover, SaaS reduces the cost of business solutions as it offers organization access to business functionality at a subscription that is cheaper than application licensing.

Before reviewing the development and deployment of a SaaS service, it is worth reflecting on the SaaS industry right now. According to statics from the SaaSX[1] website, there is an estimated *60 percent* growth rate for SaaS businesses; however, this is still not enough to venture fully into the unicorn territory.

Additionally, SaaS is four times cheaper cross-sell or upsell to current customers than to deploy a **new instance**, and it is nine times cheaper to retain a customer in the service than to acquire a new one. On average, *92 percent* of startups and companies looking to diversify will be spending their first year on the acquisition.

Subsequently, SaaS businesses invest 80 to 120 percent of their revenue on sales and marketing in the first five years of operations. Moreover, businesses are becoming more reliant on SaaS services and tools – they expect these services to communicate with each other, exchanging important information accurately, quickly, and easily.

In the first chapter, we will explore the context of the study, the hosting company, and the planning of the project.

In the second chapter, we will dive deep into the technical analysis of the project. We are explaining various concepts behind the project.

Moreover, the third chapter covers the conception and design of the intended solution.

Lastly, we will cover the implementation and deployment of the solution.

[1] https://saasx.com/

## Problematic

In 2019, the **Morocco Tourism Observatory** recorded *7.544 million* tourists an 8 percent increase compared to the previous year. To meet the demand, the hospitality industry in Morocco found it challenging to manage critical daily operations and coordinate staff members. Tradition management models such as paper and spreadsheets led to significant issues in accuracy, integrity, availability, and confidentiality. This delay and inefficiency lead to customer frustration and, in worst-case scenarios, business reputation damage, and loss of business.

In particular, developing custom-made software that satisfies the needs of hotels requires enormous time and capital investment, notably the forever-ongoing maintenance and infrastructure. Besides, the traditional methodology of developing and delivering web-software has various limitations, primarily for this case. Indeed, the limitations made updates costly and software overpriced.

As SaaS reliance increases, there is a need for offering integrations that will increase the range of other software. These limitations and shortcomings offer a business opportunity for offering **Hotel Internal Management** services as an on-demand service through SaaS. This means the hospitality industry can benefit from the service as it will an on-demand model, where they subscribe per user or package and focus all their attention on other business elements.

In this context, Digimental is the company of focus – it saw this opportunity and is developing a SaaS solution that will save the **hospitality industry** time and solve the problems as part of its services.

To achieve this goal, Digimental will develop a modern, scalable, and deployable solution on the cloud. Additionally, implementing **DevOps** strategies and follow sustainable development models, following the best practices of web development and testing, while following the current and standardized **regulations** and **policies**.

This report presents the results of a systematic mapping on developing and deploying SaaS in the Morocco Hospitality Industry, mainly focusing on the development and deployment of the service in a cloud environment. Moreover, this report investigates the role of development, technology, and business aspects of SaaS deployment and taxonomy factors that should be considered while developing or establishing a **development model**.

# Chapter One: Project Context

Context is fundamental to understand the aims of the project. In the following chapter, we will explore the context of the study, the hosting company, and the planning of the project.

## 1.1 Presentation of the Host Company

### 1.1.1 Digimental

Founded in 2017 and based in Casablanca, DIGIMENTAL is a digital marketing agency in the hospitality business. It provides a variety of digital marketing services to its partners.

The company decided to partner with industry experts and high-growth challengers to brainstorm and develop ideas that fuel new ways of doing business. It is a creative group of product design innovators who work directly with customers to understand their situation and what they want to accomplish.

Digimental helps hotels and travel agencies to get noticed online, get more bookings, and improve their online reputation, whether in search engines, social medial, or booking websites. It provides marketing and communication strategies that allow the development of "haute couture digital campaigns" designed with and for its customers.

### 1.1.2 Offered services

Digimental offers numerous digital marketing services[2] to its clients, including the following:

- **Search Engine Advertising (SEA):** Search engine advertising (SEA) is a branch of search engine marketing. It is a sponsored text ad that is displayed either above, below, or to the right of the search results on several search engines, like Google, Yahoo, and Bing. Digimental enables its client to get high-converting traffic with the minimum advertising budgets by leveraging the power of analyzing tools provided by most search engines.
- **Social Advertising:** It is the process of creating, managing, and deploying ads on various social media platforms. These ads help hotels and travel agencies to get highly interested leads. Additionally, Digimental makes it easy to build a following base on social media and increase post engagements.
- **Web design and development:** Digimental helps companies bring their ideas and visions to life. From product design to deployment and testing with real users. It's the process of building user-centered products through user interface design, user experience design, user research, mockups, data analysis, and prototypes. It's all about design thinking and problem-solving.

[2] http://digimental.ma/

- **Search engine optimization (SEO):** The process of optimizing a website and its content in order to take leading positions in the organic search results of major search engines. Understanding how search engines work, what people search for, and why and how people search is vital for SEO. Successful SEO is making a website appeal to both search engines and users. It is a mixture of technical and marketing thinking.

### 1.1.3 Principles of DIGIMENTAL

Each moment is an occasion for discovery and inspiration, curiosity, and creativity. With each experience, hotels and travel agencies have a chance to make an impact. At DIGIMENTAL, linking these experiences together is the real point of difference. In doing so, the company creates moments that leave a lasting impression.

- **Business innovation:** Today's world is continually changing. According to this principle, the company helps its partners to thrive amid changes in trends, technology, and consumer expectations. As a brand consultant, Digimental empowers companies with the tools, resources, and knowledge to adapt and evolve with the world around them. The DIGIMENTAL team assesses existing campaigns, technologies, and internal processes to develop feasible transformation strategies. It guides all the players in the hospitality industry in the right direction by discovering revenue sources, solving digital challenges with creative solutions, and establishing transformation measures to keep up with growth.
- **Data-centric products:** The product should not only be elegant; it should be user-centered and frictionless. Modern consumers are looking for clean, responsive, and personalized experiences. DIGIMENTAL develops the best digital products in their category to meet their needs by adopting an idea-driven approach for each project. The team collects and analyzes consumer data to understand each channel's role and uses iterative processes to create, test, and refine deliverables to develop cross-channel experiences that connect touch-points.
- **Integrated Marketing:** As a data-driven agency, DIGIMENTAL understands the modern consumer. Moreover, as a brand partner, it creates integrated campaigns to reach and interact with them. The team builds brand value by creating holistic, customer-centric experiences that encourage interaction. Brand growth is not just about understanding, where audiences are but anticipating specific desires at distinct times.

## 1.2 Scope of the Project

SaaS business model has flourished in the last decade thanks to business benefits and economy of scale that it brings to businesses in all industries. Strategic benefits including lower initial costs, seamless integration of Enterprise Resource Planning (ERP), high adoption rates, and provider-managed upgrades and updates.

This document's scope presents a compelling report on the development and deployment of a fully functioning SaaS application for the Morocco hospitality industry that can be integrated with other systems and used to manage their staff more so interns. Since SaaS uses a distributed computing architecture, this report will identify tools, development methodologies, third-party services, and cloud platforms used in the development of the same.

The focus will be on the backend development as DigiMental has already covered the UX and UI design. The Backend Development of the SaaS service is the bulk of the development process and will employ some of the best DevOps development methodologies.

## 1.3 Limitations

The drawback to customers is that they do not control the full software and limited on programs' customization. Drawbacks, in this context, are categorized into two technical and non-technical. The non-technical limitations include building trust with customers through regular follow-ups and better management relationships. Since it is a new service, many will shy off from adopting it as there will be uncertainty as it is with every new technology introduced in the market.

Before the SaaS solution developed gets a firm market grip, it will need robust marketing and sales. This means the first quarter will be slow. The focus will be on getting the first customer and, from their positive experience, recommend the SaaS application to their stakeholders and partners. Technical limitations include interoperability – a significant challenge that continues to hamper SaaS business adoption by enterprises. This is because most businesses prefer real-time interoperability between SaaS applications and private cloud.

Another technical limitation is lack of integration support – how well will the SaaS application integrate with another is crucial as interoperability. For example, many of the hotels in Morocco are still running on-premises solutions, which will present challenges when integrating with the cloud-based solution because designing a hybrid setup is more complicated than developing a private one.  Less customization is another limitation of the project. An on-premises solution that most of the hotels use and rely on tends to offer more customization than cloud-based solutions because they come bundled with numerous SDKs.

Nevertheless, the SaaS application will offer a degree of customization capabilities through a common toolset of development languages, database operating systems, and middleware. Data exchange challenges is another limitation – data exchange between various systems can be an arduous task. The solution must be fast; this means fast but accurate data exchange by optimizing data exchange processes that require a proper understanding of the existing problem and defining the data exchange process.

This will slow the deployment of the SaaS solution and, in some cases, cause reputable damage to businesses due to the unavailability of the service. The next section of this report will focus on the technicality of the project – how to develop the SaaS business application, tools used, DevOps development methodology used and deployment model for the service.

## 1.4 Project Planning

The project is based on the **Scrum project management** method, which is an agile method of project management. It has allowed us to define sprints with specific functionalities to be developed and delivered after the deadlines for each sprint have elapsed.

Scrum allows teams to collaborate effectively and split big projects into smaller pieces (**sprints**), which are more manageable. During a sprint, the whole team focuses on that sprint. Teams have the flexibility to adapt and change their approach to the project by working on each sprint separately and improving the project as it progresses.

A Scrum project generally starts with what is often called the **"Sprint 0"** dedicated to the preparatory work for the project (e.g., construction of the "product backlog" and the vision of the product). The idea is to get started without first developing a comprehensive plan and architecture that could risk being flexible and costly in the short and long terms.

The sprint planning meeting is the most significant event in Scrum. The purpose of this meeting is to prepare the work schedule and identify the sprint backlog.

Given this project, it is necessary to conduct a study of the progress of the project and to integrate all of the pre-established functionalities.

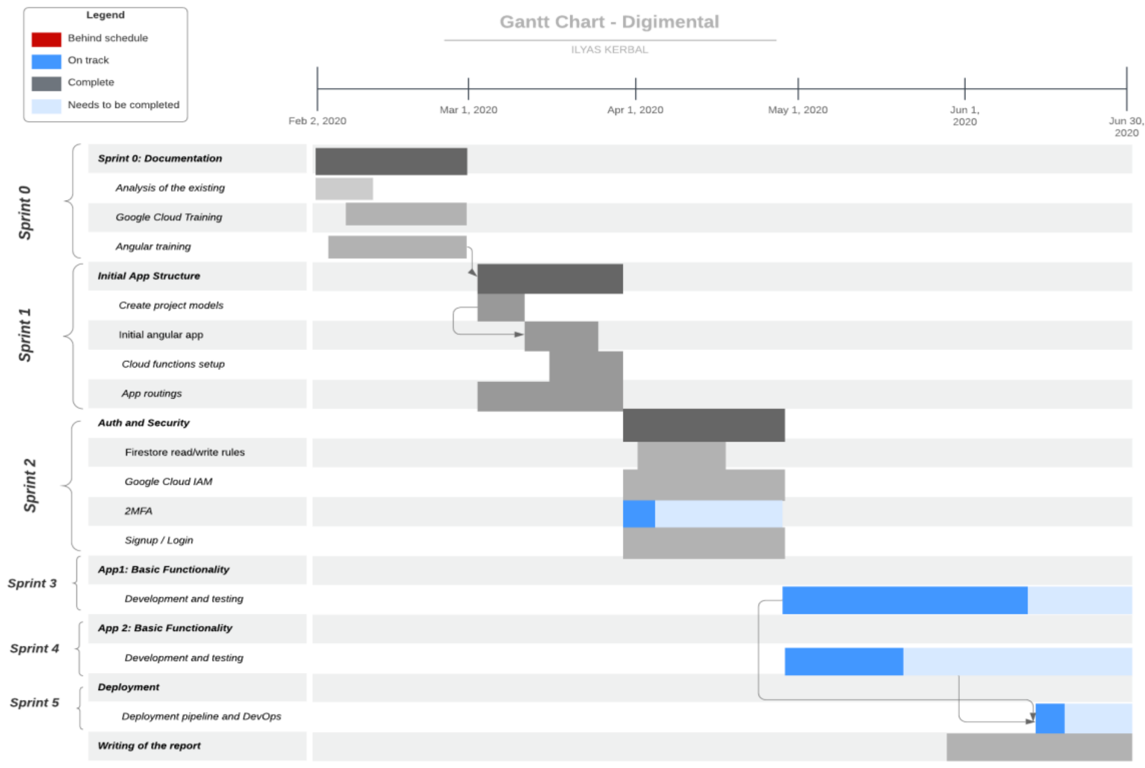


*Figure 1: Project Gantt Chart*

# Chapter Two: Technical Analysis

As the previous section has briefly introduced, this report is the documentation for developing a hotel internal management SaaS application for the Morocco Hospitality industry. This section will review the project's technical analysis by examining back-end development, database design, and third-party integration.

## 2.1 The Technical Aspect of the Project

### 2.1.1 Introduction

This section defines and reviews Software-as-a-Service (SaaS) in more detail. In this context, Software-as-a-Service (SaaS) is an application licensing and delivery model that many software vendors in both **B2C** and **B2B** markets. Moreover, SaaS is a licensing and delivery model offer service consumers remote access to both proprietary and open-source software subscription basis rather than purchasing software license and installing every software instance on various computer systems.

In this context, the software is developed, deployed, owned, and managed by a service provider (DigiMental). This rational economic analysis of the SaaS service has convinced service consumers and developers to shift towards the SaaS business model. For businesses, the service-oriented model, compared to the legacy software product development, offers benefits concerning revenue and profit as well as consumer retention and acquisition.

Consequently, both B2C and B2B markets, the new service model promises significant benefits for consumers, such as scalability to compatibility and remote access as well as and cost reduction. These benefits ensure the rapid diffusion and adoption of the model in the market. As such, SaaS is closely related to the application service provider (ASP). Therefore, the DigiMental SaaS hotel intern management application will be a software on-demand SaaS model. This means the service provider (DigiMental) will offer customers (hotels) network-based access to a single copy of a software application created explicitly for SaaS distribution.

Moreover, the source code will be the same for all the subscribers, and when new features are rolled out, they will be the same and available to all customers. Another feature of the proposed SaaS hotel intern management application is to allow organizations with other software using application programming interfaces (**API**) to integrate (discussed in more detail later in this report).

***Figure 2*** illustrates the variations between the SaaS model, PaaS, IaaS, and On-premises. The orange color represents the component managed by the service provider, and the blue color expresses the elements that must be handled by the user.

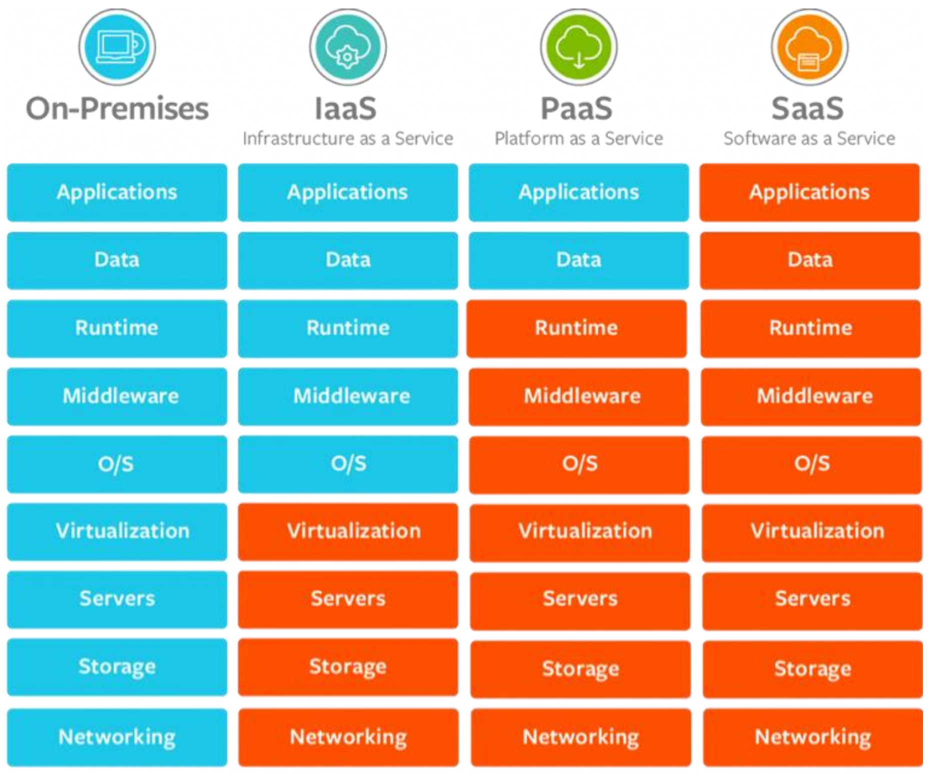


*Figure 2: The various on-demand services (Source: https://www.bmc.com/blogs/)*

Subsequently, the development of the SaaS hotel intern management application will follow the twelve-factor app methodology for its development. The **twelve-factor app**[3] uses a declarative format for setting up automation while minimizing time and costs for new developers joining the project. This implies it is easy to integrate methodology and will require little to no training for new developers joining the project.

Additionally, it offers a clean integration with the SaaS platform's underlying operating system, offering maximum portability between execution platforms.
This is highly crucial as SaaS is cloud-based – running on distributed computing environments with different servers, operating systems, and third-party peripherals. This means the twelve-factor methodology applies to applications running any programming language and using a combination of backing services such as databases, memory cache, queue among others.

Subsequently, the twelve-factor methodology is suitable for deploying modern cloud applications, as it obviates the need for servers and system administrators.
Furthermore, it minimizes divergence between development and deployment, allowing continuous deployment for maximum scalability. The twelve-factor methodology used for this SaaS application development and deployment is discussed in detail in this section.

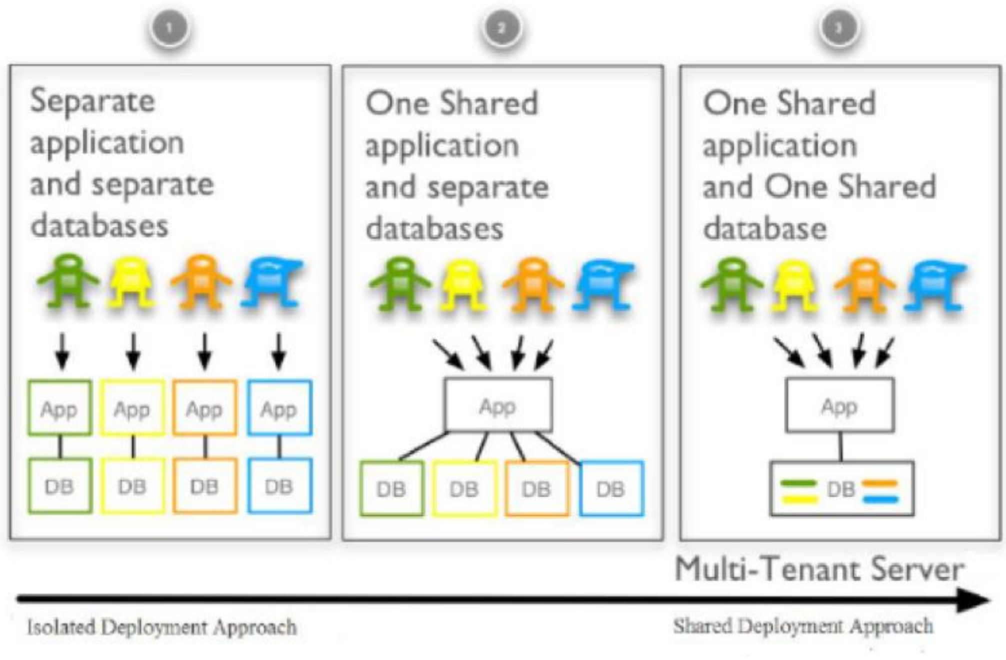
[3] https://12factor.net/

### 2.1.2 Multi-tenant vs Single-tenant SaaS

In this context, there are multiple hotels all will access and share the same resources – the SaaS application will be multi-tenant SaaS.  Multi-tenancy is a core benefit of adopting SaaS; in earlier days, businesses were reluctant to adopting cloud-based solutions. Few enterprises considered adopting technologies, controls, and policies to secure their data across cloud platforms.

However, in the last decade, the proven efficiency of cloud deployments in security, scalability, and cost has changed how enterprises view cloud technology. In this context, multi-tenant SaaS is a single software instance and its supporting infrastructure serving multiple customers simultaneously while single-tenant SaaS is a single software and its supporting infrastructure serving a single customer.

In a single-tenant- SaaS model, each customer has their independent database and software instance – no sharing with this option. While the multi-tenant SaaS, each customer shares the same software instance and supporting infrastructure such as databases. However, in this architecture, each tenant's data and presence remains invisible to other tenants.



1
2
3
Separate application and separate databases
One Shared application and separate databases
One Shared application and One Shared database
App
DB
Multi-Tenant Server
Isolated Deployment Approach
Shared Deployment Approach

*Figure 3: Multi-tenancy SaaS models*
Source: https://www.researchgate.net/figure/Multi-tenancy-models-10_fig1_311922746

The benefits of choosing the multi-tenant SaaS model in terms of performance is it offers lower costs via economies of scale, this means in a multi-tenancy model, scaling requires fewer infrastructure implications than in a single-tenant SaaS model because new customers get access to the same software and infrastructure.

Additionally, multi-tenant SaaS models are developed with high configurability so that customers can make the software perform the way they want, unlike in a single-tenant model where code or data structure must change, making the upgrade process complex and time-consuming.

Moreover, the Multi-tenant model offers a larger computing capacity – provides businesses with the ability to stay on the same data center and infrastructure. Thus, businesses will not have to rethink about adding more computing capacity or storage.
Though a multi-tenant model leaves multiple access points that may be vulnerabilities to cyberattacks, the model will be using authentication and Single Sign-On (SSO) isolation for user authorization and verification.

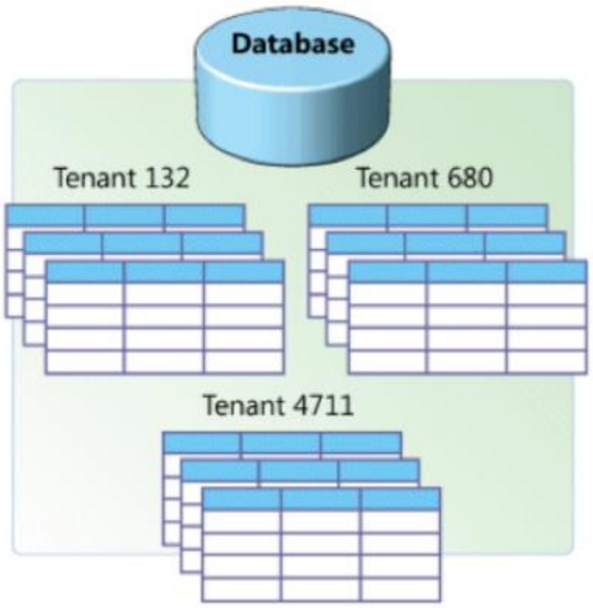


*Figure 4: Multi-tenancy model of the shared database and separate schema*

Source: https://www.researchgate.net/figure/Multi-tenancy-models-10_fig1_311922746

Most of the security measures will be on the database and schemas to ensure integrity, confidentiality while offering availability.

### 2.1.3 The Twelve-Factor App[4]

As mentioned in the introduction, the project is based on The Twelve-Factor App methodology. The methodology helps create highly-scalable web-based SaaS applications by respecting particular patterns and best practices. It serves as a framework for the entire application from codebase to services and processes. In the following sections, we will cover some essential aspects of The Twelve-Factor App.

