TECHNICAL AND QUALITY FACTOR ANALYSIS AND COMPARISON OF AWS CLOUD COMPUTING SERVICES FOR BUILDERS OF WEB AND MOBILE PROJECTS

Case Visma Consulting Oy

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Science

Master’s Degree Programme in Software Engineering and Digital Transformation

2022

Juho Kontiainen

Examiner(s): Professor, Jari Porras

Associate Professor, Ari Happonen

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Science

Master's Degree Programme in Software Engineering and Digital Transformation

Juho Kontiainen

TECHNICAL AND QUALITY FACTOR ANALYSIS AND COMPARISON OF AWS CLOUD COMPUTING SERVICES FOR BUILDERS OF WEB AND MOBILE PROJECTS

Case Visma Consulting Oy

Master's thesis

2022

110 pages, 23 figures, 19 tables and 1 appendices

Examiner(s): Professor Jari Porras and Associate Professor Ari Happonen

Keywords: AWS, factor analysis, cloud computing, web project, mobile project

Cloud services have become an essential part of software development, and IT organizations must adapt to the new paradigms and features that come with them. The case company, Visma Consulting Oy, and its product development unit, Product Creation Services (PCS), are standardizing technologies for their small to medium-sized web and mobile projects. One of the related key themes is cloud services, which is the topic of this thesis.

This thesis focuses on determining which AWS computing services best suit the PCS unit's requirements and demands for small and medium-sized web and mobile projects from a technical quality standpoint. The research is broken down into three sections: 1) gathering technical needs and quality elements for using cloud services on web and mobile projects, 2) conducting a survey to establish the requirements and quality-related technical aspects of the case company, and 3) analyzing and comparing AWS computing services.

As a conclusion, a conceptual model for detecting quality criteria in cloud-related web and mobile applications was developed, as well as a use-case guide to assist organizations in making informed decisions when assessing cloud computing services for their projects.

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT School of Engineering Science

Tietotekniikka

Juho Kontiainen

AWS-laskentapalveluiden tekninen ja laatutekijäanalyysi ja vertailu web- ja mobiiliprojektien toteuttajille

Case Visma Consulting Oy

Tietotekniikan diplomityö

2022

110 sivua, 23 kuvaa, 19 taulukkoa ja 1 liitettä

Tarkastaja(t): Professori Jari Porras ja apulaisprofessori Ari Happonen

Avainsanat: AWS, tekijäanalyysi, pilvilaskenta, webprojekti, mobiiliprojekti

Pilvipalveluista on tullut olennainen osa ohjelmistokehitystä, ja IT-organisaatioiden on mukauduttava niiden mukana tuleviin uusiin paradigmoihin ja ominaisuuksiin. Tapausyritys Visma Consulting Oy ja sen tuotekehitysyksikkö Product Creation Services (PCS) standardoivat teknologioita pieniin ja keskisuuriin verkko- ja mobiiliprojekteihinsa. Yksi siihen liittyvistä avainteemoista on pilvipalvelut, joka on tämän opinnäytetyön aihe.

Tässä diplomityössä selvitetään, mitkä AWS-laskentapalvelut vastaavat parhaiten PCS-yksikön vaatimuksia pienille ja keskisuurille web- ja mobiiliprojekteille teknisestä näkökulmasta. Tutkimus on jaettu kolmeen osaan: 1) teknisten tarpeiden ja laatuelementtien kerääminen pilvipalvelujen käyttöön verkko- ja mobiiliprojekteissa, 2) kysely tapausyrityksen vaatimusten ja laatuun liittyvien teknisten näkökohtien selvittämiseksi sekä 3) AWS-laskentapalveluiden analysointi ja vertailu.

Työssä rakennetaan käsitteellinen malli pilvipalveluihin liittyvien verkko- ja mobiilisovellusten laatukriteerien havaitsemiseksi sekä käyttötapausopas, joka auttaa organisaatioita tekemään tietoisia päätöksiä arvioidessaan projekteihinsa liittyviä pilvipalveluita.

ACKNOWLEDGEMENTS

Above all, thanks to Ilkka Toivanen and Ari Happonen, my instructors, for their excellent coaching and sparring sessions. I'd want to express my gratitude to my coworkers, family, and friends for their continued support as I worked on this thesis. The case company, Visma Consulting Oy, deserves credit for providing me with a thesis topic that I am passionate about.