[4] https://12factor.net/

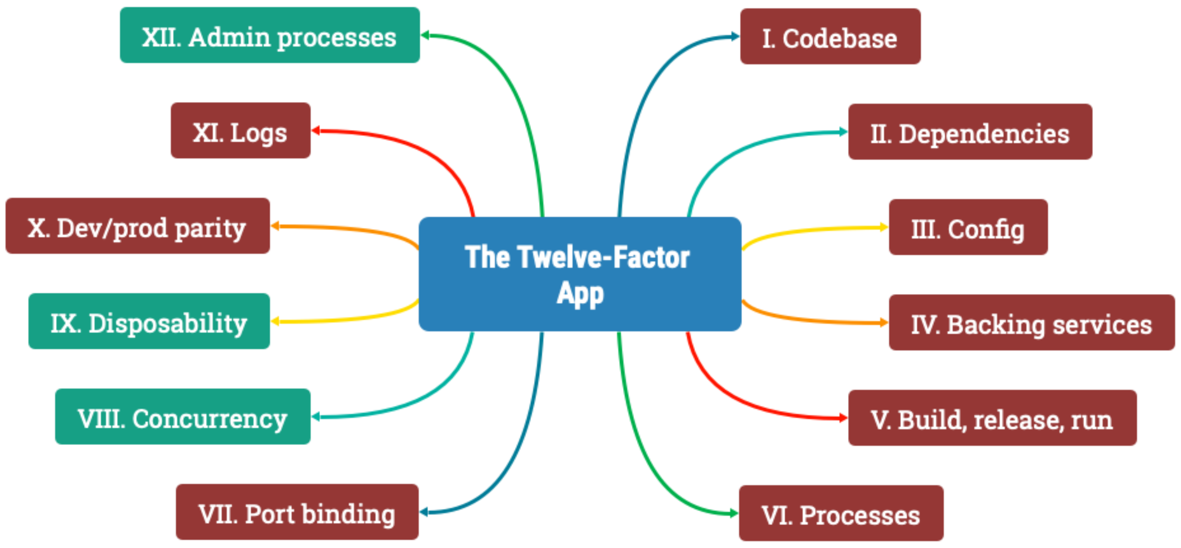


*Figure 5: The Twelve-Factor App*

### 2.1.3.1 Codebase / Version Control System

In this context, version control is the categorization of software tools and updates that help a software team manage changes to their source code during the development and deployment of the software. As such, version control software and methodologies keep track of every change in the source code. If a mistake occurs, developers or project managers can trackback the software version to its earlier version(s), thus help fix the mistake and minimize disruption of the software development process.

For all applications, the source code it the crown jewel – a previous asset that must be protected at all times. Therefore, version control protects the software's source code from both casual and catastrophic degradation of untended consequences and human error.
Software developers working in teams usually write new source code and change the initial source code. Source code in this context is usually organized in folders and subfolders in a file tree structure. As such, a developer or team may be working on a new feature or fixing a bug while another may be adding new configuration features that change the base code. Each developer or team will be changing several parts of the file tree. Version control helps solve these kinds of challenges – tracking every change by a contributor, thus helping prevent concurrent works from conflict.

Moreover, changes made to one part of the software may be incompatible to other contributor's changes working simultaneously. Therefore, a reliable version control supports a contributor's preferred workflow without imposing a particular methodology. Ideally, it works on any platform, rather than dictating which platform or contributors must use.

The **twelve-factor application** will always be tracked by a version control software such as Git, Bitbucket, or Subversion. In this context, a code repository (code repo for short) is a copy of a revision tracking database.

#### 2.1.3.2 Dependencies and Package Managers

Most, if not all, programming languages have a packaging system for distributing its support libraries. Therefore, a packet manager is a tool that programming languages use in creating project environments and accessible import external libraries. When working in a project or library, one may package the project and publish it; for example, this project will be running the JavaScript programming languages which use the npm package manager.

**Npm**[5] is a package manager for JavaScript. It is the leading software repository. Npm host popular packages such as jQuery, NodeJS, Angular, React, and Bootstrap among others. Additionally, it links to version control tools such as Git, therefore, allowing developers to create and share projects. The online npm repository is vast and diverse; as such, NodeJS backend and JavaScript front-end developers use it as the packages can be used in any environment.

Since the project will be using the package manager, this means the SaaS application will be depending on the package manager's libraries and repositories. As such, the **twelve-factor application** does not rely on the implicit existence of system-wide packages; therefore, all dependencies will be declared entirely and precisely through a dependency declaration manifest. Also, it employs that dependency isolation tool during the execution, ensuring that no implicit dependencies leak from the immediate ecosystem. A benefit of using this approach is the simplification of the setup for new developers to the project.

#### 2.1.3.3 Build, Release, Run Stages

The **twelve-factor application** includes strictly separate build and run stages. This means there is a strict separation between the release, run, and build stages of the **codebase**. The **build stage** is a transformation that converts code repo into an executable bundle – build. Version control, in this instance, commit specified deployment processed, as such, fetching vendor dependencies and compiles assets and binaries.

While the **release stage** accepts the build produced in the build stage and integrates it with the current config. The release contains both config and builds and is ready for execution. The third and last stage, run stage (runtime), runs the application in the execution environment by

[5] https://www.npmjs.com/

launching some of the application's processes in contrast to selected releases. That is why it is difficult to make changes during runtime, since, there is no way of propagating these changes back to the build phase.

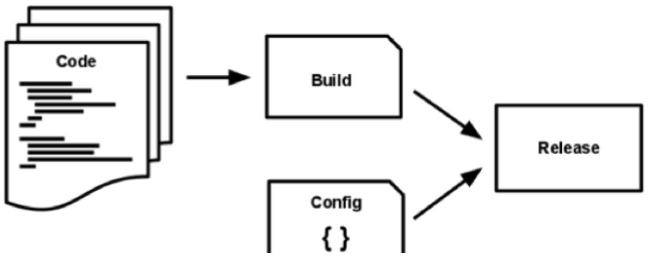


*Figure 6: Twelve-factor application build, release, and  run model*

*Source : [https://12factor.net/build-release-run](https://12factor.net/build-release-run)*

#### 2.1.3.4 Processes in SaaS Apps

The twelve-factor application processes are stateless – share-nothing. This means that any persistent data must be stored in a stateful backing service – database. Therefore, the SaaS application can be executed as one or more stateless processes. For example, if the code is a stand-alone script, and the execution environment is a developer's computer with an established language runtime, and a process is launched via the terminal (for example, node lexect.js).

Moreover, a production deploy of sophisticated applications utilizes many processes types, with some instantiated into zeros or more running processes. The filesystem or memory space of the process can be utilized as a brief – single-transaction cache. For example, downloading large files, operating them, and storing the results of the operation in a database.

The **twelve-factor model** never assumes anything in the memory or on the drive will be available on future job or request. With multiple processes of each version running, there are high chances that a different process will serve a future request.
Even if one process is running, and restart (either by config change, code deploy, or execution environment process relocation to a different physical location) will erase all local state (for example, filesystem and memory). Subsequently, in the twelve-factor model, stick sessions are not permitted.

#### 2.1.3.5 Backing Services

In this context, backing services are services that application consumes in a network environment as part of its regular operation. For example, a database such as MySQL, caching system (for example, Memcached), SMTP services for outbound email (for example, Postfix), and messaging/queueing systems (for example, Benstalkd or RabbitMQ).

A cloud-native application backing services include nay service that is communication over the network. This can be reviewed as attached services, such as database connections, email services, SFTP, file-sharing services, cache providers among others.

In the twelve-factor application model, remote services are treated the same as local services; thus, the core of this model. This is a crucial factor in reducing code complexity when dealing with network resources. In this case, a framework will be used, assuming the service is bound for the application. That is why the backing services are bound by the URL to the local or remote resource identically.

Additionally, the URL will be offered by the configurations managed by rules defined in the previous factor – configuration externalizations. Moreover, to make the code cleaner, backing services can be treated in a manner that allows swapping out of the services in every data center or environment to match the system needs. The developer often finds himself writing proof-of-concept code with a locally embedded database service and swapping the design during deployment – database as a backing service. The best part of this technique is as long as the twelve-factor model is used, this type of backing service swapping can be completed without doing any work besides creating the service itself. The power of this type of simple backing services swapping is to expand the operations model.

In summary, a typical workflow for the twelve-factor application model includes building databases in production, stopping the service and switching to bindings, and restarting the service.

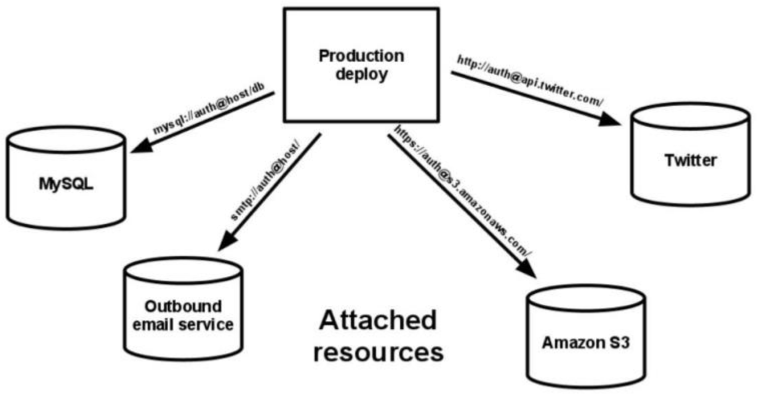


*Figure 7: Attached Resources*

*Source : https://12factor.net/backing-services*

However, a typical database change workflow for a monolithic application starts by modifying the code, then testing in lower environments, then building the database in production and keeping it in sync, after the code is deployed, and crossing figures. For example, moving from a plain text database environment to an encrypted one. This move will require redeployment of legacy applications; however, in the SaaS environment, the service is built, bind to the application, and restarted. This is important when the entire system is not running as expected; in the SaaS environment, the backing service can be replaced.

However, moving legacy code to this environment will be challenging at times, that is why the SaaS application relies heavily on frameworks designed to leverage the migration of the legacy application to the native cloud environment. As well as looking into existing patterns based on the programming language being used. Most, if not all, have patterns that have been implemented to solve this.

The SaaS application will assume that the connections to the database are injected into the code by a container. In this development model, when an equivalent one swaps the SaaS application backing service, the app's traffic will automatically move to the new service.
Thus, the development model builds the backing services of the application during the runtime instead of compile-time; this way, the whole ecosystem becomes flexible.

#### 2.1.3.6 Config Variables

In any application, config is everything. The project is developing and deploying a web-based service. The config in this context is what makes the entire system (network, databases, and dataset schemas, third-party APIs, configuration files, data centers, operating systems, and other IT assets work together. It includes resource handles for the database service, Memcached or any other **backing services**, credentials to **external services** such as Facebook or Google Cloud, and per-deploy values such as the canonical hostname for deployment.

Consequently, it is managed carefully and monitored keeping track of the configuration changes ensuring traceability or the system end-users will suffer data leaks, system outages, and data leaks. That is why some developers store config strictly separate from the source code.
Every deployment has its config. For example, to ensure an app's config is properly factored out of the code, the codebase can be made open-source without **compromising the credentials**.

In this context, config does not include any external app config, for example, configs/config.js in JavaScript or how the code modules are connected. Another technique is using config files that are not factored into revision control, for example, config/database.js in JavaScript. This is a significant enhancement overusing constants which are factored into the code repository but are still vulnerable; it will be easily be mistakenly checked in a repository – there is a tendency of placing config files in different files or formats. This makes it harder to identify and manage all config in one location. Furthermore, these formats tend to be framework or language-specific.

The twelve-factor application model stored config in the environment variables (env or env vars). Env is easy to modify when deploying between environments without changing the underlying code, unlike config files that have a slim chance of being checked accidentally. In the twelve-factor app model, env vars are coarse controls, entirely orthogonal to other env vars.

#### 2.1.3.7 Port Binding in SaaS Apps

At times, web applications are executed within webserver containers; for example, PHP application may run as modules in the Apache webserver container, Java-based application in Tomcat, or NodeJS applications in the Node.JS server.

The *twelve-factor app* model is fully self-contained and does not depend on runtime injection by webservers into the execution platform to create web services. In this context, the web application will export HTTP as a service by binding to a port and listening to requests from that port.

Alternatively, in a local environment, developers may visit a URL (for example, http://127.0.0.1:5000) to access the services exported by their applications. During deployment, a routing layer handles the requests. This is an example of a typical implementation through dependency declaration – adding a webserver to the web app library, for example, Tornado for Python. However, HTTP is not the only service that is exported during port binding.

In this context, container-determined ports are avoided. Web apps, especially enterprise-based applications, regularly execute within some form of server container. While others might run inside containers, such as the *Microsoft Internet Information Server (IIS)*. In a non-cloud platform, web apps are deployed to such containers, as the container is responsible for port assigning and listening during startup. In a cloud-based environment, where multiple applications are hosted in the same container, application URL hierarchy or port number are separated. Then a DNS is employed to offer user-friendly disguise around the server environment. Instead of making the web app users remember the port numbers, the DNS associate automatically handles this.
In this environment, one web application can be a backing service of another web app, by including the URL to the backing application and handling the resources via the config for the consumer web app.

#### 2.1.3.8 Development, Staging, and Production Environment

Traditionally, there have been significant gaps between production (running an app deploy accessible by end-users) and development (a developer making live edits on a local app deploy). These gaps can be identified as a time gap – a developer working on code that takes time to go into production, personal gap – a developer writing code and an Ops engineering deploying it, and tools gap – a developer may use a stack like Nginx, OS X, and SQLite. In contrast, in production, deploy the application using Apache, Linux, and MySQL.

Using the twelve-factor app model in this context, the app architecture is designed for continuous redeployments by reducing the gap between production and development. Reviewing the three gaps mentioned, the twelve-factor app model reduces gap by making the time gap small – a developer may write code and redeploy it hours of minutes later, make personal gap small – developers who wrote the code are involved in the deployment and monitor it during production, and make tools gap small by keeping development and production similar.

#### 2.1.3.9 Logs and Log Management

Logs are part of every system. They offer visibility of the behavior of the web application, server environment, and user movement. They are commonly written in files in the drive (log or log file). However, this is only an output format.

Logs are streams of accumulated time-order events from output streams, backing services, network port forwarding, user movement logs. Subsequently, there are no beginning or endings unless the web application is permanently pulled down, for one reason or another. Logs are typically generated in their raw form and stored in text files one event per line.

Additionally, they are continuous as long as the web app is operational. The twelve-factor app model does not concern itself with storage or routing of its output streams, it does not attempt to write or manage log files. However, each running process will create and write its event stream unbuffered. Moreover, in a local environment, the developer views the output streams in the foreground while observing the web app's behavior.

### 2.1.4 NoSQL Databases

Not only SQL (NoSQL) is a non-tabular database that stores data differently from relational database tables. NoSQL is purposely building for specific data models and have flexible schemas, unlike relational databases. They are widely recognized for their ease of integration, functionality, performance, and development.

NoSQL databases use a variety of data models for managing and accessing data. Additionally, they are optimized specifically for a web application that requires and sue large data volumes, flexible, and low latency data models, which is achieved by relaxing some of the data consistency limitations of other databases. For example, a simple book database, in a relational database structure, a book record is often normalized and stored in separate tables, and primary or foreign key constraints usually define relationships between the entities. In a relational database structure, the entities are designed to ensure referential integrity between them is that the database is maintained.

In NoSQL, a book record will usually be stored in a JSON document structure; all book attributes are stored in the same document. Unlike in the relational database structure, NoSQL optimized data for intuitive development and horizontal scalability. Types of NoSQL databases include key-value databases – highly partitionable and scale horizontally. Document database – data is often represented as JSON, graph database – makes it easy to build and run applications that utilize highly connected datasets, and in-memory database – mainly for ad-tech and gaming applications.

### 2.1.5 Serverless Architecture

Serverless is the new trend in most development and server deployments that is gaining much attention from professionals and tech industry enthusiasts. The popularity is mainly due to the high adoption of technologies such as AWS, which are hyping cloud computing and serverless architecture.

In this context, serverless architecture is a cloud computing model where a service provider dynamically manages all provisioning and allocation of servers. A serverless application runs in a stateless execution container that is event-triggered and fully manages by the service provider. This web app development and deployment runs on the serverless architecture. Traditionally, web applications have been running on servers that needed to be patched, updated, and continuously looked after regularly due to the high probability of errors that would break the architecture.
In the serverless environment, the subscriber no longer needs to worry and focus on the execution and underlying servers. The reason is that they do not need to manage the server architecture anymore as the entire responsibility falls on the cloud service provider. However, in some cases, traditional architecture outperforms the serverless architecture.

**In this project**, serverless architecture is used for the payment and notification app. The pricing is pay-as-you-grow.

### 2.1.6 Containerization

In cloud computing, containerization has become a popular trend in software development as it is an alternative to virtualization. It involves packaging or encapsulating source code and its dependencies, so that runs consistently and uniformly on any platform. This is resulting in measurable benefits for developers. As such, containerization allows web app developers to create and deploy apps faster and more secure.

Traditionally, source code is generated in a specific computing platform which often results in errors or bugs when migrated to a new platform. For example, when a developer migrates a web app from a local computer to a virtualization environment or for a Unix platform to a Windows environment. Containerization reduces this challenge by building app source code together with its configuration files, dependencies, and libraries required for its execution. This single app installation is called a container and is abstracted from the operating system, therefore, standing as portable and able to execute on various platforms including the cloud.

### 2.1.7 Message Brokers

In this context, message brokers are programs that translate messages to a formal messaging protocol in the sender to a formal messaging protocol in the receiver. For example, the web app will be having multiple messages thousands, millions, or even billions. A message broker can be created to centralize the storage and processing of these messages so the application and users can work with the messages. Currently, there are many message brokers out there, such as service bus, Redis, JMS, OMS, RabbitMQ, Kafka, ActiveMQ among others.

The web app will be communicating with various third-party apps such as the payment and notification APIs, using a messaging middleware simplifies this process, and it allows common communications infrastructure that scales to meet the growing demand. Messaging middleware is a server-based model that employs the message broker. The defining characteristics of a message broker are a discrete service and applications, and users communicate with it using standard proprietary protocols.

A message broker can be seen as state management that tracks customers, so the application does not need to perform this task, and the message delivery's complexity is built into the message broker.

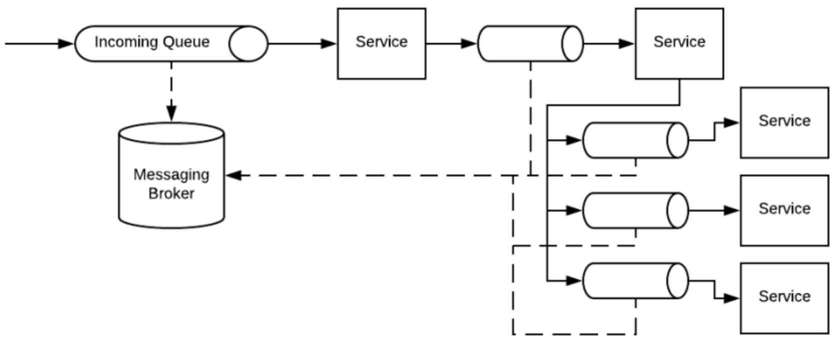


*Figure 8: Messaging Broker*

*Source :* [http://cpitman.github.io/microservices/2018/03/25/microservice-antipattern-queue-explosion.html#.XvsvMZNKjjA](http://cpitman.github.io/microservices/2018/03/25/microservice-antipattern-queue-explosion.html#.XvsvMZNKjjA)

### 2.1.8 WebSockets

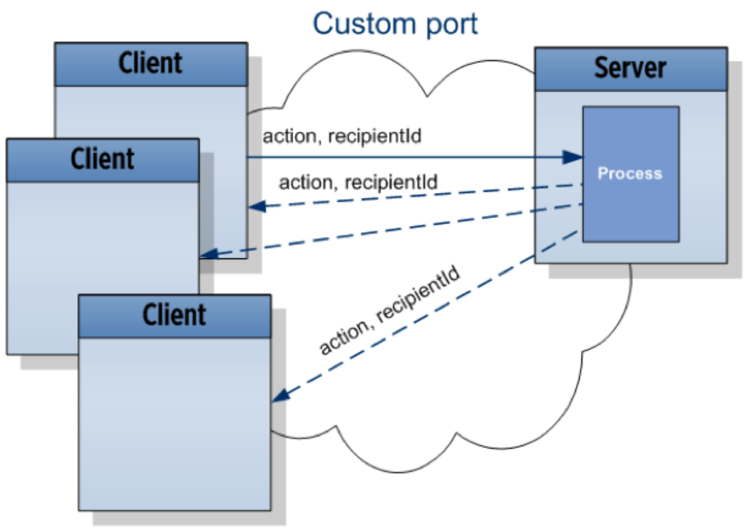


*Figure 9: WebSockets*

*Source: https://developer.mozilla.org/*

Mozilla Developer Network (MDN) defines WebSockets as a technology that allows opening interactive communications sessions between a web app and the server. This technology allows users to send messages to the cloud platform and receive event-driven responses without the need for long-polling – without having constant server checks for a reply.

For example, when replying to an email in Yahoo, and the bottom of the screen, there is a pop-up notifying the user of (2 unread messages from ‘sender’) coming from the sender that one is responding to. This kind of real-time feedback is due to technology such as WebSockets. Emphasizing the importance of WebSockets involves looking at HTTP long-polling, which originated from the need to send data to the sure without being requested. Without HTTP-polling, a customer may continuously poll the server (cloud platform) for new information and responds with the information once available.

Essentially, the customer constantly polls the cloud platform for information about the database state changing. As soon as the customer receives the information, it sends another request, and the operation becomes continuous. In WebSockets, single TCP socket connections are log-held to establish a link between the SaaS platform and web app, allowing for bi-directional, full-duplex, and constant messaging. Subsequently, this is done with minimal overhead resulting in low latency locations.

### 2.1.9 DevOps

**DevOps** is a form of software engineering culture and best practices aiming at unifying software development (Dev) and software operations (Ops). DevOps strongly advocates for

monitoring and automating all software construction steps, from releasing, integration, testing to infrastructure management and deployment.
DevOps aims to shorten development cycles, increased deployment frequency, and more stable releases, in close alignment with the business objectives. Its benefits include delivering a solution at higher speeds, allowing easy scalability, and increases reliability.

Maintaining software is generally categorized into two key concepts: cross-functional teams – rapid collaboration is crucial, no separation between operations, database, testing, and development teams. Secondly, automation – container-based technology aligned with microservice architecture, changing the way software is built, allowing full automation of software development cycles, including monitoring, deployment, testing, and development.

The entire SaaS platform can be fully automated. It is crucial to note that large monolithic applications and large micro-services are complex systems. The microservices do not simplify the development; in some instances, it may complicate it.

### 2.1.10 Continuous Integration / Continuous Delivery (Ci / CD)

Continuous integration (CI) and continuous delivery (CD) exemplify a culture, collection of practices, and set of operating principles that enable application developers to deliver changes more reliable and continuously. This is known as CI/CD pipeline. CI/CD is considered one of the best practices by DevOps teams. Additionally, an agile methodology best practice that enables web app developers to focus on meeting the SaaS web app requirements, security, and code quality because the deployment processes are automated.

Therefore, continuous integration (CI) is a development paradigm and set of best practices that drive developers to implement small modifications and checks to source code versions control repository frequently. Because modern web apps require the development of source code for the various deployment infrastructure, there is a need to integrate and validate the modifications.

While continuous delivery picks up from continuous integration. CD automates the deployment of web apps to selected infrastructure. CD ensures that there is a way of quickly automating the push code process. Moreover, CI/CD improves collaborations and quality as well as enabling frequent code deployments. CI/CD workflow allows the automation of each stage (verify, package, and release).

# Chapter Three: Conception and Design

We will dive deep into the conception and design of the intended solution as we covered the technical aspects of the project. This section will review the database modeling, DevOps pipeline, application design, and features.

## 2.1 Introduction

As mentioned in the first chapter, The focus will be on back-end and front-end development as DigiMental has already covered the UX and UI design, and the app specifications. This chapter covers everything from cloud architecture design to UML diagrams, and non-functional requirements.

## 2.2 Capturing non-functional needs

These are the needs that characterize the system. These are needs in terms of performance, type of cloud services, or type of design. Hence, these requirements will result in the choice of the stack, programing language, or cloud services.
The platform must first satisfy a set of technical requirements:

- ✓ It should be easy to extend, maintain, add, or modify new features and services.
- ✓ It must implement a user-friendly and intuitive interface.
- ✓ It must guarantee fast access to information.
- ✓ It must be highly secure because the information should not be publicly accessible.
- ✓ It must be protected against external and internal attacks.
- ✓ It must work on any browser and device - responsive design, broad browser support.

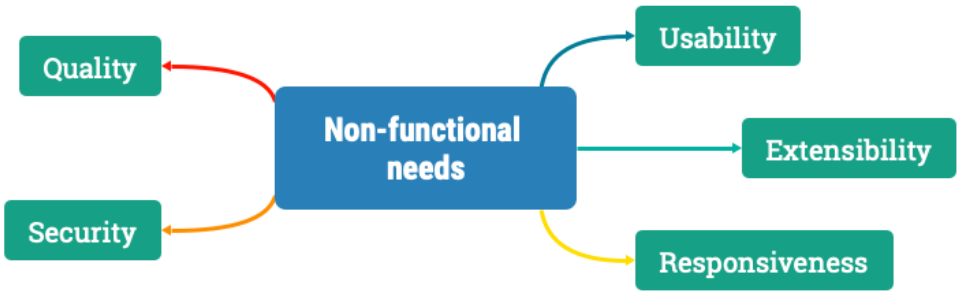


*Figure 10: Non-functional needs*

## 2.3 Identification of the main actors

An actor is a person, natural or legal who interacts with the system.
The analysis of this project begins with an identification of the actors acting on the different parts of the system.

*Table 1* summarizes the actors interacting with the system by specifying the functions of each of them before defining more precisely their interactions with the system using UML diagrams.

*Table 1: Main Actors of the Application*

| Actor | Function |
|---|---|
| **Hotels** | The manager of a hotel; they can manage all the assets, staff, and rooms of the hotel. They create and assign tasks to employees. They have an eye bird view of the whole operations at the hotel. |
| **Hotels' Staff** | User with an account associated with a hotel and accessing all the platform's functionalities and interacting with managers and other hotel staff. |
| **Digimental** | Superuser with all permissions (add, modify, and delete) on all models without exceptions via the administration interface. The only user allowed to manage payments and subscriptions. |

Developers, on the other hand, are not a user of the application. They have all the permission to manage and remodel the cloud structure of the project. They deliver new code, monitor the cloud, and fix issues that may occur during production. As we will cover later, they are mostly concerned with the DevOps pipeline.

## 2.4 Application Cloud Architecture Design

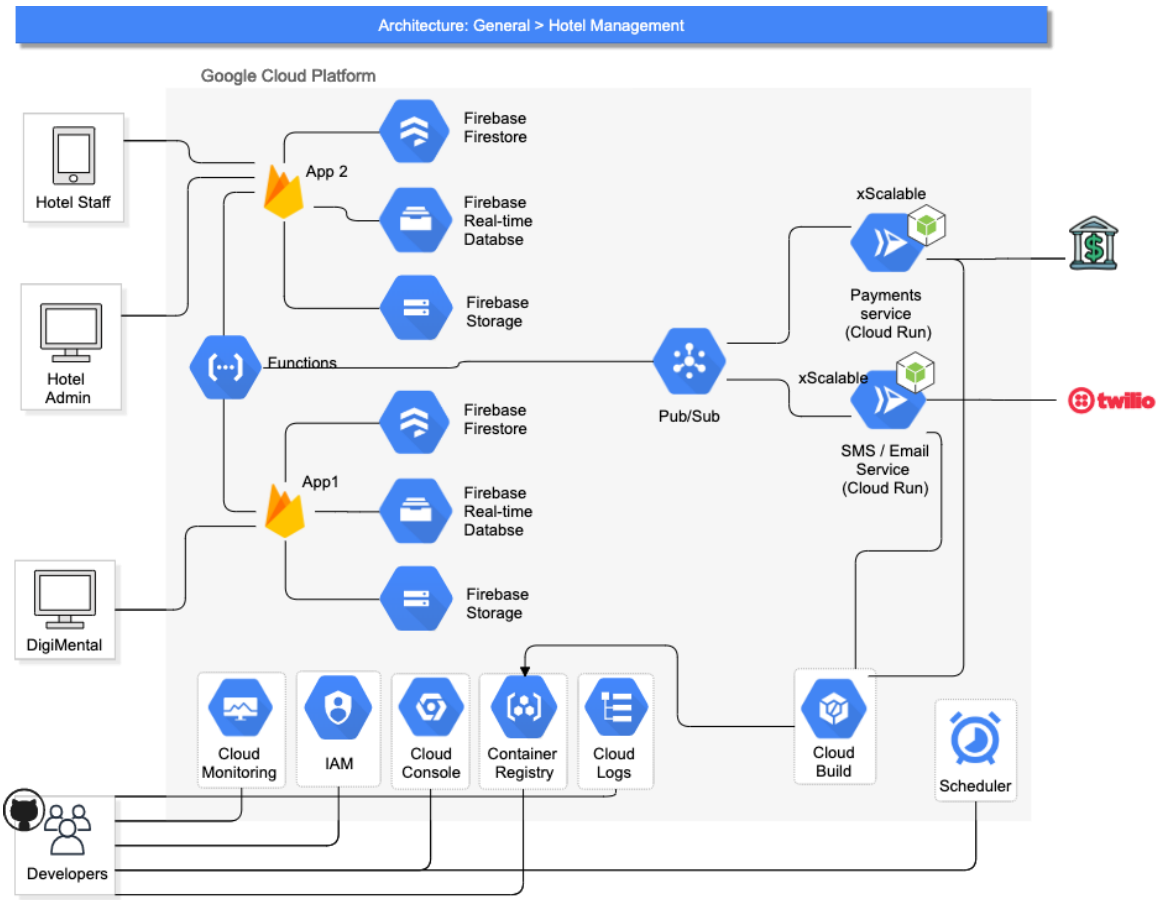


*Figure 11: Cloud Structure of the App*

The figure above illustrates the general cloud architecture of the app. The proposed architecture is not permanent and can change as the app evolves. Each change in the future will take into consideration the app growth, changes in the stack, or the number of developers, among others. Additionally, integrating Firebase with other Google Cloud products is straightforward; because any Firebase project is Google Cloud project by default. The general architecture will consist of the following:

- **App 1:** Digimental will use this app to administrate the whole system. It enables managing hotels, issuing invoices, and managing plans and subscriptions. Further, it contains a dashboard to monitor significant KPIs.
- **App 2:** This is the main app. Hotels and their staff will use it to manage critical daily tasks and coordinate among them.
- **Payment service (Cloud Run):** A NodeJs application runs in the Cloud Run to ensure scalability and high availability without wasting cloud resources. The app will handle anything related to payments and subscription validation.

- **<u>SMS/Email Service (Cloud Run):</u>** A NodeJS application, the same as the payment service mentioned above. It will handle outside communication with the users. It connects to Twillio through the official API. Hence, it must handle every request effectively and use the API reasonably.

Communication among the principal components of the architecture will go through the Pub/Sub and Cloud functions. For the bottom components, we will dive deep into the deployment pipeline later in this chapter. Developers can use those components to update the app code, monitor the cloud, and schedule operations.

## 2.5 DevOps Pipeline

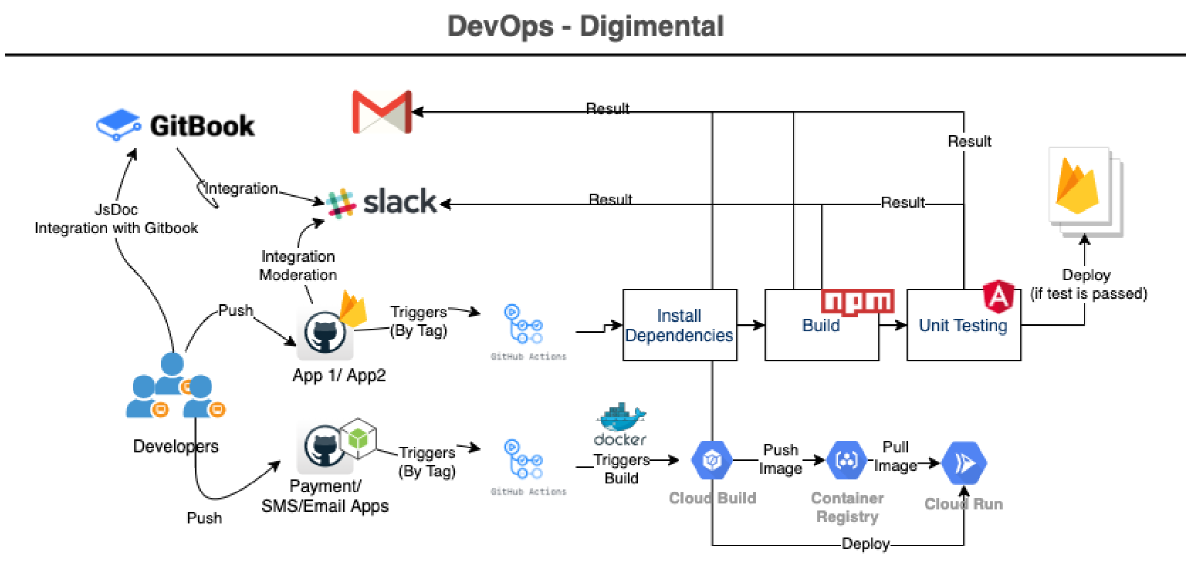
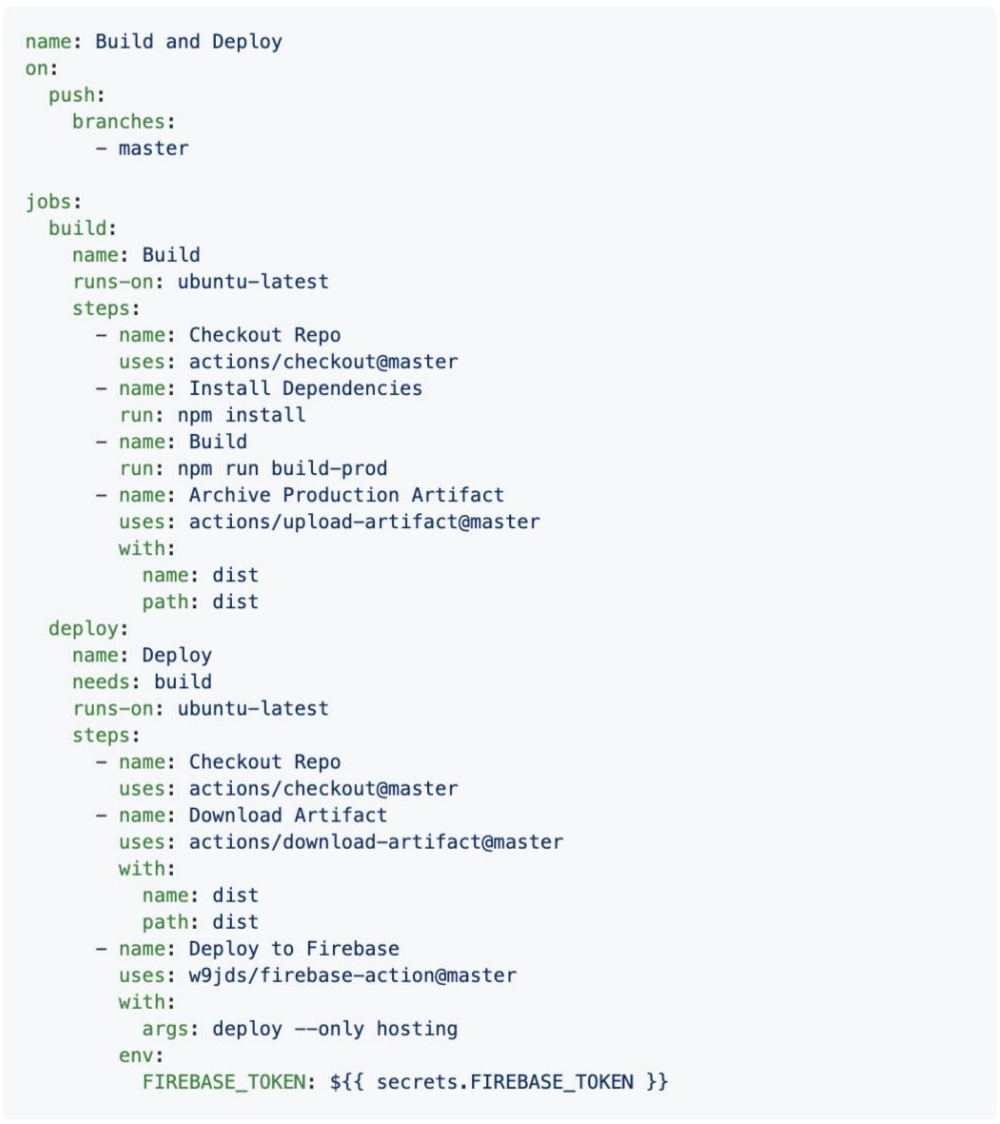


*Figure 12: DevOps Pipeline*

The DevOps pipeline is direct and straightforward. Developers will push new code to Github repository. After review by other developers, the code will be tagged, e.g., v1.8.5.1. The tagging will trigger the whole automated process of building, testing, and sending operation results through a Slack channel. After the new code is deployed, JsDoc is used to generate new documentation automatically through a GitBook integration.

Each Git commit goes through a review by a senior developer. Hence, the senior developer can tag new code and trigger new deployment and releases of the software. If the deployment process fails, the developers receive a slack/email notification with details of the failure.

However, this pipeline is not permanent. Future updates may occur when the app gets more users or a change in the stack.



```yaml
name: Build and Deploy
on:
  push:
    branches:
      - master

jobs:
  build:
    name: Build
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repo
        uses: actions/checkout@master
      - name: Install Dependencies
        run: npm install
      - name: Build
        run: npm run build-prod
      - name: Archive Production Artifact
        uses: actions/upload-artifact@master
        with:
          name: dist
          path: dist
  deploy:
    name: Deploy
    needs: build
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repo
        uses: actions/checkout@master
      - name: Download Artifact
        uses: actions/download-artifact@master
        with:
          name: dist
          path: dist
      - name: Deploy to Firebase
        uses: w9jds/firebase-action@master
        with:
          args: deploy --only hosting
        env:
          FIREBASE_TOKEN: ${{ secrets.FIREBASE_TOKEN }}
```

*Figure 13: Github Actions Firebase Build and Deployment Scrip*

## 2.6 UML Modeling

### 2.6.1 Use case diagram

Use case diagram describing, in the form of action and reaction, the behavior of a system from a user's point of view. It allows us to define the functioning and the limits of the system as well as the description of the interactions between the actor and the system. It, therefore, allows him to achieve his objective by using it.

Therefore, it is necessary to identify all the functionalities which the various actors concerned by the package will need. The "include" relationship is used to enrich a use case with another use case (it is a sub-function). The inclusion relationship is imperative and, therefore, systematic. Like the inclusion relationship, the extension relationship enriches a use case with another use case for an optional sub-function. The figure above illustrates the different use cases for our project.

As covered previously, we have three different actors, each with specific actions. Some actions, like subscriptions and invoicing, are included or extended in other actions. Let us take the example of "Contacting Support," it extends subscription to assign priority to support tickets based on plans. Additionally, each action performed takes into account the role and permissions the user has, as many users will use this SaaS app at the same time.

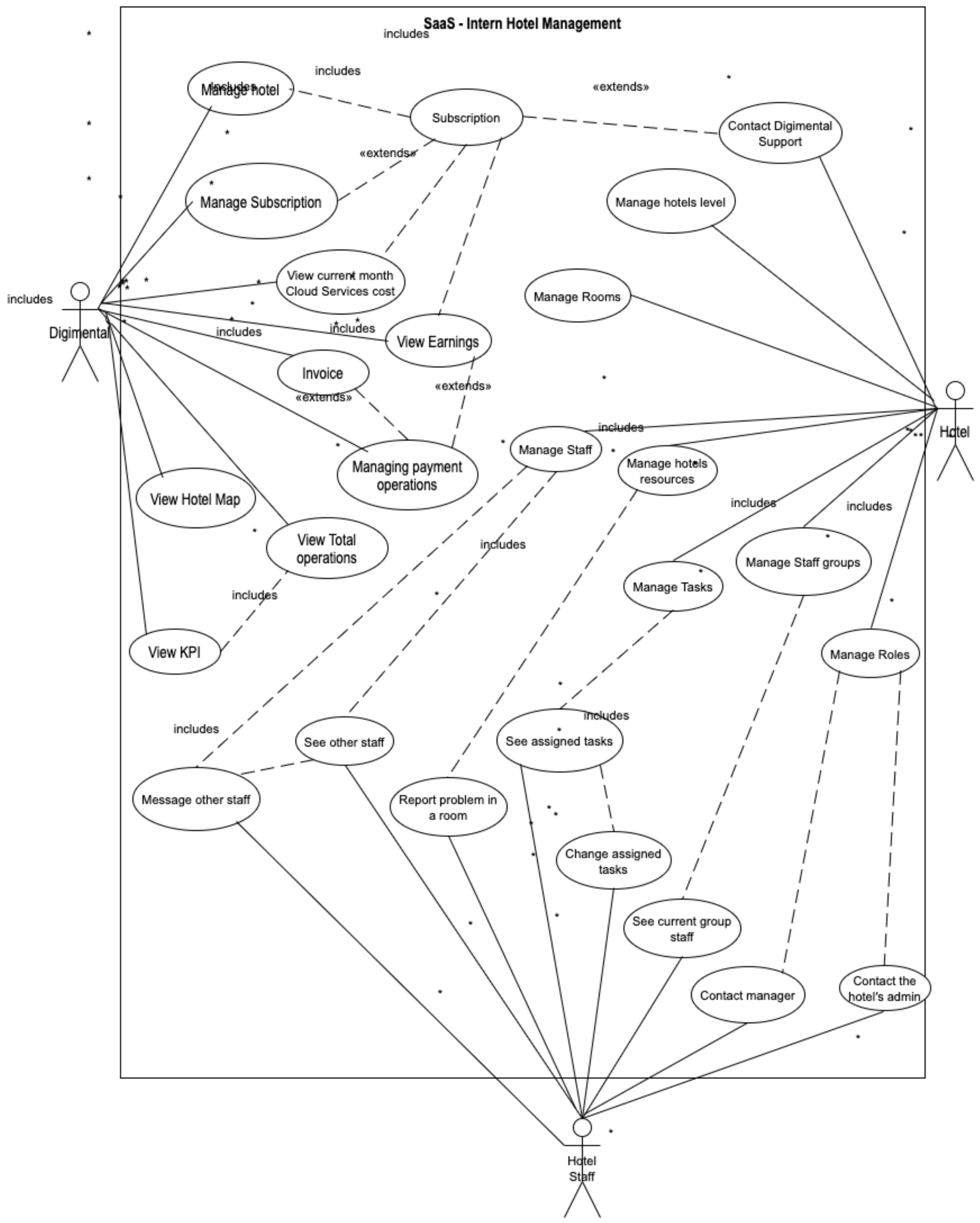
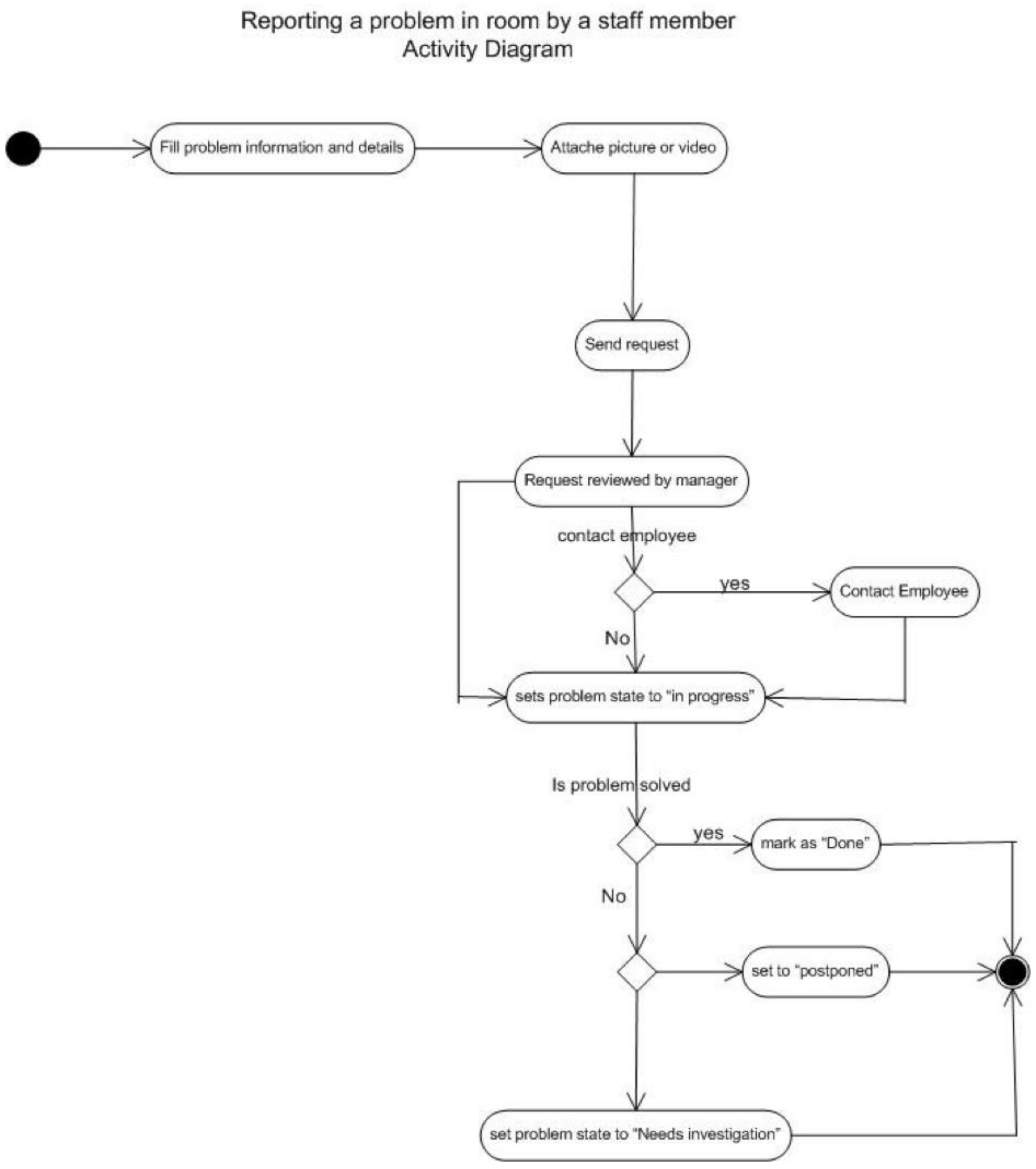


Figure 14: Use case Diagram

## 2.6.2 Activity Diagrams



Reporting a problem in room by a staff member
Activity Diagram
Fill problem information and details
Attache picture or video
Send request
Request reviewed by manager
contact employee
yes
Contact Employee
No
sets problem state to "in progress"
Is problem solved
yes
mark as "Done"
No
set to "postponed"
set problem state to "Needs investigation"

*Figure 15: Reporting a Problem in Room by a Staff Member*

Let us start with the activity diagram for reporting a problem (in the room, for example) by staff. The process is simple and straightforward. The user starts by filling information about the issue, e.g., room number, type, and description. Then, the user can attach a picture or a video of the issue.