Finally, I'd want to express my gratitude to Kata for her unwavering support.

SYMBOLS AND ABBREVIATIONS

AKS	Azure Kubernetes Service
API	Application Programming Interface
AWS S3	Amazon Simple Cloud Storage
AWS	Amazon Web Services
CaaS	Containers as a Service
CC	Cloud Computing
CDC	Change Data Capture
CI/CD	Continuous Integration / Continuous Development
CSLCP	Cloud Software Life Cycle Process
DBaaS	Database as a Service
EC2	Amazon Elastic Compute Cloud
ECR	Amazon Elastic Container Registry
ECS	Amazon Elastic Container Service
FaaS	Function as a Service
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IoT	Internet of Things
IT	Information Technology
MBaaS	Mobile Backend as a Service
NaaS	Network as a Service
NIST	National Institute of Standards and Technology
PaaS	Platform-as-a-Service

PCS	Product Creation Services
RDS	Amazon Relational Database Service
REST	Representational state transfer
RPC	Remote Procedure Call
SaaS	Software-as-a-Service
SDLC	Software Development Life Cycle
SLA	Service Level Agreement
SME	Small to Medium-sized Enterprise
SOAP	Simple Object Access Protocol
VM	Virtual Machine
VPC	Virtual Private Cloud
WBA	Web-Based Application
XaaS	Everything as a Service

Table of contents

1 Introduction

Projects in the realm of software product development are diverse due to the wide range of customer needs and technologies available. With most software projects now incorporating cloud services (Cloud Computing Study, 2020, 2), those designing and developing software must be familiar with cloud service concepts, as well as the drawbacks and benefits of various types of cloud services, as well as the related factors and needs in specific projects that guide service selection. The case company Visma Consulting Oy's Product Creation Services (PCS) performs a wide range of software projects, with small to medium-sized web, mobile, and integration projects being the most common. These projects necessitate and make use of cloud services as a deployment solution, as well as Platform as a Service (PaaS) / Software as a Service (SaaS) products including user management, cloud storage, and databases. The cloud architecture team at PCS has picked AWS (Amazon Web Services) as the 'de facto' cloud platform for projects, however with such a wide range of services available and diverse project demands, deciding on the recommended services has proven problematic, therefore more research is required. The goal of this thesis is to determine the computing needs of the target company's web and mobile projects, as well as the optimal AWS cloud services for them. This chapter covers the study's background, a brief explanation of the case company and its current state, and the study's objectives and delimitations.

1.1 Background

In the start of 2020s, the vast majority (90%) of IT organizations either are already utilizing cloud services or are planning to do so as according to a study made by EDG (Cloud Computing Study, 2020, 2). Therefore, it can be generalized that most software projects done in these days involve cloud services in a one way or another. Cloud services are not a new thing by any means, the idea for the cloud concept comes back from the 1960s by researchers, such as John McCarthy and Douglas Parkhill, who suggested that computation in the future would be performed by public services, renting computation capacity from somewhere else, providing users access to a variety of services through Internet (Durao et al., 2014, 1322). The development of cloud computing (CC) finally took off in the 2000s, as users' needs to expand their information systems globally increased, examples of companies that had adapted CC successfully emerged and as the industry leaders such as Intel, Google and Amazon started adapting CC as a concept. Many organizations in the 2000s built their own private clouds and on-premises solutions, as using public clouds was not as popular as they are today due to the security issues in the early public cloud services. In the late 2010s the focus of cloud services began to shift more towards being developer-friendly and developer-driven, meaning that the services provided started to be more targeted towards application and software developers, and the same trend has continued to this day. Additionally, in late 2010s, open data and public access to business information became more popular among industrial players too, as it offered new possibilities to reduce costs and speed up global business operations development efforts. (Arutyunov, 2012, 173-174; Foote, 2017; Metso et al., 2019; Metso et al., 2020)

Amazon Web Services (AWS), launched by Amazon back in 2006, is currently the most widely used cloud platform according to a survey made by Stackoverflow, Google's Google Cloud Platform (GCP) being the second most popular and Microsoft's Azure third most popular (Stack Overflow, 2021). Since 2006, the cloud services provided by AWS has grown from just a few services to over 200 fully featured services (Amazon Web Services, 2022z). The services provided range from compute, storage, and databases to networking, IoT and analytics. Some of the well-known services include Amazon Elastic Compute Cloud (Amazon EC2), AWS Lambda, Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (RDS) and Amazon CloudFront (Amazon Web Services, 2022z).

Developers looking to host a set of API (Application Programming Interface) endpoints might find themselves using Amazon EC2, where they need to handle the management of operating systems, configuration of firewall and installation of any software on the instances and where AWS handles the infrastructure including hardware, networking and the actual facilities – or they might consider using AWS Lambda, where the developer can run the code without having to manage the underlying infrastructure at all, as AWS handles all of the operational and administrative activities behind the service (Amazon Web Services, 2022[s,v]). In some cases, the developer might need complete control of the service, custom utilities installed on the service or longer and heavier workloads so a particular service such as AWS Lambda would not be a great fit. The broad range of services available, varying levels of configuration needed and how well the services can be integrated to and from are some of the things that makes cloud platforms so appealing for application developers. For instance, some projects may require external data storage, user management and databases – being able to nitpick the needed services along with the needed level of control from the platform brings flexibility and saves developer time for the project. However, choosing services for the project without thoroughly researching the project needs can lead to poor cost-effectiveness and performance issues because different services are often designed for different use-cases and the pricing models vary (Amazon Web Services, 2022h). The following subsection reveals the case description and forms the basis for the research objectives, research questions and delimitations of the study.

1.2 Case description

Visma Consulting Oy is a medium-sized consulting company that provides IT-consulting for complex information system projects, service design and digital services (Visma Consulting Oy, 2022). Demand for the thesis topic comes from the Product Creation Services (PCS) unit, which is responsible for the case company's product development as a part of their internal process of “Standardization of technologies for small and medium-sized projects”. The case company’s projects use a variety of technologies and tools, leading to a fragmentation of developers' skills. The information transferred between projects remains weak because the technologies and tools used are not standardized - even if the next project has similar project needs it may have a completely different stack of technologies due to the general ambiguity about the recommended technologies. This has also made it more difficult to plan and implement competence development. Through standardization, the PCS unit expects to enhance development work in projects, enable replicability of Continuous Integration / Continuous Development (CI/CD) solutions, make it easier for developers to switch between different projects, improve quality and provide opportunities to instruct potential job seekers to study the so-called right things.

The standardization process involves different areas of development in the projects such as web and mobile projects, server functionalities and cloud platforms. The scope of this thesis is restricted to the cloud services on the AWS cloud platform as it is chosen as the primary cloud platform by the cloud and architecture team at PCS. Focus of the thesis is specifically on the computing services which allow for the deployment of server functionalities. In addition, some projects may require other services, such as external storage and databases, that should be taken into consideration when selecting the services to use, as integration between services may not be straightforward in all cases and might affect performance and cost. Due to these complexities, the selection criteria should be considered at a higher level and through large projects, so that the solutions built through the choices made are also scalable solutions at a later stage and should therefore not limit the applicability of the solution in the future. implementation.

The expertise of the company's developers is extensive, some focusing on web development, Internet-of-Things (IoT) and embedded solutions, and some on mobile development. Not everyone has vast experience with cloud services, nor they need to, but this means that they also do not know the differences and similarities between different cloud service models, or what the consequences of choosing a particular model will be. Although AWS provides quick guides, they can be interpreted in different ways and it would be to the company’s advantage if, through unified standardization, developers knew which cloud services are recommended for use and the typical use cases for them.

The cloud and architecture team has made preliminary recommendations for the use of cloud services, which are presented next as they will be used as a baseline during the study. The recommended compute service is AWS Lambda, and the use of containerization is highly recommended. Using the Amazon Elastic Container Registry (AWS ECR) is preferred when working with container applications. Services are recommended to be built using Infrastructure-as-code software tools such as Terraform. In projects where user management and application-level authentication are required, they should use AWS Cognito and AWS Amplify.