After submitting the form, a manager reviews the request and may contact the employee for an investigation. The manager then will contact the right party and update the status of the issue to "in progress." If the issue is solved, then it will be closed. In the case of a complicated issue (need the mediation of higher authority), it may be labeled "for investigation."

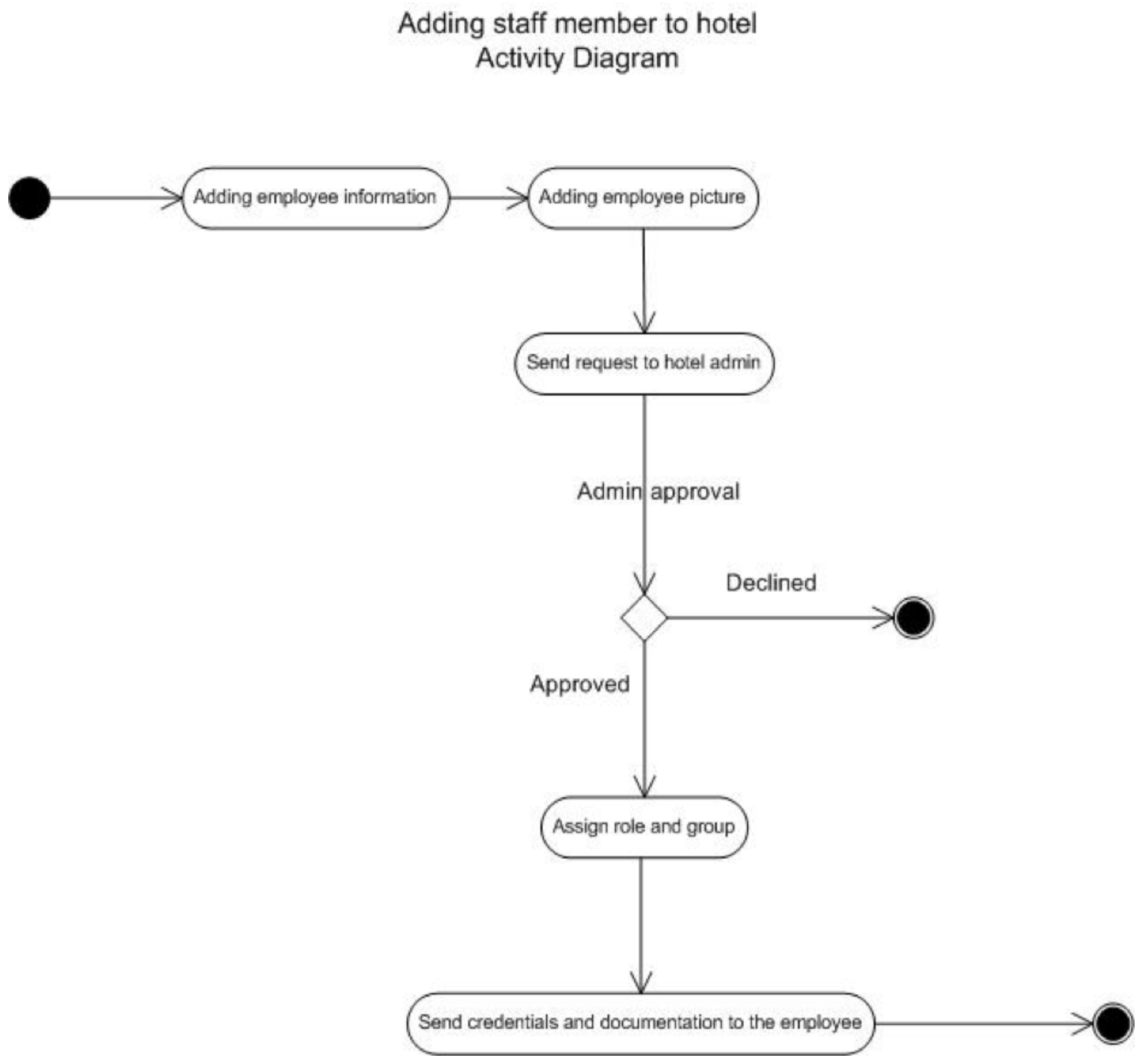

Adding staff member to hotel
Activity Diagram
Adding employee information
Adding employee picture
Send request to hotel admin
Admin approval
Declined
Approved
Assign role and group
Send credentials and documentation to the employee

*Figure 16 : Adding a Staff Member to a Hotel*

Adding a staff member activity diagram is direct and straightforward to the point. After submitting employee information, a hotel admin must review the application. Then, upon approval, the new employee gets assigned to a role and group.

Finally, the employee receives a text message or email with credentials and a simple guide of the platform.

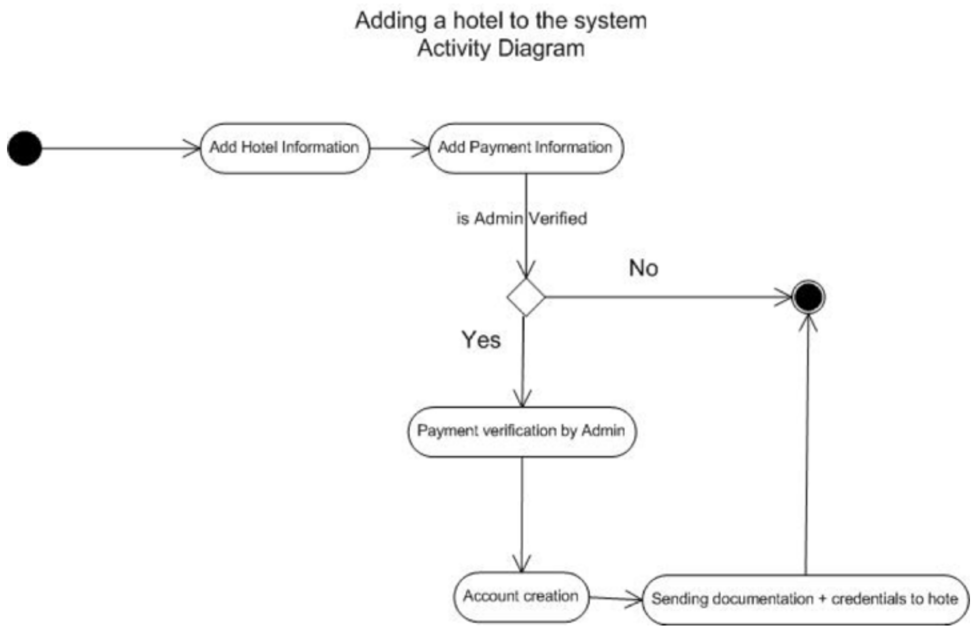

Adding a hotel to the system
Activity Diagram
Add Hotel Information
Add Payment Information
is Admin Verified
No
Yes
Payment verification by Admin
Account creation
Sending documentation + credentials to hote

*Figure 17: Adding a Hotel to the System*

After closing a deal with a hotel, a member of the sales team will add the hotel information and payment credentials or documents. A Digimental admin will review the application and verify the information provided, payments, selected plan, and act accordingly. After a successful review, the account gets created, and the hotel receives the credentials and a welcome guide to using the platform. If the application is rejected, the sales team receives an email to investigate the matter further and act accordingly.

### 2.6.3 Communication Diagrams

For brevity's sake, many details have been omitted. A communication diagram is also known as a collaboration diagram. On the communication diagrams, the objects are presented with association connectors between them.

Messages are added to associations, and arrows also show short ones pointing in the direction of the message flow. The sequence of messages is presented through a numbering system.

The diagram below lists all the interactions among objects dealing with the hotel admin.

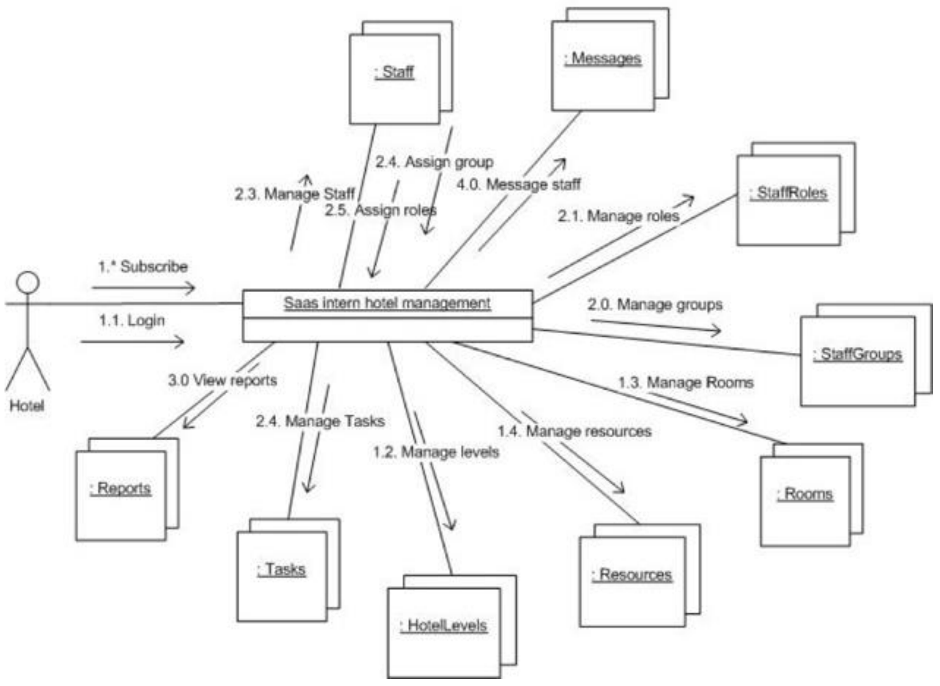


*Figure 18: Hotel Communication Diagram*

The figure above illustrates a simplified communication diagram for the hotel staff. It shows the four most important objects and the communication among them.

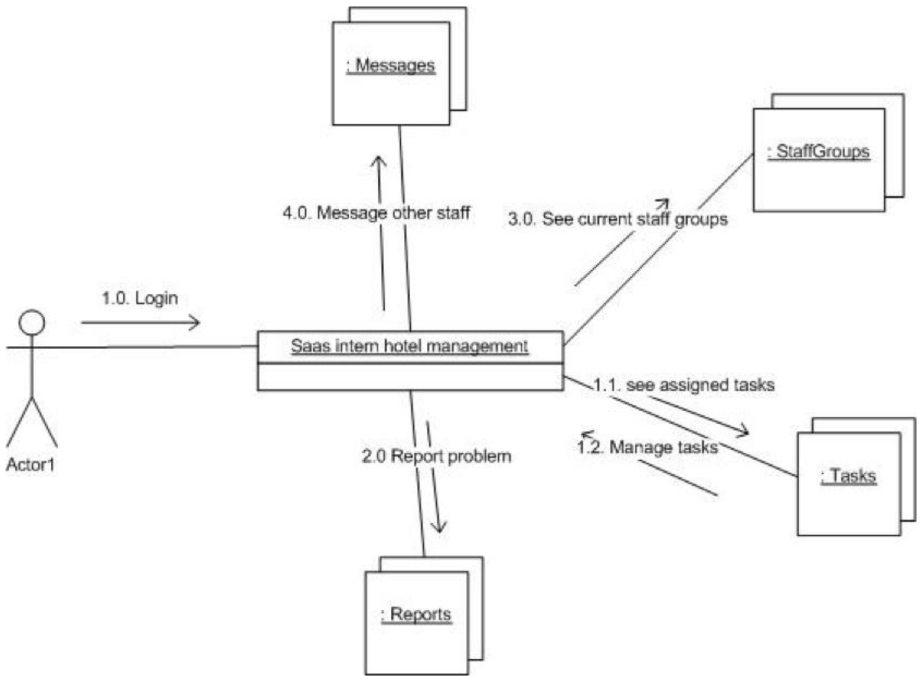


*Figure 19: Hotel Staff Communication Diagram*

### 2.6.4 State Diagrams

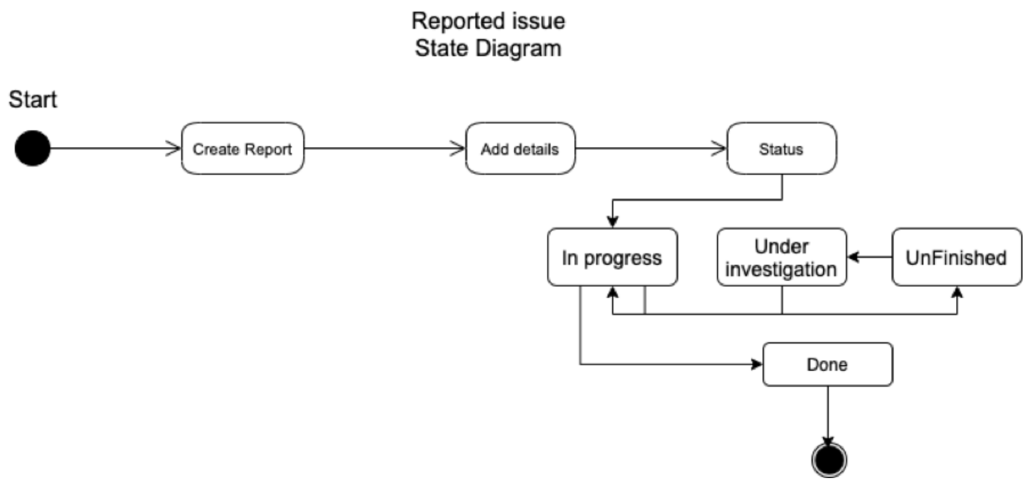


*Figure 20: State diagram of a reported issue*

A state diagram describes the dynamic behavior of the system in limited time instances. Hence, the state diagrams help understand the different states and transitions of the system. Let us start by the state diagram for a reported issue (covered in the activity diagram ).

The figure above illustrates the various states of an issue.

The next diagram is for the state of hotel resources. Each resource has its lifespan and can go through each state separately. The figure below illustrates the various states:

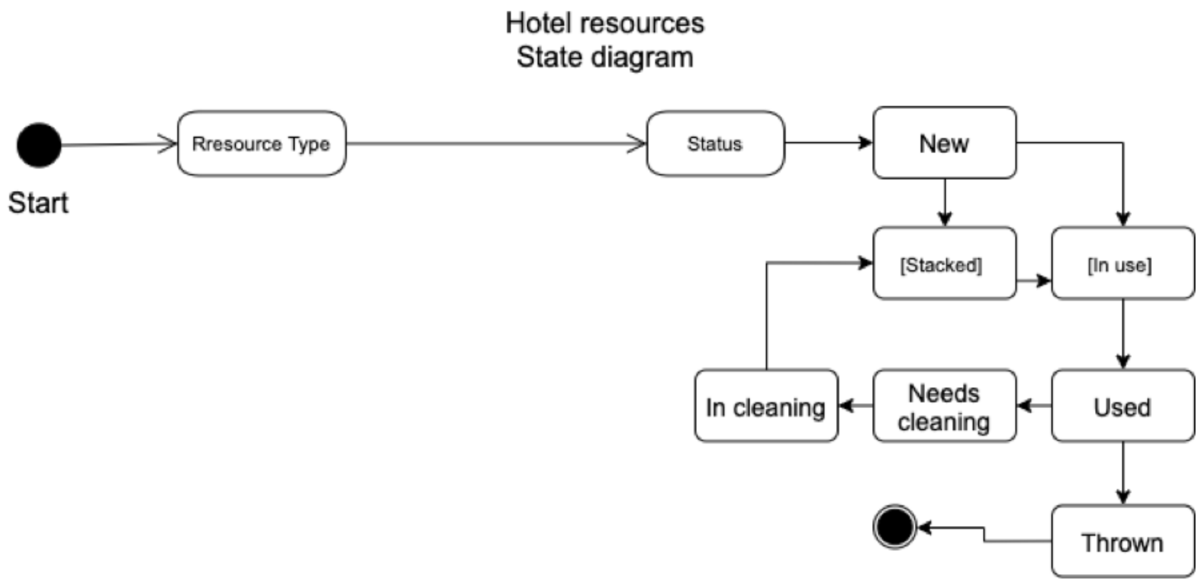


*Figure 21: State diagram for Hotel Resources*

## 2.7 Database Modeling

As mentioned previously, the project is based on Firestore and Real-time database, which are NoSQL databases. Defining the optimal data model in NoSQL is not an easygoing task because we need to anticipate how the data will be queried. There is no right or wrong modelization as long as we consider the tradeoffs and computing expenses of each model. However, anticipating the application UI will help leverage the power of data duplication and aggregation.

Unlike SQL databases, it is practically impossible to model a NoSQL database **visually** for the following reason:

- **Data duplication:** For data retrieving efficiency, we may have the same data in different collections.
- **Subcollections:** A document can have a nested collection inside.
- **Composite document IDs:** A document can have a composite document ID as a reference. For example, the Book-Reviews-Users model.

In a nutshell, the figure below illustrates some of the notable collections of the database (without modeling relationships, nested collection, aggregation, or indices).

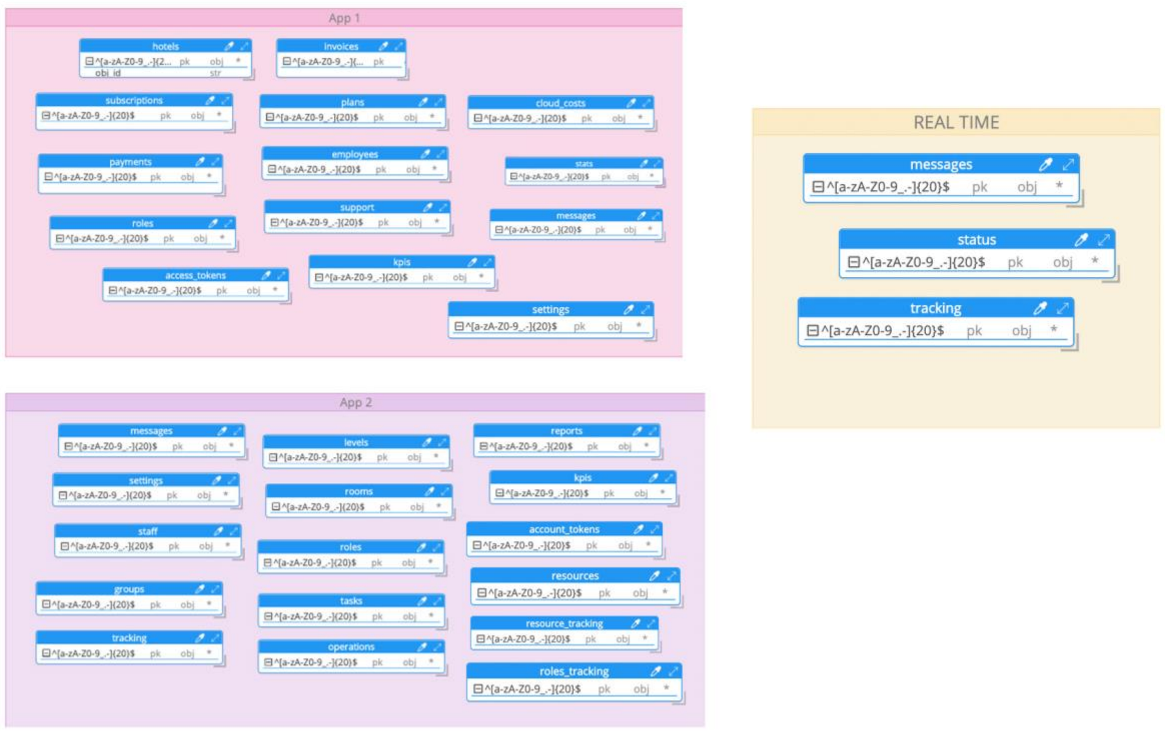


*Figure 22: Important Collections of the Database*

## 2.8 Application Design

As mentioned in the report's introduction, the Digimental designer will provide a high-fidelity design of the app components. Any prototyping software can be used to prototype the different components, e.g., Figma, inVision. It is an opportunity to collaborate with the designer and solve design-architecture problems.

The goal of the project is to bring these prototypes to life. The laborious work is creating reusable Angular components that can be easily used for front-end development. This step involved abstracting every UI component and well-examining the state object. Also, these Angular components must allow **data-binding** from, to, and within.

To achieve the desired result, we must take into consideration the following parameters:

1. **UI presentation:** how the component is represented in the tree.
2. **Business logic:** the business logic involved in the component.
3. **Injected services:** the interaction with services.

# Chapter Four: Implementation of the Solution

This chapter covers the development tools and implementation of the application. We will start by presenting the different tools, their benefits, and drawbacks. We will present the last section in terms of scenarios and use cases. *All the screenshots represent the current progress and not the final complete application*.

## 4.1 Details Of The Adopted Technologies

### 4.1.1 Javascript / TypeScript

*Figure 23: TypeScript Logo*

*Source :* *https://www.typescriptlang.org/branding*

Currently, JavaScript is the most popular programming language. It a scripting language that is used in creating interactive web apps and web pages. It follows the client-side programming principle; this means it runs on the user's web browser without the need for resources such as web servers. JavaScript can be integrated with other technologies such as XML, REST, and APIs among others. However, JavaScript is not designed for large and complex applications, but for simple web apps with fewer code lines.

TypeScript is considered as a superset of JavaScript and primary offers optional static typing, interface, and classes. Subsequently, developing a large project with JavaScript supplemented with TypeScript might be robust and is deployable where a native JavaScript application can run. Both scripting languages are open-source. This is because TypeScript adds optional types to the native JavaScript app, supporting the development of large and complex JavaScript-based applications on any platform, or for any browser. Also, TypeScript compiles into readable and standard-based JavaScript.

TypeScript can be installed and managed using the npm package. Additionally, TypeScript can be used for cloud functions, however, TypeScript renames the .js files to .ts. Hence, the beginning of JavaScript interoperability.

### 4.1.2 NodeJs

*Figure 24: NodeJS Logo*

*Source :* *https://nodejs.org/en/about/resources/*

NodeJS is a framework for developing server-side JavaScript apps. It is built on the V8 JavaScript runtime and non-blocking I/O, event-driven user model, making it perfect for data-intensive and real-time web apps.

Large companies offering SaaS services run Node in production, such as Uber, Netflix, Walmart, and PayPal. Node is many used in building backend services that communicate with client-side applications. These applications send and get data via backend services – APIs. APIs in this context will serve as interfaces between the various applications so they can integrate. For example, a web app will leverage the same API to send emails, initiate workflows, store data, or push notification in the cloud environment. Before Node, JavaScript was only run via a browser; however, Chrome’s V8 engine (fastest JavaScript engine available) was embedded inside a C++ program; hence, Node was conceived.

Node allows the manipulation of file systems by creating and deleting folders, creating web servers to serve the data, and query databases directly. Node is used extensively in the web app for server-side scripting.

### 4.1.3 Angular

*Figure 25: Angular Log*

*Source : https://angular.io/presskit*

Angular is a JavaScript-based library aimed at creating interactive user interfaces, and it extremely used by big companies’ websites such as Google and Alibaba for their frontend development. It is fast, modular, responsive, and can create **reusable components** and libraries. Additionally, it is flexible and scalable. In the web development community, it is considered as the best for data-driven analytics and user animations. In the **model-view-controller (MVC)** model, Angular concentrates just on the view aspect.

Angular shifts from the traditional frontend development methodology, where multiple lines of code were written in a separate HTML file and then grouped to present a particular component.

Angular development uses TypeScript as its primary language, and the **Angular-CLI**[6] empowers it. Combining Angular-CLI and **Narwhal extension**[7] is vital to develop a world-class, highly-structured application as Google does. Nx eases scaling the development from one team building one application to many teams, building multiple frontends and backend applications in the same workspace.
The advantage to the developer and development model it there is no space for conflict or lousy architecture. Angular follows a component-based approach. Introduction of Angular sort to provide a solution to developers as it supports many templating frameworks, which brings flexibility. This efficiency reduces coding time and enhances code quality more so in constructing machine-readable codes.

### 4.1.4 Google Cloud Platform

#### 4.1.4.1 Overview

*Figure 26: Google Cloud Logo*

*Source: https://cloud.google.com/press*

The Google Cloud Platform (GCP) is a suite of Google's computing resources made available to consumers through services as a public cloud package. GCP resources include physical hardware infrastructure – hard drives, processors, networking, and servers clustered within Google's globally distributed data centers. Subsequently, each component is custom built using the same patterns as those present in the **Open Compute Project (OCP)**.

The hardware resources in the Google server clusters are made available through virtualization. This is the most prolific approach to customers building and maintaining their physical infrastructure. Since it is a public cloud, the software and hardware resources are offered as integrated services providing access to the underlying resources.

Currently, GCP offers over 50 services that include software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS). Offerings in categories such as storage, database, compute, machine learning, developer tools, big data, identity, and security.

[6] https://cli.angular.io/

[7] https://nx.dev/

The benefit of using the GCP ecosystem is these components can be used independently, or as a combination of one or more, this depends on the developer's preference; therefore, GCP offers the ability to construct custom cloud-based infrastructure.

In terms of performance, GCP is hosted with the same underlying infrastructure Google uses for their end-user products such as YouTube and Google Search.

#### 4.1.4.2 Firebase

*Figure 27: Firebase Logo*

*Source : https://firebase.google.com/brand-guidelines*

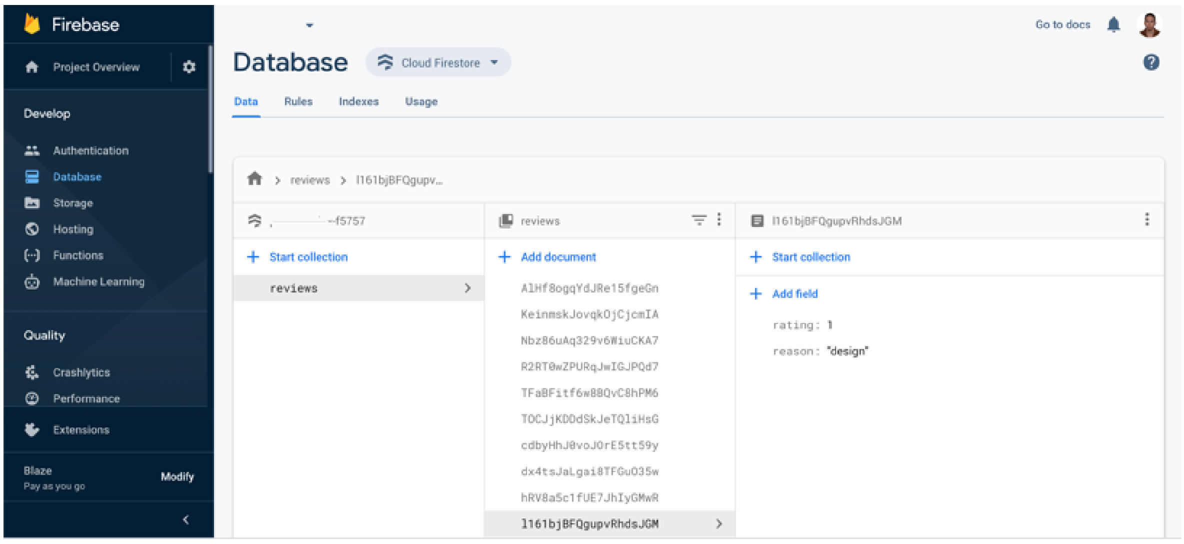
Today, there is almost an app for everything. For developers that tackle users' urgent needs through a mobile or web app, Firebase is the most popular platform.

Firebase is Google's mobile app development module that helps build, grow, and improve web apps. Firebase is a toolkit for building, growing, and improving web apps. Additionally, the tools offer developers an extensive coverage of services that they would typically build themselves. This includes tools such as databases, push messaging, file storage, configuration, authentication, and analytics among others. These services are all hosted in a cloud platform, and scale depending on the application's needs.

Furthermore, these tools are fully hosted and managed by Google – this means the client SDKs by Firebase interact with the backend services directly. Therefore, no need for middleware between the end-user applications and the services.
For example, if a developer is using the database option of Firebase, then when they write a query to the database, it directly interacts with the client application. This is a different approach from the traditional development model, that typically involved developing both backend and frontend applications.

Subsequently, the frontend would initiate an API to expose the backed, which in turn interacts with the database and other backend services.

#### 4.1.4.3 Firestore (NoSQL Database)



```
reviews

A1Hf8ogqYdJRe15fgeGn
KeinmskJovqkOjCjcmIA
Nbz86uAq329v6WiuCKA7
R2RT0wZPURqJwIGJPQd7
TFaBFitf6w8BQvC8hPM6
TOCJjKDDdSkJeTQliHsG
cdbyHhJ0voJOrE5tt59y
dx4tsJaLgai8TFGuO35w
hRV8a5c1fUE7JhIyGMwR
l161bjBFQgupvRhdsJGM

l161bjBFQgupvRhdsJGM
rating: 1
reason: "design"
```

*Figure 28: Firestore Interface*

Firebase uses real-time databases and cloud Firestore to offer database services. Furthermore, Firestore is a NoSQL database variant, unlike SQLite, which is a relational database mainly used in embedded devices, Redis – an in-memory database great for large and fast caches, Neo4j – graph database best for graphical analysis and expressing relationships between entities. This means all databases are specialized, and there is no one database right for all tasks, and it depends on the application needs and developer preference.

Firestore by Google's Firebase platform is a scalable, flexible database engine suitable for server, mobile, and web development. It is ideal for developing in a cloud environment. Similar to the Firebase real-time database, Firestore offers data synchronization across client applications via a real-time listener and offers support for offline mobile and web application that promoting responsive development regardless or network latency and connectivity.

Other key features include flexibility – Firestore supports hierarchical data structures, and stores the data in documents organized into collections, expressive querying – queries can be used in data retrieval, real-time updates – regular data synchronization to update everything, offline support – it caches data that the web application is actively using, therefore, writing, reading, listening, and querying data in the device in offline mode, and lastly designed to scale – bringing the best of the Google ecosystem, it offers automatic multiregional data replication, real-time transaction support, atomic batch operations, and strong consistency guarantees.

#### 4.1.4.4 Firebase RealTime Database

Firebase Realtime Database

*Figure 29: Firebase Realtime Database Logo*

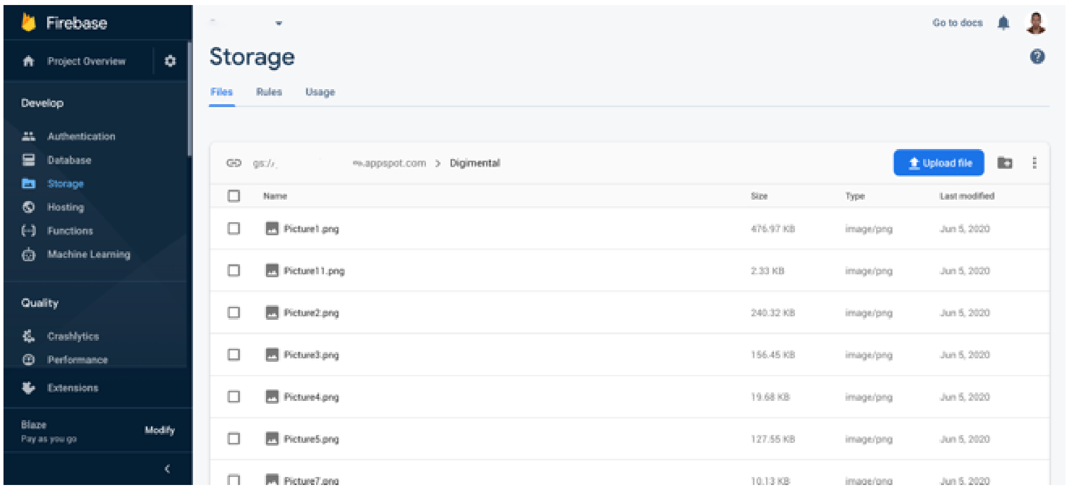
Source: https://firebase.google.com/brand-guidelines

Firestore stores and synchronizes data with the NoSQL cloud-based database. This means the data is synchronized across multiple customers in real-time and remains available even if the web app is offline.

Firebase's Realtime Database is hosted in the cloud as its data stores in JSON format. It regularly synchronizes with customers in real-time across various platforms such as Android, JavaScript, and iOS SDKs. As such, the Firebase Realtime Database lets developers build collaborating and rich web and mobile applications allowing secure access to the database engine from a user's side.

The data is available locally and offline; real-time events continue to give end-users a better responsive experience. Subsequently, when the device regains connectivity, the Realtime Database synchronizes with the local database and updates the offline application's dataset. This process merges conflicts automatically, offering flexibility and expression-based rules language – Firebase Realtime Database Security Rules. This is crucial when a developer is interacting with the Firebase Authentication. Since Firebase Realtime Database is a NoSQL variant, it has different functionality and optimizations compared to the traditional relational database engines.

#### 4.1.4.5 Firebase Storage



gs://. ...appspot.com > Digimental

| Name | Size | Type | Last modified |
|---|---|---|---|
| Picture1.png | 476.97 KB | image/png | Jun 5, 2020 |
| Picture11.png | 2.33 KB | image/png | Jun 5, 2020 |
| Picture2.png | 240.32 KB | image/png | Jun 5, 2020 |
| Picture3.png | 156.45 KB | image/png | Jun 5, 2020 |
| Picture4.png | 19.68 KB | image/png | Jun 5, 2020 |
| Picture5.png | 127.55 KB | image/png | Jun 5, 2020 |
| Picture7.png | 10.13 KB | image/png | Jun 5, 2020 |

*Figure 30: Firebase Storage Interface*

Cloud-based storage is built for developers who require storage and serve as user-generated content such as videos or photos. Cloud storage in Firebase is robust, cost-effective, and simple

object storage service primarily built for Google's upscaling. Through Firebase SDKs, Google's security is added into the file downloads and uploads for the Firebase based applications regardless of the network latency.
SDKs can be used to store video, audio, or other user-generated content. To access the Firebase Storage, developers must use Firebase SDKs for the cloud file download and upload from the user-end. If there is a poor network connection, the end-user is allowed to try again where they left off, saving end-user bandwidth and time. The files are stored in the Google Cloud Storage bucket, making them available via both Google Cloud and Firebase.
Moreover, Firebase SDKs for the cloud storage interaction seamlessly with Firebase authentication, which easily identify end-user and offer a declarative security language.

#### 4.1.4.6 Firebase Cloud Functions

Cloud Functions
for Firebase

*Figure 31: Cloud Functions Logo*

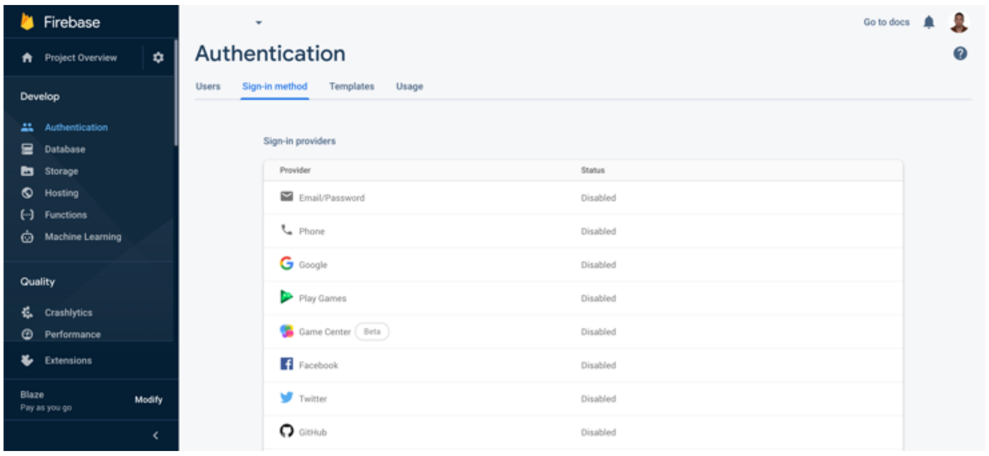
Source: *https://firebase.google.com/brand-guidelines*

Today developers can make web APIs or web apps, write the source code, and deploy the application to Cloud Functions. It is fast and scales without the responsibility of managing the backend. Traditionally, developers found it difficult to manage an application's backend, and it does not scale in almost all instances.
Additionally, Cloud Functions hosted in Google Cloud are available from Firebase. Firebase Cloud Functions are a serverless framework that allows developers to automatically execute backend source code responding to events triggers by HTTPS requests and Firebase features.
The JavaScript source code stored in Google Cloud can run in a managed environment. Therefore, no need for managing and scaling the servers. It works by a developer writing a function or complete source code; Google servers start function management immediately. Also, the function can be executed by directly initiating an HTTP request. In some cases, they are run in the background; hence, Google servers listen for events and execute the functions directly if triggered.

#### 4.1.4.7 Firebase Authentication

In this context, the web app will need to identify the end-users. Identifying the user identity allows the web application to securely save the user data in the cloud platform and offer the same level of personalized experience to all end-users in the cloud platform.
Firebase Authentication offers backend services, and easy to use SDKs for authenticating web and mobile applications. Firebase Authentication supports authentication through passwords, popular featured identity providers such as Google, Twitter, Facebook, and phone numbers.

Additionally, Firebase Authentication incorporates closely with other Firebase services, leveraging industry standards such as **OpenID** Connect and **OAuth 2.0**, this means it can be easily integrated through backend customization. To authenticate a user, a web app must get the user credentials from the end-user; these credentials can be username or email, password, **OTP** code from a dedicated identity provider. Then, they are passed to the Firebase Authentication SDK where the backend services response to the end-user events.



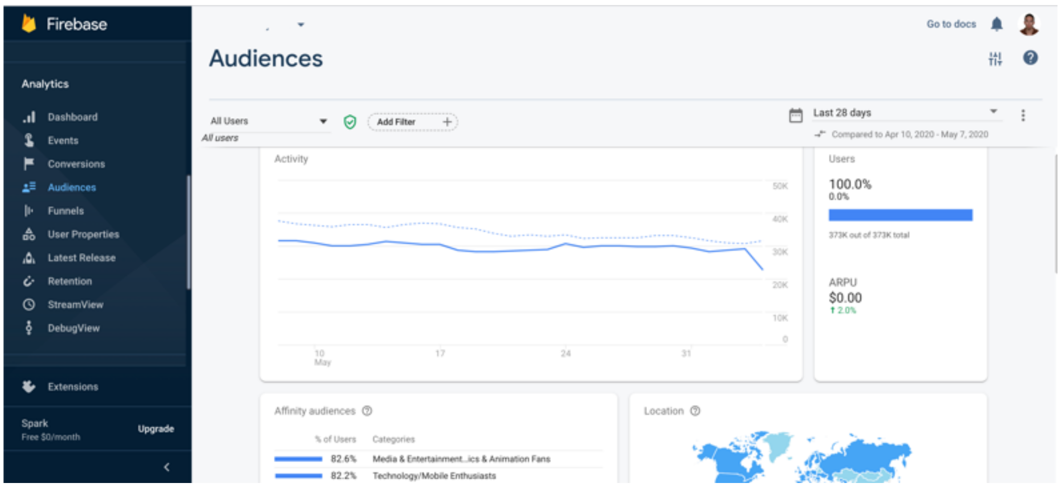
Sign-in providers

| Provider | Status |
|---|---|
| Email/Password | Disabled |
| Phone | Disabled |
| Google | Disabled |
| Play Games | Disabled |
| Game Center Beta | Disabled |
| Facebook | Disabled |
| Twitter | Disabled |
| GitHub | Disabled |

Figure 32: Firebase Authentication Setup Interface

## 4.1.4.8 Firebase Analytics

Firebase offers Google Analytics, a free but unlimited analytic tool. Google Analytics allows developers to integrate across the entire Firebase ecosystem, offering unlimited reporting to over 500 customers events that can be integrating via the Firebase SDK.



| % of Users | Categories |
|---|---|
| 82.6% | Media & Entertainment...ics & Animation Fans |
| 82.2% | Technology/Mobile Enthusiasts |

Figure 33: Firebase Analytics Interface

Analytics reports help businesses and service providers understand how customers behave. This will allow the service provider to identify and make informed decisions concerning the web

app service marketing and sales. Besides unlimited reporting, Google analytics offers audience segmentation.
For startups and new developers, Google Analytics offers a way of understanding end-user behavior when using the web app from various platforms such as Android, web, or iOS. The SDKs automatically capture the number of events and user behavior to interpret and present custom events that measure the things that uniquely identify the business.

Subsequently, Analytics has integrated with several Firebase features; for example, it can automatically log events corresponding to notification messages via the notification composer, therefore, offering reports on the impact of the campaign.

#### 4.1.4.9 Cloud Run (serverless)

Cloud Run is a Google Cloud is a fully manageable compute environment that automatically scales a subscriber's stateless containers. Hence, Cloud Run is serverless. This means abstracting all infrastructure management, so the developer can focus on other matters such as building high applications; this feature reduces development and deployment time. Additionally, container execution is fully managed by Cloud Run, supporting both on-premises and Google Cloud ecosystems. Though owned by Google, Cloud Run is built on the open standard – Knative, enhancing web app portability.

The key feature of Cloud Run include development with any language, any library, any binary – this means a developer can use any programming language on any operating system library, leverage containers workflow and standards, and offer pay per usage – this means the subscriber pays only for when the code is running, billed to the nearest 100 milliseconds.

Traditionally, running web apps was challenging, with all the overheads of managing and deploying virtual machines, services, pods, clusters, and more.

#### 4.1.4.10 Cloud Build

Google Cloud Build allows developers to create consistent, reliable, and fast builds across the Google Cloud ecosystem regardless of the development language. The benefits of using Google build it automatically create containers and non-containers artifacts that can be committed to a version control system like Git or Bitbucket.

Deployment tools offer the ability to roll back to the previous release, through a series of release management tools. Also, it employs that dependency isolation tool during the execution, ensuring that no implicit dependencies leak from the immediate ecosystem. A benefit of using this approach is the simplification of the setup for new developers to the project.

The build stage is a transformation that converts code repo into an executable bundle – build. Version control, in this instance, commit specified deployment processed, as such, fetching vendor dependencies and compiles assets and binaries. Additionally, it lets developers control the definition of custom workflows for deploying, building, and testing across various platforms such as Firebase, Kubernetes, serverless, or virtual machines.

Other features include zero configuring Docker builds – running builds, and testing changes pushed to Git repository in one phase, scalable – allows running single builds on a local machine to running multiple builds in parallel, allows extremely fast builds – access devices are connected through Google's global network reducing the build time, additionally, allowing run builds on high capacity virtual machines caching the source code, media files and other dependencies enhancing build speeds. Unparalleled security – allows running builds on proprietary Google security infrastructure. In this mode, it identifies package vulnerabilities in containers.

#### 4.1.4.11 Cloud Pub/Sub (Message broker)

Cloud Pub/Sub is an asynchronous messaging service that categorizes services that initiate events from services that process events. Pub/Sub uses message-oriented event ingestion or middleware delivery of real-time analytics pipelines.

Pub/Sub are the GCP counterparts. Subsequently, as mentioned earlier, Cloud Functions can be triggered in significant ways: through direct HTTP requests and background events. Paring Google Cloud's platform Cloud Pub/Sub and Cloud Functions is one of the most common integrations. For example, Cloud Functions is a direct subscriber to a specific Cloud Pub/Sub backing service.

Other common uses include balancing workloads in network clusters, integrates asynchronous workflow, logging to multiple systems, and reliability improvement.

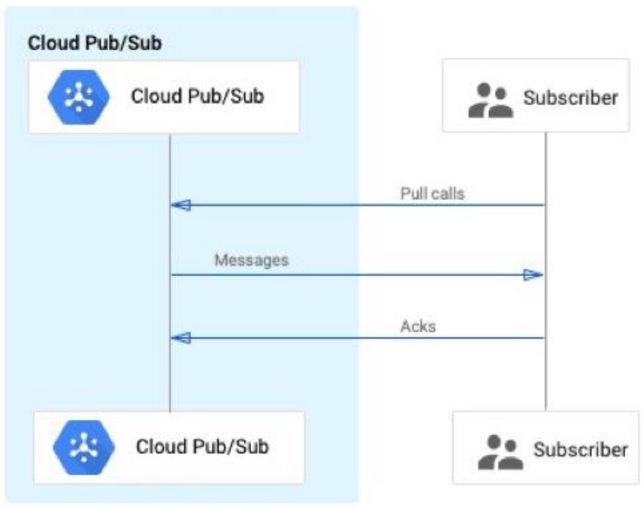


*Figure 34: GCP Pub/Sub*

#### 4.1.4.12 Cloud Log

Cloud Logs allows developers and system administrators to store, analyze, alert, search, and monitor log data for events from the Google Cloud Platform. The Google Cloud Log API allows developers to interpret custom log data from various sources. Subsequently, cloud logging is a fully managed service that runs at scale and integrates with cloud-based applications. Subsequently, there are no beginning or endings unless the web application is permanently pulled down, for one reason or another. Logs are typically generated in their raw form and stored in text files one event per line.

Additionally, they are continuous as long as the web app is operational. The twelve-factor app model does not concern itself with storage or routing of its output streams, it does not attempt to write or manage log files. However, each running process will create and write its event stream unbuffered. Moreover, in a local environment, the developer views the output streams in the foreground while observing the web app's behavior.

Logs are part of every system; they offer visibility of the behavior of the web application, server environment, and user movement. They are commonly written in files in the drive (log or log file). However, this is only an output format. Logs are streams of accumulated time-order events from output streams, backing services, network port forwarding, user movement logs. Subsequently, there are no beginning or endings unless the web application is permanently pulled down, for one reason or another.

#### 4.1.4.13 Cloud monitoring

Cloud Monitoring offers visibility into the uptime, overall health, and performance of the cloud-based platform. This means the cloud service collects events, metrics, and metadata from the Google Cloud Platform. Other services include application instrumentation, a variety of common applications, and uptime probes. This ensures the developer or system administrators understand how the Logs are streams of accumulated time-order events from output streams, backing services, network port forwarding, and user movement logs.

Cloud monitoring offers flexible dashboards and reports that help in identifying emergent issues. Any anomaly detected initiates an alert based on its priority and severity level.

#### 4.1.4.14 Cloud Scheduler

The SaaS platform will offer a fully managed and enterprise-level schedule. A cloud scheduler it a tool that allows a user, in this case, the developer, or end-user to schedule virtually any task or event, including big data jobs, batch, cloud environments and more.