1.3 Objectives and delimitations

The goal of this study is to determine which AWS computing services best suit Visma Consulting Oy's PCS team’s small and medium-sized web and mobile projects. The suitability of computing services is evaluated in terms of technical quality based on project factors and requirements. The research is conducted as a case study, consisting of both qualitative and quantitative data gathered through literature reviews, a survey and the analysis and comparison of AWS computing services. A literature review is used to gather the factors and needs of web and mobile application projects, following which a conceptual model is constructed based on prior research, with an emphasis on the technical aspects of web and mobile apps and cloud services. The conceptual model is constructed to establish theoretical framework to create a survey for the employees of the case company. A survey is conducted to prioritize the most critical factors and needs from the perspective of case company’s software developers and management, as well as to identify them. Following the survey, AWS computing services are assessed and compared to resolve which services offer the best quality results for the case company. The research aims to answer the following research questions:

RQ:1 What requirements and quality factors should be considered when choosing cloud computing services for web and mobile application projects from a technical viewpoint?

The purpose of the first research question is to find out the essential quality factors and technical requirements of software projects when choosing cloud services, emphasizing the design, implementation, and maintenance of the server side of the web and mobile projects. A literature review on cloud computing and its concepts is conducted to establish a knowledge base for cloud computing, including the technical limits and requirements that it entails. A literature review on web and mobile application projects is conducted to identify the main characteristics and the related typical back-end requirements. Finally, the quality factors are identified by developing a conceptual model based on web-based applications and web services, as well as by evaluating the nature of the cloud software development life cycle (SDLC). The literature review identifies prior research's ideas and concepts, and the conceptual model is built to connect the identified factors together. Because the focus is on web and mobile applications, this work does not cover other sorts of projects, thus it's possible that something is overlooked when analyzing the literature. The conceptual model is also made based on previous research, which itself may have shortcomings. Furthermore, the literature study and concept model are focused on the technical viewpoint, thus the perspectives of end-users and others involved in development process, for example, are not emphasized in the research. The purpose of the literature review and the conceptual model is to form a theoretical background for the following research question:

RQ:2 What are the most important quality-related technical factors and needs when choosing cloud computing services for the case company's typical web and mobile projects according to the case company's developers and management?

The second research question's objective is to determine what the most important project requirements and criteria are for the case company's web and mobile projects. This question is answered by a questionnaire, the content of which consists of the theoretical background presented previously as well as the literature reviews conducted on cloud computing and the characteristics of web and mobile applications as well as their development life cycle. The purpose of the survey is to collect as many responses as possible from the case company's developers and project managers to get a fundamental understanding of the factors that influence employee quality while working with cloud computing services The survey findings enable a comparison and evaluation of the services against the factors that were prioritized along with project needs. However, because the survey relies significantly on the established conceptual model, if the model is flawed, the survey responses will be as well. There's also a chance that the survey won't get enough responses to be credible, and the results won't accurately reflect the company's project demands and quality issues. The results of the survey will be used as a basis for the second and final research question:

RQ:3 Which of the AWS cloud computing services designed for hosting server applications offer the greatest quality for the case company's typical web and mobile projects based on the previously identified factors and needs?

The purpose of the third research question is to determine how effectively AWS' cloud computing services fit the requirements and factors of the case company's web and mobile projects. Preliminary research is conducted on cloud services in general, the different cloud service models present, AWS cloud computing and the services provided to form a theoretical basis for selecting the services suitable for server-end solutions, after which the evaluation of computing services can begin based on the previously gathered factors and needs. The computing services are evaluated by empirically collecting data based on survey responses and analyzing how well each service fits the identified and prioritized factors and needs. Based on the evaluation results, the case company is given a use-case guide to facilitate decision-making for selecting the best fitting AWS computing services for their projects and application workloads.

2 Cloud Computing

This chapter provides a definition of CC and the key concepts, including essential characteristics, service models and deployment models, and an introduction to the concept of serverless. The purpose is to form a strong knowledge domain about CC for the subsequent evaluation of cloud services for the case company's projects.

2.1 Characterization of CC

The National Institute of Standards and Technology (NIST) issued the most recent official definition of CC in 2011. They recognize that CC is an evolving paradigm, and their definition attempts to characterize the most important aspects revolving around CC (Mell et al., 2011, 1). It has been over 10 years since the definition by NIST was created and although the core aspects of CC have remained similar, it must be admitted that 10 years is a long time in information technology (IT). This is partly due the rapidly evolving nature of IT and the emergence of ground-breaking concepts such as container orchestration and micro-services that eventually led to a new service model, Function as a Service (*FaaS*), and which over time led to the concept of serverless (van Eyk et al., 2018, 3). Therefore, the NIST's definition is used as a baseline for the definition of CC and modern sources are used to complement parts of the definition.