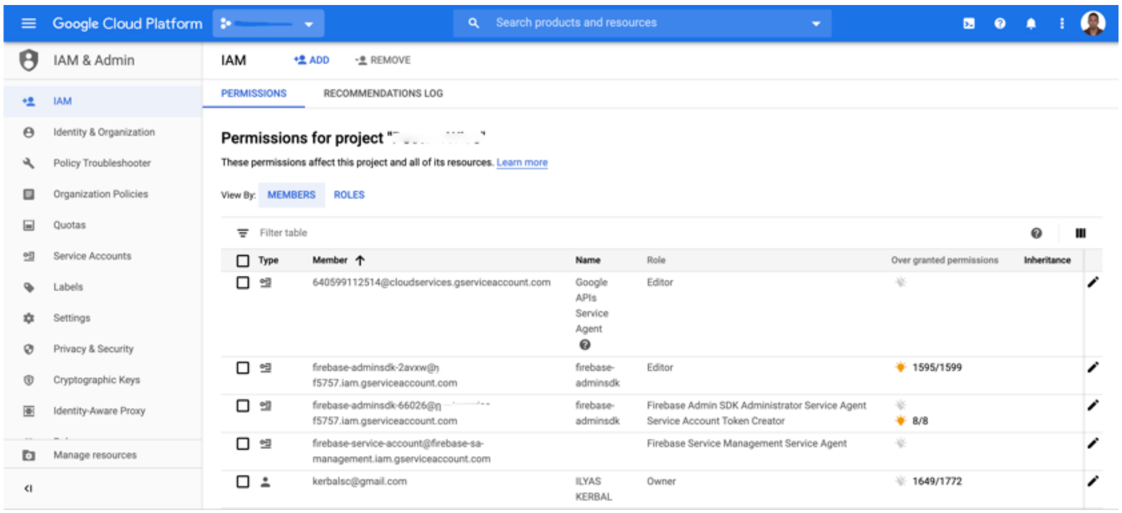
Additionally, it offers the ability to automate almost everything in the SaaS web app platform. The Cloud scheduler can act as a single unit for managing all automation tasks in one location under the platform. The main benefits of using a cloud scheduler are reducing human error, reducing automation time, and offers a faster technique for managing task automation.

### 4.1.4.15 Cloud Identity and Access Management (IAM)

The SaaS web app platform offers cloud identity and access management (IAM) that lets system administrators authorize and mage user privileges in the server administration access.

For organizations with complex business structure, numerous workgroups, or many projects, the cloud identity and access management (IAM) offers a unified but centralized security policy view across the SaaS platform. This tool ensures there is inbuilt auditing to ease compliance processes.

Additionally, Cloud IAM offers tools that efficiently manage user access controls with high automation and minimum fuss. This means it makes it easy for administrators to manage user and system permissions.



Permissions for project "

These permissions affect this project and all of its resources. Learn more

| Type | Member | Name | Role | Over granted permissions | Inheritance |
|---|---|---|---|---|---|
| | 640599112514@cloudservices.gserviceaccount.com | Google APIs Service Agent | Editor | | |
| | firebase-adminsdk-2avxw@ f5757.iam.gserviceaccount.com | firebase-adminsdk | Editor | 1595/1599 | |
| | firebase-adminsdk-66026@ f5757.iam.gserviceaccount.com | firebase-adminsdk | Firebase Admin SDK Administrator Service Agent<br>Service Account Token Creator | 8/8 | |
| | firebase-service-account@firebase-sa-management.iam.gserviceaccount.com | | Firebase Service Management Service Agent | | |
| | kerbalsc@gmail.com | ILYAS KERBAL | Owner | 1649/1772 | |

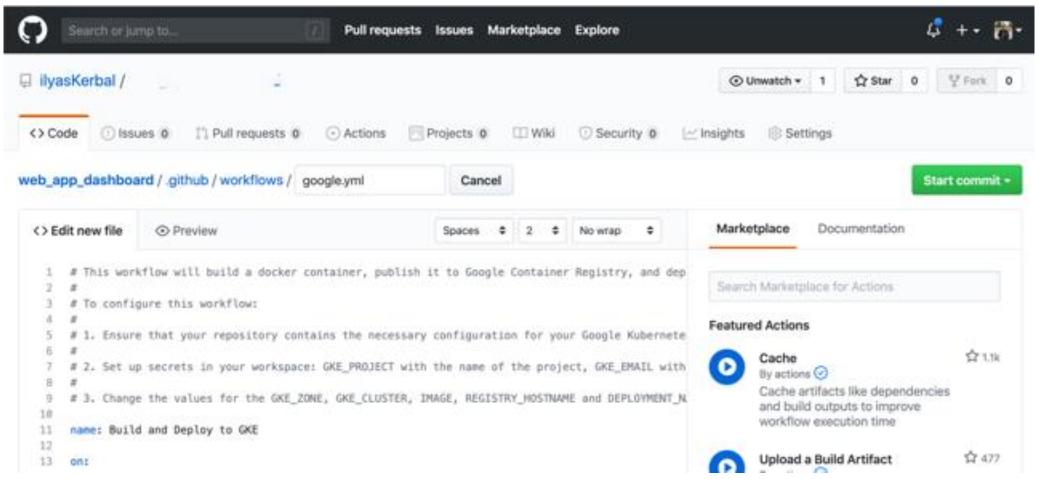
Figure 35: IAM Permissions Interface

### 4.1.4.16 Multi-factor authentication (MFA)

Information is now one of the most critical assets of any organization. Protecting information is crucial for every organization. The SaaS system offers a tool for securing user accounts and a wide variety of data through multifactor authentication techniques such as Google Authenticator, push notifications and OTP phishing resistant Titan Security keys.

Maintaining confidentiality and integrity is crucial to the SaaS web app service, MFA ensures that confidentiality is maintained by ensuring more layers of user authentication. This means if a malicious agent gets past the first layer, they have other layers before gaining access. Firebase allows for multifactor integration on its platform.

### 4.1.5 Git / Github / Github Actions



```
# This workflow will build a docker container, publish it to Google Container Registry, and dep
#
# To configure this workflow:
#
# 1. Ensure that your repository contains the necessary configuration for your Google Kubernete
#
# 2. Set up secrets in your workspace: GKE_PROJECT with the name of the project, GKE_EMAIL with
#
# 3. Change the values for the GKE_ZONE, GKE_CLUSTER, IMAGE, REGISTRY_HOSTNAME and DEPLOYMENT_N

name: Build and Deploy to GKE

on:
```

*Figure 36: Configuring Github Action with GCP*

Git or GitHub is the world's most popular version control system. It offers free and subscription packages to its user. Git can be viewed as the best example of Software-as-a-service. Git can be integrated with the Firebase platform, as discussed in the version control system section of this report. It used by the SaaS web app developer to track code modifications during the development and deployment stages.

Furthermore, the GitHub actions make it effortless for developers to automate the software workflows. Additionally, it can be integrated with the CI/CD that allows developers to build, deploy, and test web apps straight from GitHub. As such, a developer or team may be working on a new feature or may be fixing a bug while another may be adding new configuration features that changing the base code. Each developer or team will be changing several parts of the file tree. Version control assists in solving these kinds of challenges – tracking every change by a contributor, thus, helping prevent concurrent works from conflicting.

Moreover, changes made to one part of the software may be incompatible to other contributor's changes working simultaneously. Therefore, good version control supports a contributor's preferred workflow without imposing a particular methodology. Ideally, it works on any platform, rather than dictating which platform or contributors must use. The twelve-factor application will always be tracked by a version control software such as Git.

### 4.1.6 Npm (packages manager / Task runner)

Npm is a package manager for JavaScript. It is the leading software repository. Npm host popular packages such as jQuery, NodeJS, Angular, React, and Bootstrap among others. Additionally, it links to version control tools such as Git, therefore, allowing developers to create and share projects. The online npm repository is large and diverse, as such, NodeJS backend and JavaScript frontend developers make use of it as the packages can be used in any environment.

## 4.1.7 Docker (Containerization)

Figure 37: Docker Logo

Source: https://www.docker.com/company/newsroom/media-resources

Docker is one of the tools used for the idea of segregation of resources, creating tools that allow applications to repackage with all their dependencies bundled together and run whenever initiated. Docker views a container as a standard unit of software that categorized source code and its dependencies so a web application can execute faster and ensure reliability when migrating from one deployment platform to another. Similar GCP, Docker containerization uses the same system resources and infrastructure, there is no separation, or dedicated hardware-level components that reveal the fact that they operate as an independent system.

## 4.1.8 Webpack

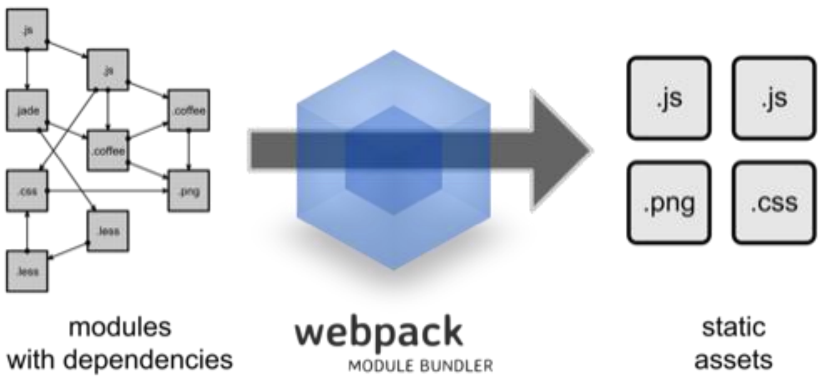
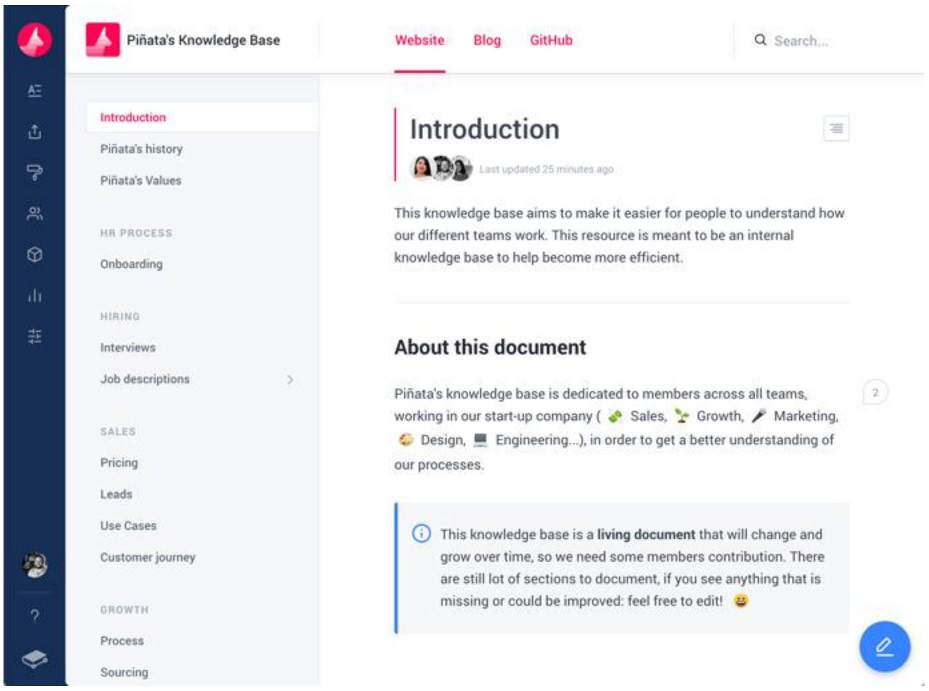


Figure 38: Webpack Overview

Webpack is a software solution that is used by developers to transform the development files (modified files) to production files (files ready to deploy). Its importance is dependencies management – it makes it easy to track the dependencies between the Angular and JavaScript files in the Firebase environment.
Additionally, it can be viewed as a webpack for modules, whose main goal is bundling the JavaScript files ready for deployment. For its installation, it uses the npm packer manager.

## 4.1.9 Gitbook / Slack / JsDoc / (Collaboration)

**GitBook** is considered a markdown editor. In this context, it is used to create rich and detailed documentation that can be exported to HTML, PDF, or ePub. It is the primary tool used for documentation throughout the SaaS web app development and deployment, as well as the project's contributors and the installation steps.



Introduction

Last updated 25 minutes ago

This knowledge base aims to make it easier for people to understand how our different teams work. This resource is meant to be an internal knowledge base to help become more efficient.

About this document

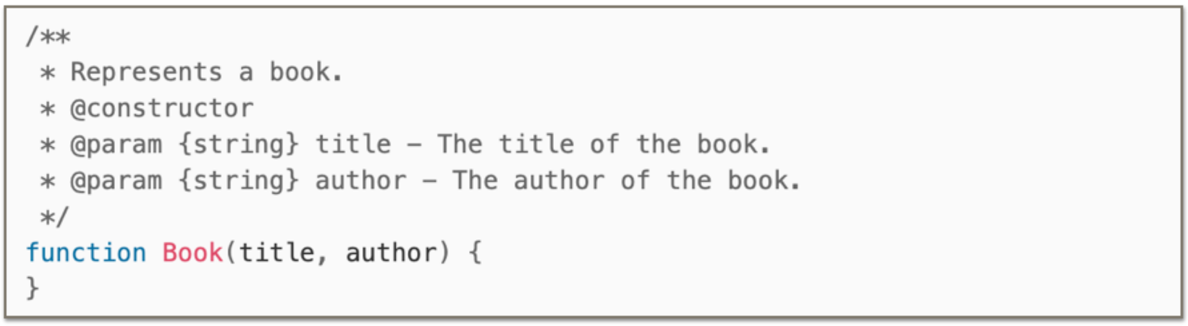
Piñata's knowledge base is dedicated to members across all teams, working in our start-up company ( Sales, Growth, Marketing, Design, Engineering...), in order to get a better understanding of our processes.

This knowledge base is a **living document** that will change and grow over time, so we need some members contribution. There are still lot of sections to document, if you see anything that is missing or could be improved: feel free to edit! 😀

*Figure 39: Using Gitbook as a Knowledge Bases*

**Slack** is a collaboration tool for teams working in knowledge-based companies. While Slack is similar to GitBook, however, this tool keeps is third-party and offers a platform for collaborating and managing the various project stakeholders. It involves access by remote teams as it is an online platform.

**JsDoc** is a documentation generator for JavaScript. It is critical to automate the process of generating documentation. The developers must add the documentation comments directly to the source code, right alongside the code itself. Furthermore, developers can integrate JsDoc and Gitbook.



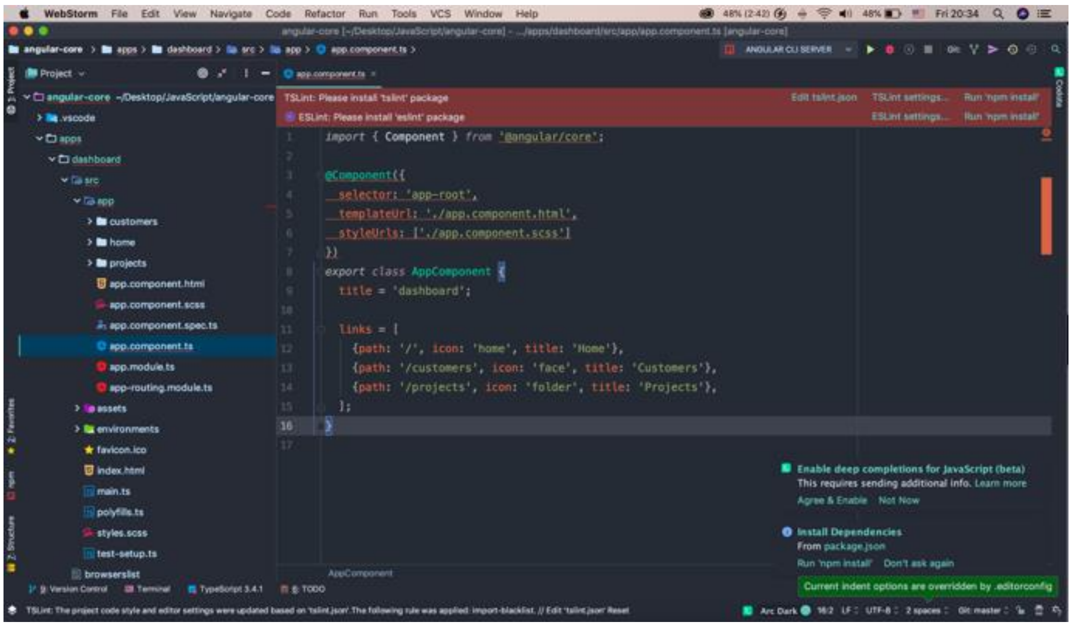
```
/**
 * Represents a book.
 * @constructor
 * @param {string} title - The title of the book.
 * @param {string} author - The author of the book.
 */
function Book(title, author) {
}
```

*Figure 40: Example of documenting a function using JsDoc*

### 4.1.10 SaSS / CSS

SaSS and CSS are used in the styling of the SaaS web application. Based on the technologies being used and the current development trends, CSS3 and SaSS are used commonly. Additionally, the SaaS web application will be using SaSS to complement the CSS3 used. SaSS can be viewed as an advanced CSS module, additionally, it is compatible with CSS and CSS3, feature-rich than CSS3, mature, and offers an extensive library of frameworks that can be built into any platform.
SaSS helps modulize app styling and offers powerful tools e.g., lightning or darken a color, math operations, functions, and mixins. SaSS can be easily integrated with WebPack for smooth compilation and bundling.

#### 4.1.4.23 IDE: Webstorm



TSLint: Please install 'tslint' package

ESLint: Please install 'eslint' package

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'dashboard';