The NIST defines cloud computing as: *"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."* (Mell et al., 2011, 1). The afore mentioned cloud model definition can be seen visualized in the figure below (**Figure 1**).

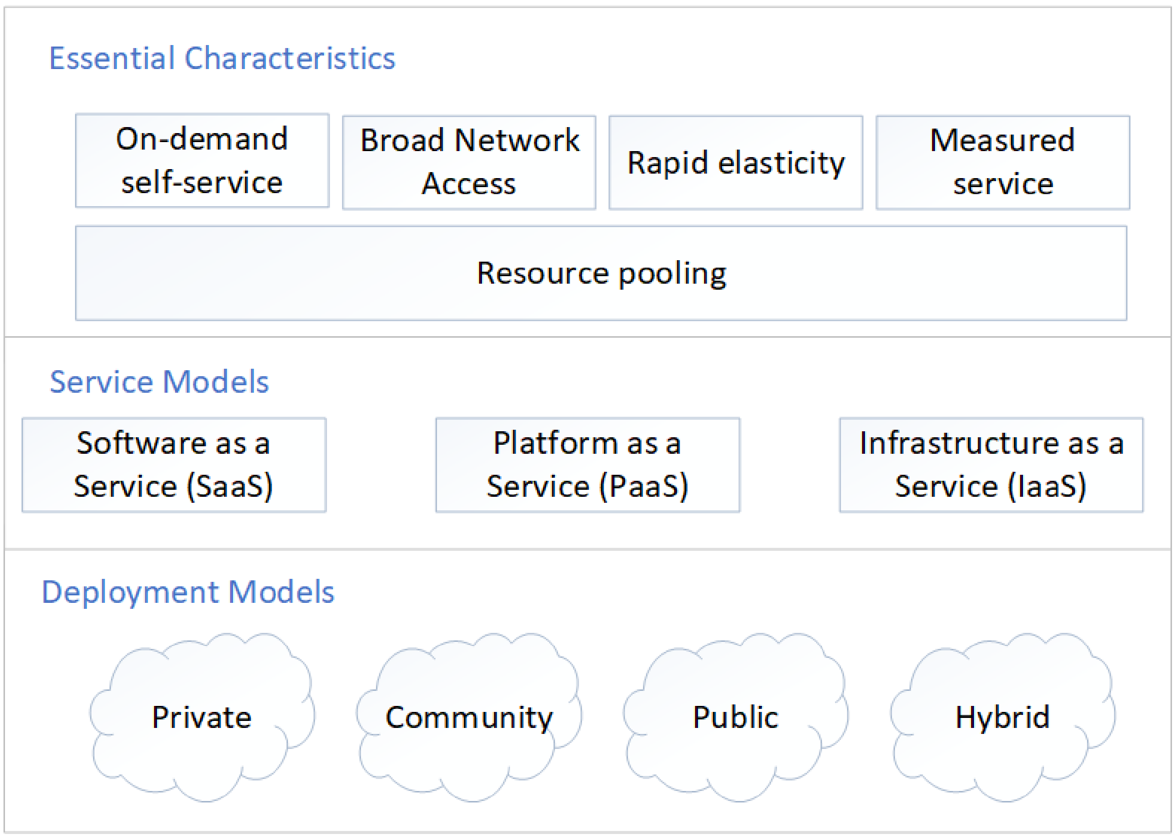


Figure 1. NIST Cloud Computing definition visualized (Mell et al., 2011, 2-3)

The most essential characteristics of cloud model, according to Mell et al. (2011), are On-demand self-service, Broad Network Access, Resource pooling, Rapid elasticity, and Measured service. The three specified service models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Finally, there are four different deployment modes categorized: private, community, public, and hybrid.