  links = [
    {path: '/', icon: 'home', title: 'Home'},
    {path: '/customers', icon: 'face', title: 'Customers'},
    {path: '/projects', icon: 'folder', title: 'Projects'},
  ];
}
```

Enable deep completions for JavaScript (beta)
This requires sending additional info. Learn more

Install Dependencies
From package.json

Current indent options are overridden by .editorconfig

TSLint: The project code style and editor settings were updated based on 'tslint.json'. The following rule was applied: import-blacklist. // Edit 'tslint.json' Reset

*Figure 41: Webstorm IDE Interface with the Project Open*

WebStorm is an IDE from JetBrains. It is platform-independent, this means it can be installed on a Unix, Windows, or Mac operating environment. Some of the features that make it preferable for web development and deployment is that it is built explicitly for JavaScript development, testing, version control, and deployment. It comes as either a free version and a one-time license purchase. Moreover, it supports both frontend and backend development.

However, the choice of an IDE is mainly based on the developer's preference and features. For example, WebStorm offers multiple plugins out of the box. This is the most crucial feature that makes WebStorm preferable to other IDEs such as Atom or Brackets. This means a developer can write source code, test it, execute it, debug it while the web app is still running directly from WebStorm. Some developers might view WebStorm as an advanced tool, however, in a complex and commercial environment, WebStorm offers the best features, functionality, security, compatibility, and integration required to deliver a high-quality software solution.

### 4.1.12 Sending Emails

*Figure 42: SendGrid Logo*

*Source: https://sendgrid.com/resource/brand/*

Primary the SaaS web App uses the SendGrid email service. It is an excellent service as it offers its API free to developers. Additionally, it is a tremendous emailing service that offers an easy to use platform where service providers can integrate with their web apps and easily manage promotional as well as confidential emails. Since the SaaS web app deployment environment is Google Cloud, more so Firebase, SendGrid it the best open-source solution as it can be used in NodeJS running on a virtual machine instance. Additionally, it a cloud-based solution that can deliver emails to a large customer base.

Moreover, SendGrid can serve anyone from open-source web developers to enterprises. Ideally, it is fast, easy to use, can be configured to use any domain, and it the most reliable open-source emailing service. Since the project uses Git for its version control, SendGrid has provided a GitHub repo which any developer can access and use in their development. The only drawback of using open-source software is, the source code cannot be repackaged by a different vendor and software as proprietary software. However, SendGrid does not have these limitations as the developers encourage its adoption.

### 4.1.13 Sending SMS

*Figure 43: Twilio Logo*

*Source: https://www.twilio.com/company/brand*

Twilio is a public service company that offers an open-source SMS and Voice service. The Company allows developers to receive and make calls programmatically. The Twilio API offers various services. As mentioned at the beginning of this report, the most challenging stage of the project is external communication – email, SMS, and more. For bulk SMS and keeping track of the contacts, Twilio Flex offers a simple mark-up language and API for developers and businesses to quickly scale, that is reliable, fast, and with advanced SMS and voice management.

Twilio offers a cloud-based telephony infrastructure enabling developers to integrate its real-time phone calls and SMS services to web apps. For commercial use, Twilio offers a pay-as-you-grow package that offers flexibility to the cloud-based web app service. For this integration, it only requires the SMS API, which can be implemented for shortcode technology, allowing faster and larger SMS messaging.

### 4.1.14 Trello

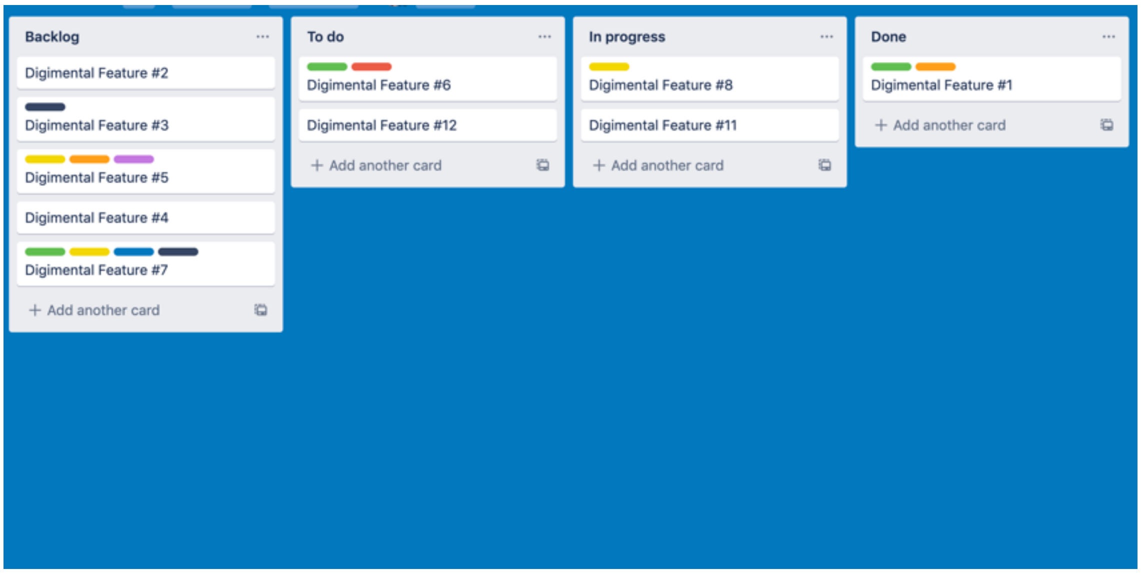


*Figure 44: Trello Kanban Board*

Trello is an integration tool for managing teams' collaboration in Kanban-style. Trello organizes the projects into boards where team members collaborate and notify each other about project updates. Additionally, it gets more done. As a project manager, it is important to use tools that ensure easy, flexible, and manageable collaborations.

**Kanban** is one of the popular project management tools that offers agile software management development. It is a framework that offers development teams real-time communications and enhanced capacity with full transparency of work. It presents its work items visually on the dashboard allow developers and other relevant stakeholders to view and manage their tasks.
Kanban is a popular visual board and cards setup that effectively displays the status of work. With team collaboration in mind, Kanban offers software developers an easy to view plan of a project. Ideally, its detailed but straightforward display allows software developers to view tasks that need to be addressed, work in progress, completed work, and the teams working on them. This visualization simplifies the project management workload and enhances development processes. Software developers can appreciate this approach as it incorporates the agile model.

### 4.1.15 Summary of Stack Choices

*Table 2: Summary of Stack Choices*

| Technology | Sumary |
|---|---|
| **Javascript** | Javascript is everywhere, from web development to desktop and IoT. Using Javascript has the following advantages:<br>- Using one stack for front-end (Angular) and backend (NodeJs, Firebase).<br>- JavaScript is expressive<br>- Access to the NPM package manager with more than 350,000 packages (Frameworks, tools, and more).<br>- Javascript is popular with a large developer community. |
| **TypeScript** | TypeScript is considered as a superset of JavaScript and primary offers optional static typing, interface, and classes. Some of the TypeScript benefits are:<br>- Makes Javascript code easy to debug, read, and share.<br>- Open-source.<br>- Helps write a structural code rather than nominal.<br>- Extensive support from frameworks, IDEs, and tools.<br>- Offers features available in newer versions of Javascript. |
| **NodeJs** | Gains of using NodeJs include:<br>- Open-source based on Chrome's V8 engine (fastest JavaScript engine available).<br>- Asynchronous request handling and Non-blocking code.<br>- Large and active community.<br>- Integrable with Google Cloud Run. |
| **Angular** | Angular is a front-end framework for creating flexible, scalable, and well-structured web apps. Using Angular has the following advantages:<br>- Open-source backed by a vast community and big companies, e.g., Google, Alibaba, Tencent.<br>- Model-view-controller (MVC) architecture.<br>- Quality component-based architecture using Narwhal extension and Angular-CLI.<br>- Unit-test friendly and Powerful ecosystem. |

| Technology | Sumary |
|---|---|
| **Google Cloud Platform** | Even if Google was one of the late entrants to the cloud services market (third in market share), Google is overdelivering in terms of data centers and fast response time. Here are some pros of adopting GCP:<br>- Flexible plans, pricing, and extensive discounts.<br>- Fully integrated ecosystem offering all the tools need for high-scalability applications.<br>- Adopted by big companies, e.g., Paypal, Bloomberg, eBay.<br>- Wide range of tools and CLI utilities.<br>- Support and commitment to open-source. |
| **Firebase** | Firebase is an extension of the GCP acquired by Google in 2014. The platform offers a complete backend solution. Here are some pros of using Firebase:<br>- Integration with GCP - every Firebase project is a GCP project.<br>- Free to get started with flexible pricing.<br>- Quickly prototyping and testing ideas without APIs and backend development.<br>- Easy and straightforward integration with several frameworks and languages. |

## 4.2 Development and Implementation

### 4.2.1 Digimental Dashboard Login

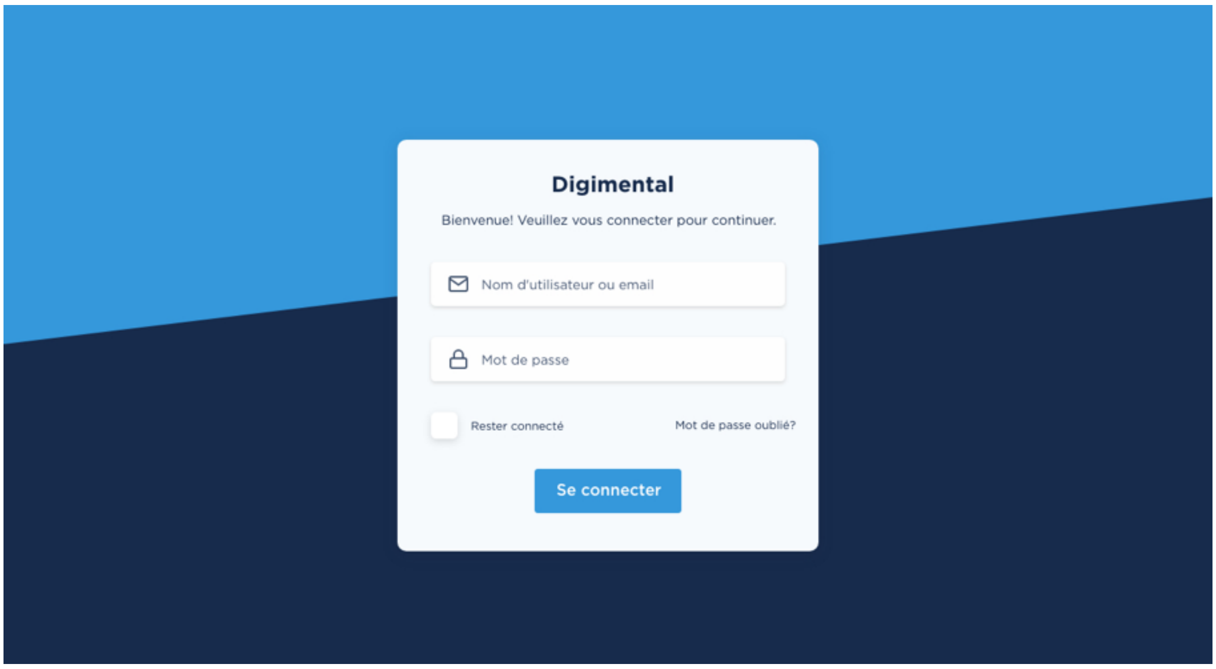

Digimental
Bienvenue! Veuillez vous connecter pour continuer.
Nom d'utilisateur ou email
Mot de passe
Rester connecté
Mot de passe oublié?
Se connecter

*Figure 45: Digimental Admin Login*

The figure above illustrates the login page for a Digimental admin or employee. The user must enter a valid account that has been activated by an admin. The user can restore the password via the phone number. Finally, a user can stay logged in after exiting the platform.

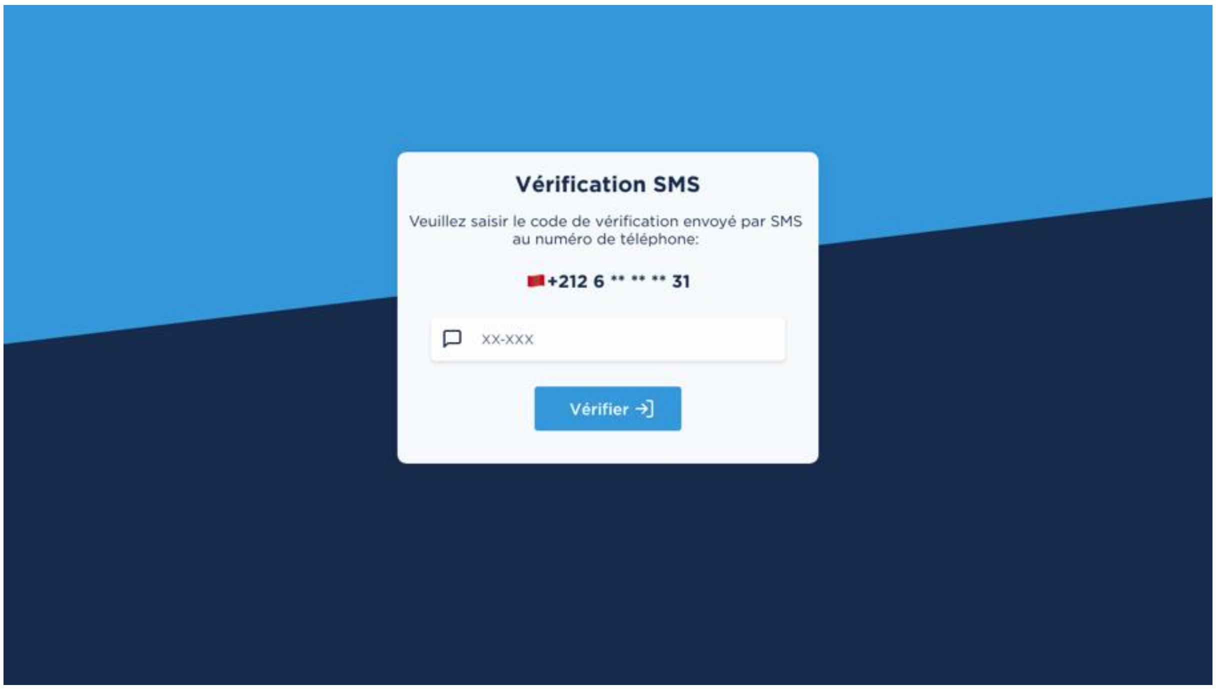

Vérification SMS
Veuillez saisir le code de vérification envoyé par SMS au numéro de téléphone:
+212 6 ** ** ** 31
XX-XXX
Vérifier

*Figure 46: Login SMS Verification*

For security purposes, any new login must go through SMS verification. The SMS/Email app (discussed before) will send an SMS through Twilio API to the associated phone number. If the user failed to verify an alert, SMS would be sent to the account owner. This will serve a layer of protection from password leaks.

### 4.2.2 Admin dashboard

The role of the dashboard is to provide important KPIs and efficiently visualize them. This will allow the decision-makers to get an insight into the whole system. Additionally, the dashboard serves as a navigation hub among all the other pages and sections. The whole platform contains the following sections:

1. **"Tableau de bord"**: the main dashboard also shown in the figure above.
2. **"Hôtels"**: Section to manage, add, view, and delete hotels. The search bar always redirects to this section with the search keyword.
3. **"Plans d'abonnement"**: Section to manage, add, and delete subscription plans. Setting plans help the company maximize its profit and consolidate its offerings.

4. **"Factures / Abonnement"**: Section for managing, creating, sending invoices, and verifying subscriptions.
5. **"Messagerie"**: Section for communicating among Digimental employees.
6. **"Service Client":** Section dedicated to customer support representatives. It helps them locate accounts quickly, see recent activities, force account log out, and revoke access. This is crucial to having a great customer support experience.
7. **"Employés"**: Section to manage, add, and delete employees.
8. **"Paramètres"**: General settings of the app.

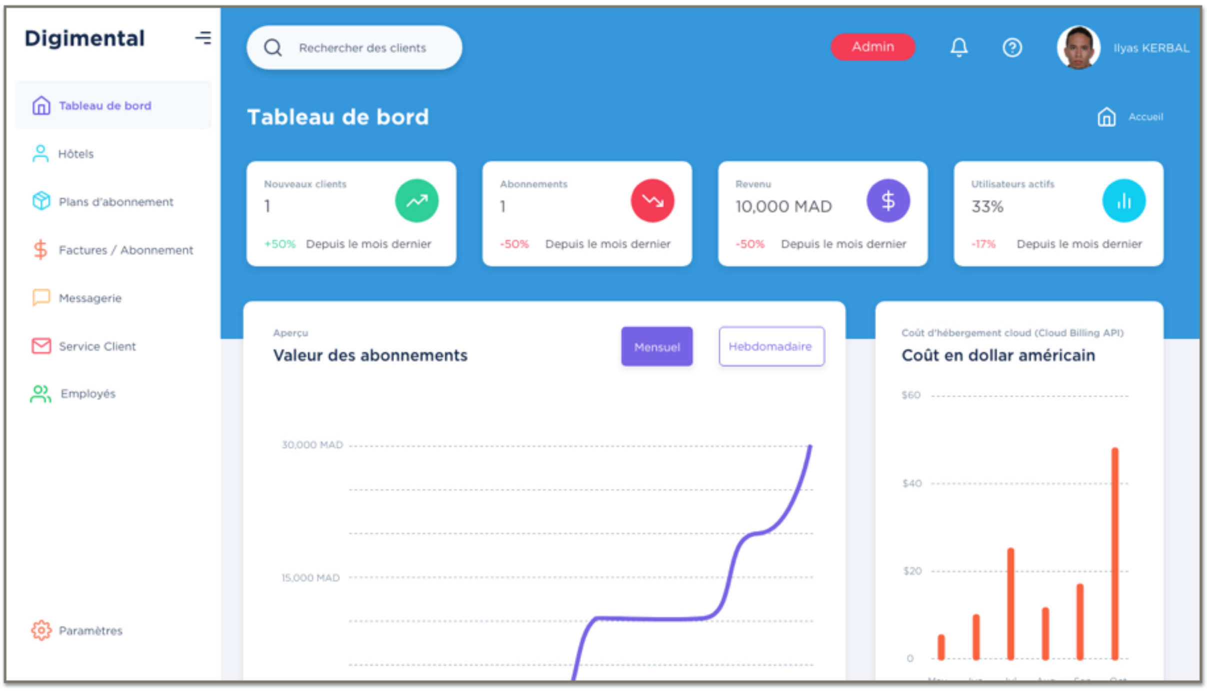


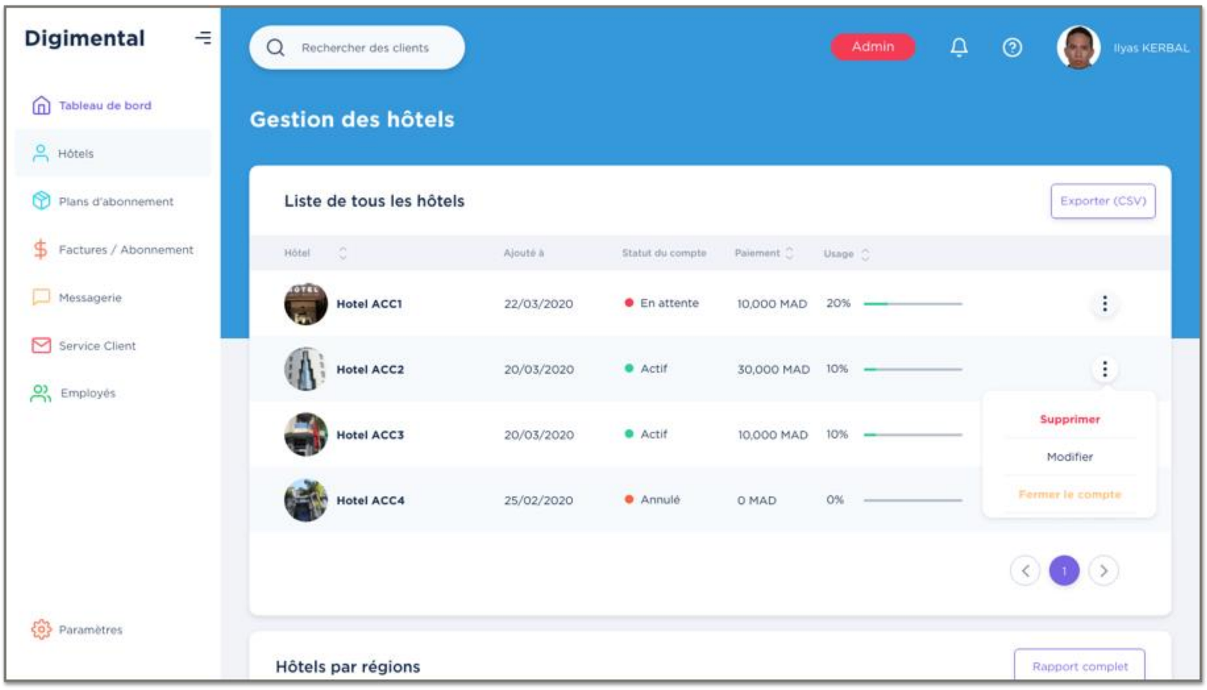
*Figure 47: Digimental Admin Dashboard*

### 4.2.3 Hotels Management

This section is for managing, adding, deleting hotels from the system. It has some basic sorting and pagination system to help locate the hotels quickly.

For a keyword search, the user must use the search bar located at the top of the page. Finally, the export option downloads all the hotels' data from the database in CSV format. Later, new features will be added for advanced search and map view for different hotels.

The simple pagination system is JavaScript based and does not require a page reload. This minimizes user distraction, data usage, and saves Digimental cloud resources. Additionally, each critical step will require admin approval, e.g., removing a hotel, suspending accounts.



Gestion des hôtels

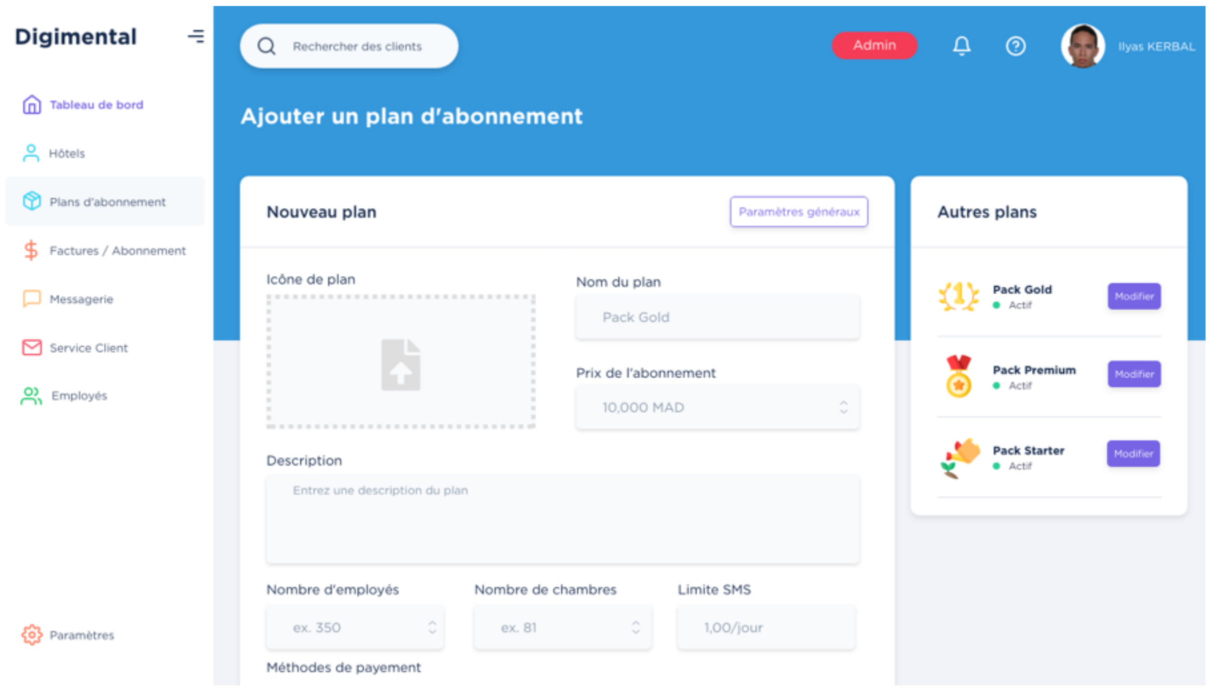
Liste de tous les hôtels

| Hôtel | Ajouté à | Statut du compte | Paiement | Usage |
|---|---|---|---|---|
| Hotel ACC1 | 22/03/2020 | ● En attente | 10,000 MAD | 20% |
| Hotel ACC2 | 20/03/2020 | ● Actif | 30,000 MAD | 10% |
| Hotel ACC3 | 20/03/2020 | ● Actif | 10,000 MAD | 10% |
| Hotel ACC4 | 25/02/2020 | ● Annulé | 0 MAD | 0% |

Hôtels par régions

*Figure 48: Hotel Management Section*

## 4.2.4 Adding a New Subscription Plan



Ajouter un plan d'abonnement

Nouveau plan

Icône de plan

Nom du plan: Pack Gold

Prix de l'abonnement: 10,000 MAD

Description: Entrez une description du plan

Nombre d'employés: ex. 350

Nombre de chambres: ex. 81

Limite SMS: 1,00/jour

Méthodes de payement

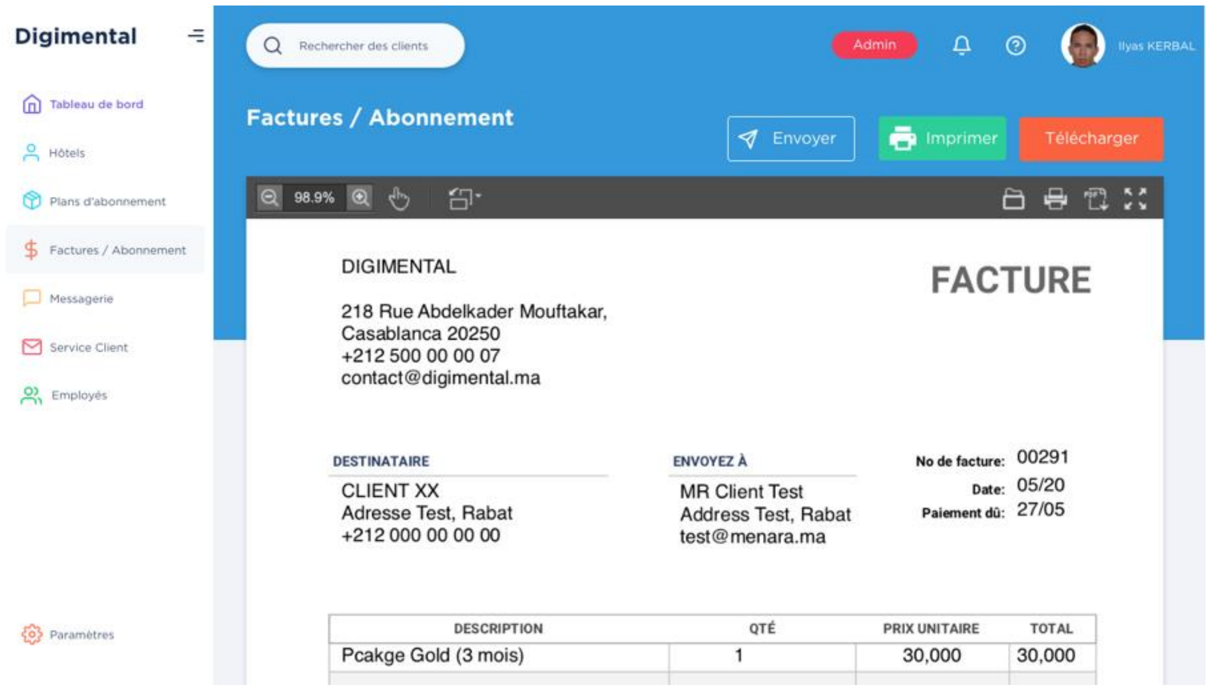
Autres plans

| | |
|---|---|
| Pack Gold | ● Actif |
| Pack Premium | ● Actif |
| Pack Starter | ● Actif |

*Figure 49: Adding a New Subscription Plan*

Subscription plans are a way of consolidating the company's offerings. Every plan comes with a specific amount of resources, whether fix or periodic. If the hotel exceeds these amounts, it must upgrade to an upper plan. Plans define how much money Digimental could charge and the cycle of the billing.

## 4.2.5 Serving PDF Invoices



DIGIMENTAL

**FACTURE**

218 Rue Abdelkader Mouftakar,
Casablanca 20250
+212 500 00 00 07
contact@digimental.ma

DESTINATAIRE
CLIENT XX
Adresse Test, Rabat
+212 000 00 00 00

ENVOYEZ À
MR Client Test
Address Test, Rabat
test@menara.ma

| | |
|---|---|
| No de facture: | 00291 |
| Date: | 05/20 |
| Paiement dû: | 27/05 |

| DESCRIPTION | QTÉ | PRIX UNITAIRE | TOTAL |
|---|---|---|---|
| Pcakge Gold (3 mois) | 1 | 30,000 | 30,000 |

*Figure 50: Serving a PDF Invoice*

After successful payment and account verification, the backend generates a PDF invoice for the customer. All pending and past invoices are accessible through the "Factures / Abonnement" section. Any PDF invoice can be downloaded, printed, or sent to the customer via email.

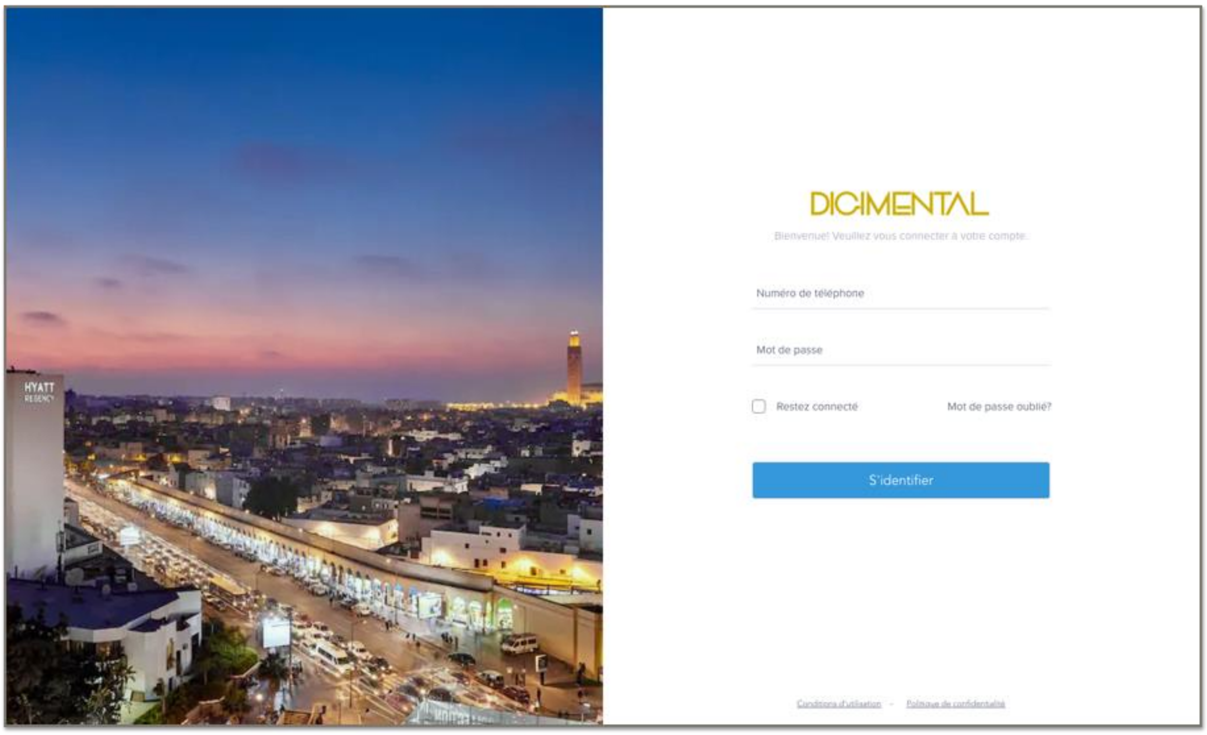
This section saves time on laborious and repetitive tasks. Finally, Digimental can easily replace the invoice template from the "Paramètres" section (template fields must be respected).

## 4.2.6 Hotels' Platform Login

Like the previous login page, this one requires a mobile phone to login instead of an email or username. The reason behind this is to smoothen the login experience (as a Digimental requirement). After entering the credentials, the user will be prompted an SMS verification.

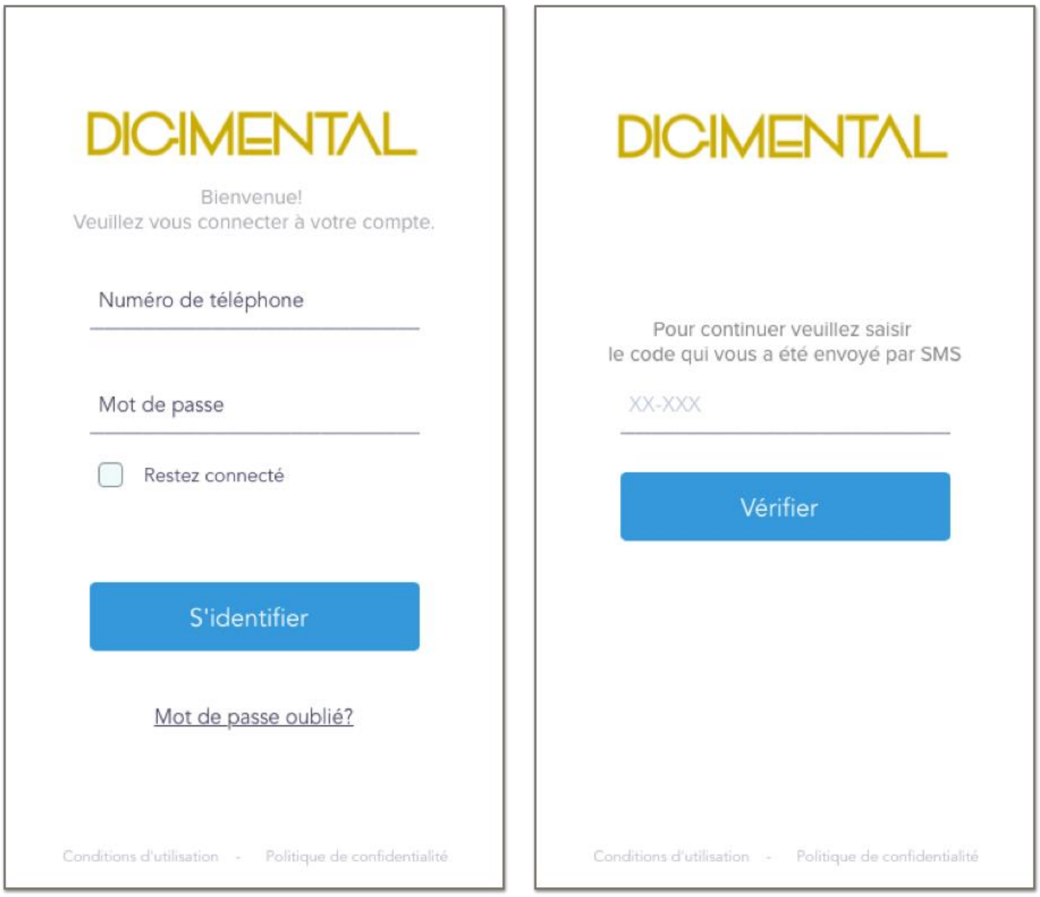
For mobile browsers, the login experience is no different. As discussed previously, the design of the app must be responsive and mobile-friendly. This can be achieved by leveraging the power of CSS media queries and Javascript.

The figures below illustrate the login page on desktop and mobile.



Bienvenue! Veuillez vous connecter à votre compte.

Figure 51: Hotels' Platform Login (Desktop)



Bienvenue!
Veuillez vous connecter à votre compte.

Pour continuer veuillez saisir
le code qui vous a été envoyé par SMS

Figure 52: Hotels' Platform Login (Mobile)

### 4.2.7 Staff Associated With Multiple Hotels

This project is a SaaS application with multiple-tenants. Hence, staff may be associated with two or more hotels. In this case, the staff will be prompt to select which hotel to log in.
For security reasons, the staff cannot switch hotels after log in. This will prevent undesired access between hotels and causing problems.

In case of a problem, the user can open a ticket or call Digimental by clicking on the link *"J'ai un problème"*.

The figure below illustrates the interface on mobile (same as Desktop).

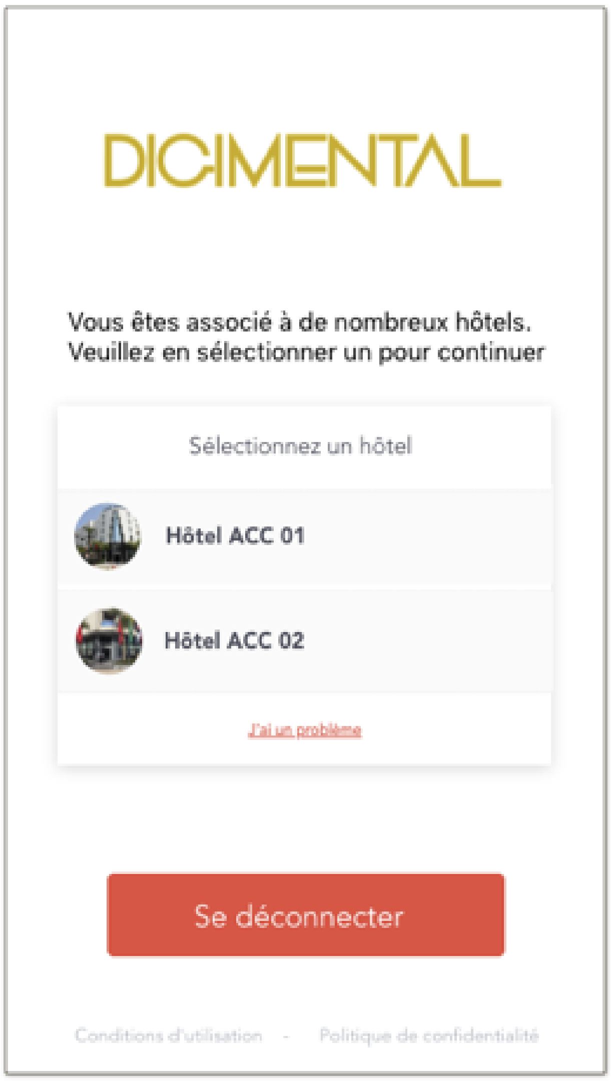
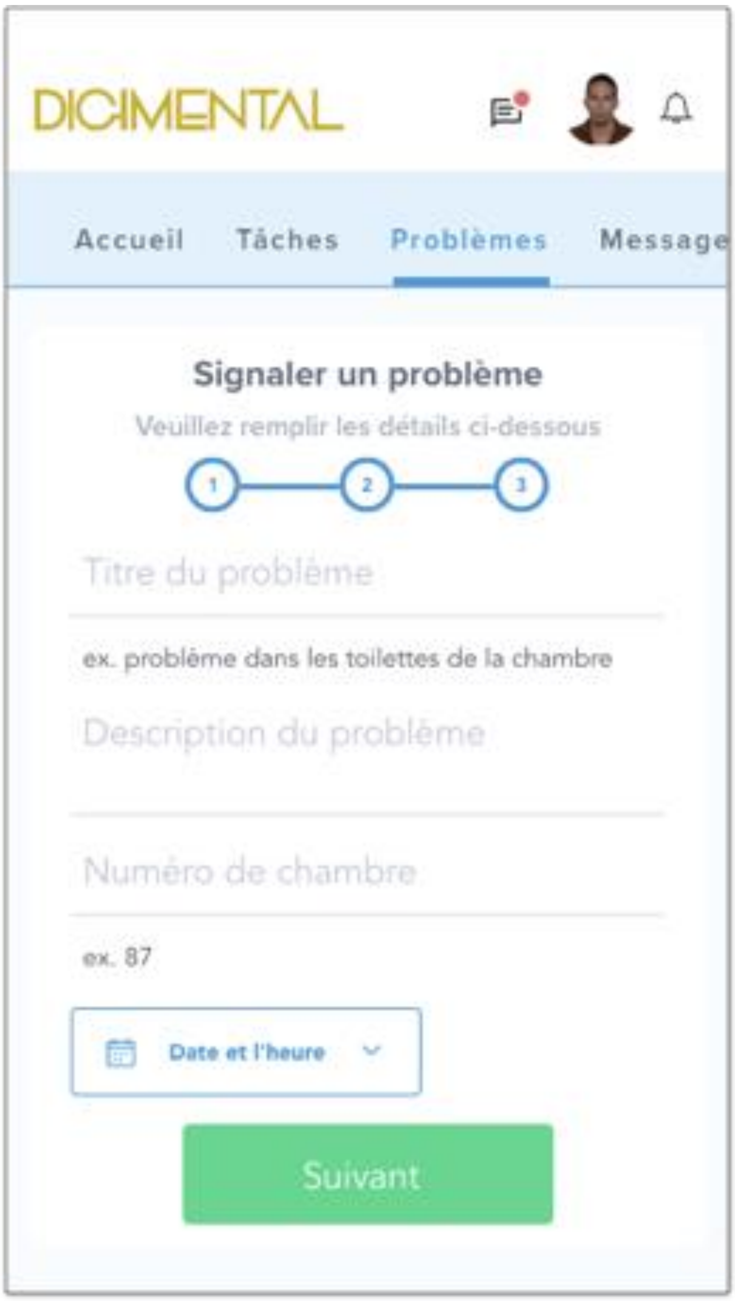

Vous êtes associé à de nombreux hôtels. Veuillez en sélectionner un pour continuer

*Figure 53: Hotel Selection Interface*

### 4.2.8 Scenario of Reporting an Issue

Let us examine the simple scenario of reporting an issue in a room by a cleaning service. The user must be logged in. After actively putting the room "on hold," the user will have a button to report a problem.

Any reported problem will be visible in the "Problèmes" section of the dashboard. Then, the user can enter the issue information.



Signaler un problème

Veuillez remplir les détails ci-dessous

Titre du problème

ex. problème dans les toilettes de la chambre

Description du problème

Numéro de chambre

ex. 87

*Figure 54: The First Step of Reporting an Issue*

After filling the necessary information, the user needs to select the problem's urgency. This steep requires only a simple radio group input.

This steep is vital to classify issues and add a sense of priority to the decision.

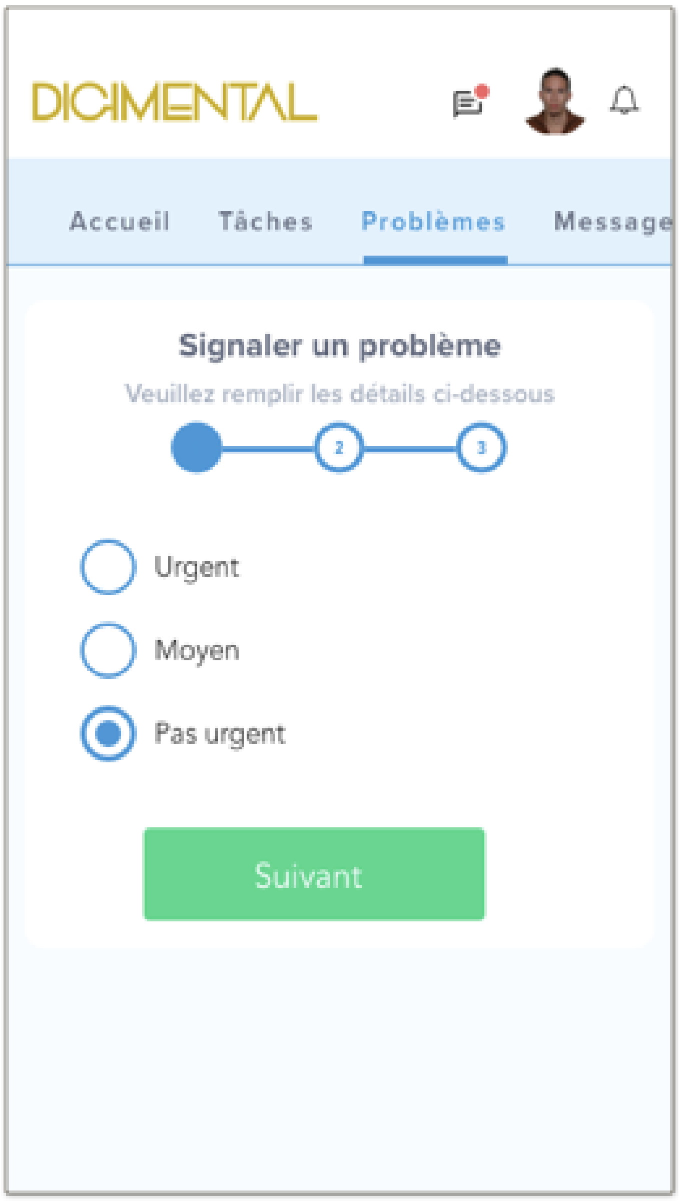


*Figure 55: Step Two of Reporting an Issue*

Finally, the user can include pictures or videos of the issue to help the manager make the right decision. These videos or images can be uploaded from the gallery of the phone or the file system of the computer.

After successfully uploading the files, the user will be redirected to a status page.

The figures below illustrate the step 3 and the status page.

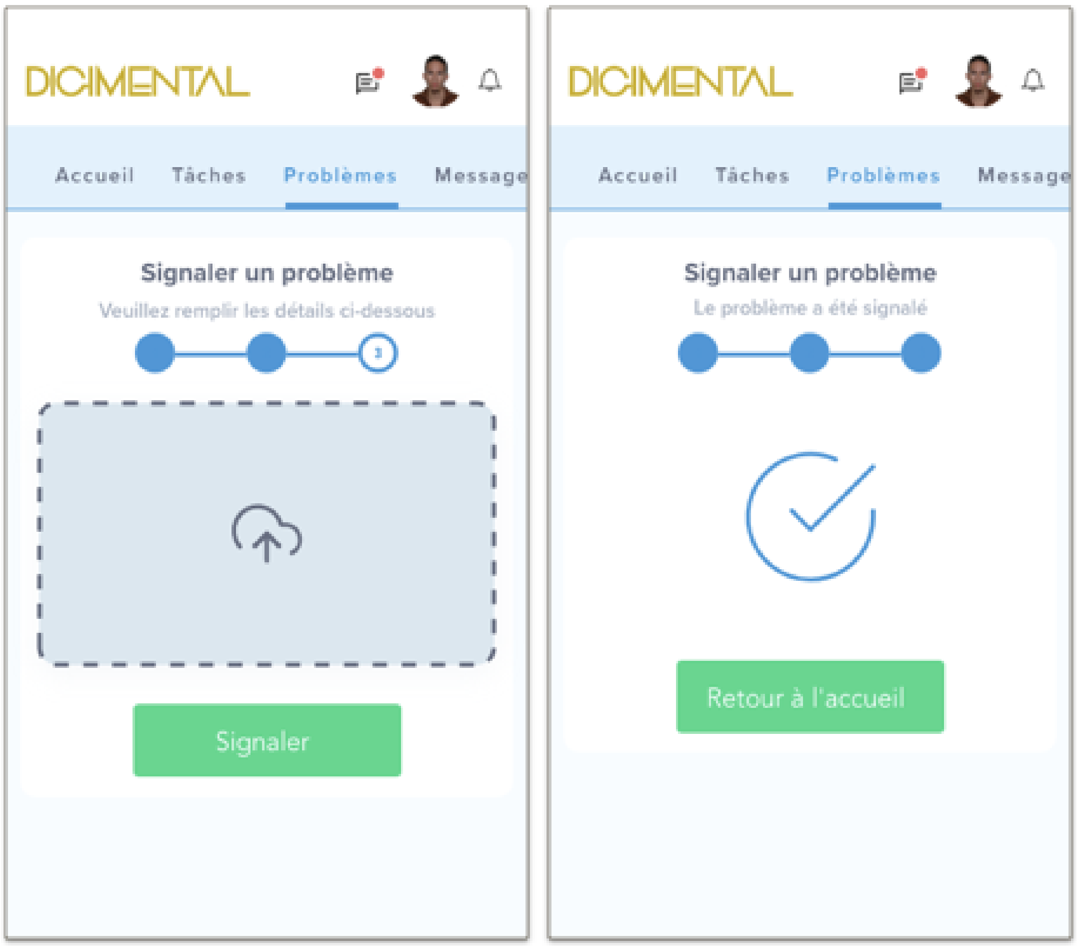


*Figure 56: Step Three and Four of Reporting an Issue*

### 4.2.9 Cloud Functions Memory Usage

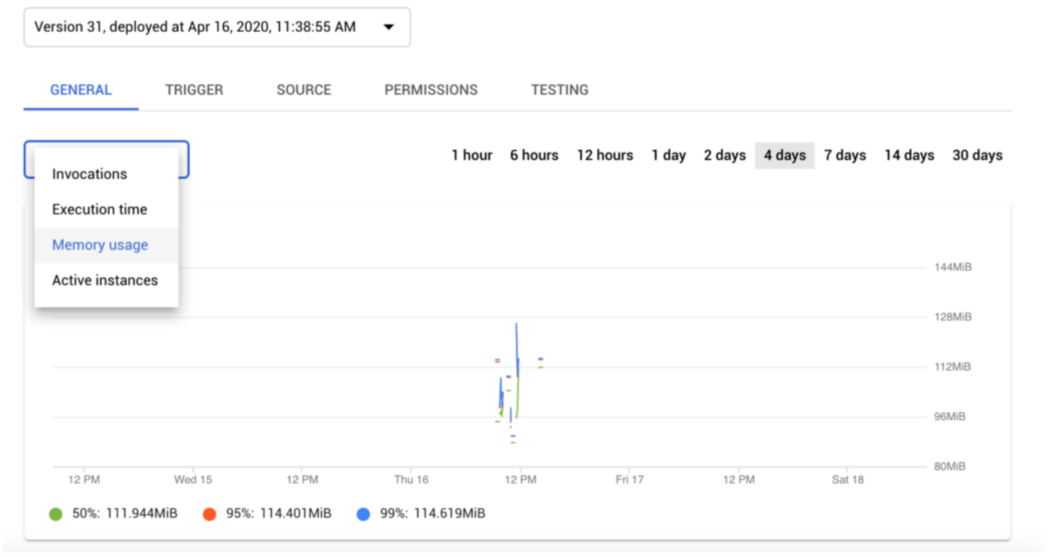


*Figure 57: Viewing memory usage in Cloud Functions*

### 4.2.10 Coclusion

This chapter presents the last stage of the project, which is that of realization and implementation. The latter highlights the development of the functionalities achieved so far. All the realization so far is just the "tip of the iceberg." The critical work is done, creating reusable Angular components that will make the development easier later.

# General conclusion

This graduation project consists of the development and implementation of a modern hotel internal management SaaS application while following the best practices of SaaSa and agile development.

The internship period is devoted at the start of training on practicing on simple applications and learning new technologies such as Angular, Google Cloud Platform, Firebase, Docker. Then, the focus on the design and the development of the application.

The significant technical difficulties encountered during the course of the internship are mainly related to the control of specific Frameworks or working with NoSql databases and optimizing requests. The supervision provided during the internship helped overcome most of the problems. Given this project, it is necessary to conduct a study of the progress of the project and to integrate all of the pre-established functionalities.

Overall, the project is an opportunity to experience closely working in a team of professionals and to apply the knowledge acquired during the training at the School of Information Sciences.

# References

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps Handbook: Amsterdam, Netherlands: Amsterdam University Press.

Moriarty, A. (2019, April 11). ▹ What is SaaS? 5 benefits of software as a service. Retrieved April 28, 2020, from https://www.cloudworldwideservices.com/en/what-is-saas-5-benefits-of-software-as-a-service/

Nx: Extensible Dev Tools for Monorepos. (n.d.). Retrieved May 4, 2020, from https://nx.dev/angular

Rouse, M. (2020, April 7). Software as a Service (SaaS). Retrieved April 25, 2020, from https://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service

Software as a service examples. (n.d.). Retrieved April 26, 2020, from https://www.hometowninsurance.com/blog/6uaol.php?bb6be5=software-as-a-service-examples

Stanton, T. (2020, January 17). An Introduction to Software as a Service (SaaS). Retrieved April 26, 2020, from https://www.whymeridian.com/an-introduction-to-software-as-a-service-saas

What is SaaS? (n.d.). Retrieved April 19, 2020, from https://www.redhat.com/ja/topics/cloud-computing/what-is-saas

Wiggins, A. (2017). The Twelve-Factor App. Retrieved April 15, 2020, from https://12factor.net/build-release-run