2.1.1 Essential Characteristics

On-demand self-service. The service's consumer can provision and configure services automatically and on-demand. No human interaction with the service provider is required to manage the service. This includes provisioning, halting, and removing services, changing the computational capabilities present with the service such as CPU, memory, storage, and networking capacity and the changing other service specific configurations (Amazon Web

Services, 2022i). Provisioning services is unilaterally in the hands of the consumer, independent of the service provider. (Mell et al., 2011, 2)

Broad network access. The service's consumer can use a wide range of client devices, including smartphones, PCs, laptops, and tablets, to access the service's capabilities across the network. The service can be accessed from virtually anywhere through the network by a thin or thick platform e.g., a browser or locally installed application. The network itself can be public or private – if the consumer has access to the network and is authorized, they can access the service. (Mell et al., 2011, 2)

Resource pooling. The service provider uses a multi-tenant model to pool and share the resources across multiple customers. In the multi-tenant model, individual consumers share the same physical computing unit while remaining completely unaware of one another thanks to the model, which allows a single software instance's infrastructure and hardware to serve multiple consumers (Brown et al., 2012). Both physical and virtual resources are shared using the model (e.g., processing and storage). The model works by reallocating the shared resources dynamically based on demand. Due to resource pooling, the consumers have no actual control or awareness over the exact location of their resources. However, depending on the service utilized, they may have control over the location at a higher abstraction level, such as nation, region, or availability zone. Resource pooling allows the service provider to utilize their resources efficiently, as computing units' computational capacity can be shared. (Amazon Web Services, 2022j; Mell et al., 2011, 2)

Rapid elasticity. The services can be deployed and delivered on an elastic basis to scale vertically in response to increasing or decreased traffic, for example. Vertical scaling refers to increasing and decreasing the number of active instances, scaling out means increasing and scaling in means decreasing (Amazon Web Services, 2022l). The scaling can be automatic or manual, depending on the services used. As a result, the available computer resources might look to the client to be nearly limitless, and they can be provided in any number at any time. (Mell et al., 2011, 2)

Measured service. The service provider manages and optimizes the service's resource utilization automatically. This is accomplished by employing metering (typically pay-per-use or charge-per-use) at an appropriate abstraction level for the service type (e.g., processing, storage, and bandwidth). The service's resource utilization can be tracked, regulated, and reported as a basis for invoicing, for example. Hence, a high level of transparency is achieved for both the provider and consumer. (Mell et al., 2011, 2)

2.1.2 Service Models

Along with these three main models (SaaS, PaaS, and IaaS), many varieties of “as a service” models exist (collectively known as X as a Service, XaaS) such as Network as a Service (NaaS), Database as a Service (DBaaS), Mobile Backend as a Service (MBaaS), Function as a Service (FaaS) and Containers as a Service (CaaS). The most relevant service models to the study along with the three main service models are FaaS, MBaaS and CaaS, as they are suitable for running server-side applications. Furthermore, FaaS and MBaaS are closely related to the concept of serverless, which is an important topic for the study. The concept of serverless will be explained in more detail in the up-coming section.

The study's most relevant service models, SaaS, MBaaS, FaaS, PaaS, CaaS and IaaS, are presented below, in order of abstraction level, beginning with SaaS, which has the highest abstraction level:

1) *Software as a Service (SaaS).* The consumer of the service is provided access to the provider’s software application through a thin platform, such as a web browser, or an API. Third-party suppliers who supply end-user apps to their clients on a subscription basis often employ this service model. The consumer may have limited access to application-level configuration settings such as managing users and customizability to the service such as changing colors, logos etc. Examples of SaaS products are Google Apps, Slack and Salesforce. (Mell et al., 2011, 2; Sarkar & Shah, 2018, 9)

2) *Mobile Backend as a Service (MBaaS)*. MBaaS, also referred to as Backend as a Service (BaaS), provides the consumer with a functional back end out of the box. These services provide generic components that a consumer can typically use through APIs to replace server-side components in their application. Utilizing such services frees up the time from coding and managing said functionalities themselves but produces vendor lock-in. Common *BaaS* service types include user management, push notifications and user authentication. Examples of MBaaS products are AWS Amplify, Firebase and Auth0. (Oulevay, 2021; Roberts, 2017, 6)

3) *Functions as a Service (FaaS).* The consumer of the service is provided with the capability to deploy individual functions or operations on the service provider's platform and run them in response to events. The functions stay idle until an event that is configured to invoke the function occurs. Consumer can configure the function to invoke from various synchronous sources (HTTP API Gateway) to asynchronous sources (cron). Access to cloud resources like database and storage is integrated by the vendor. Examples of FaaS products are AWS Lambda, Azure Functions and Google Cloud Functions. (Oulevay, 2021; Roberts, 2017, 7-9)

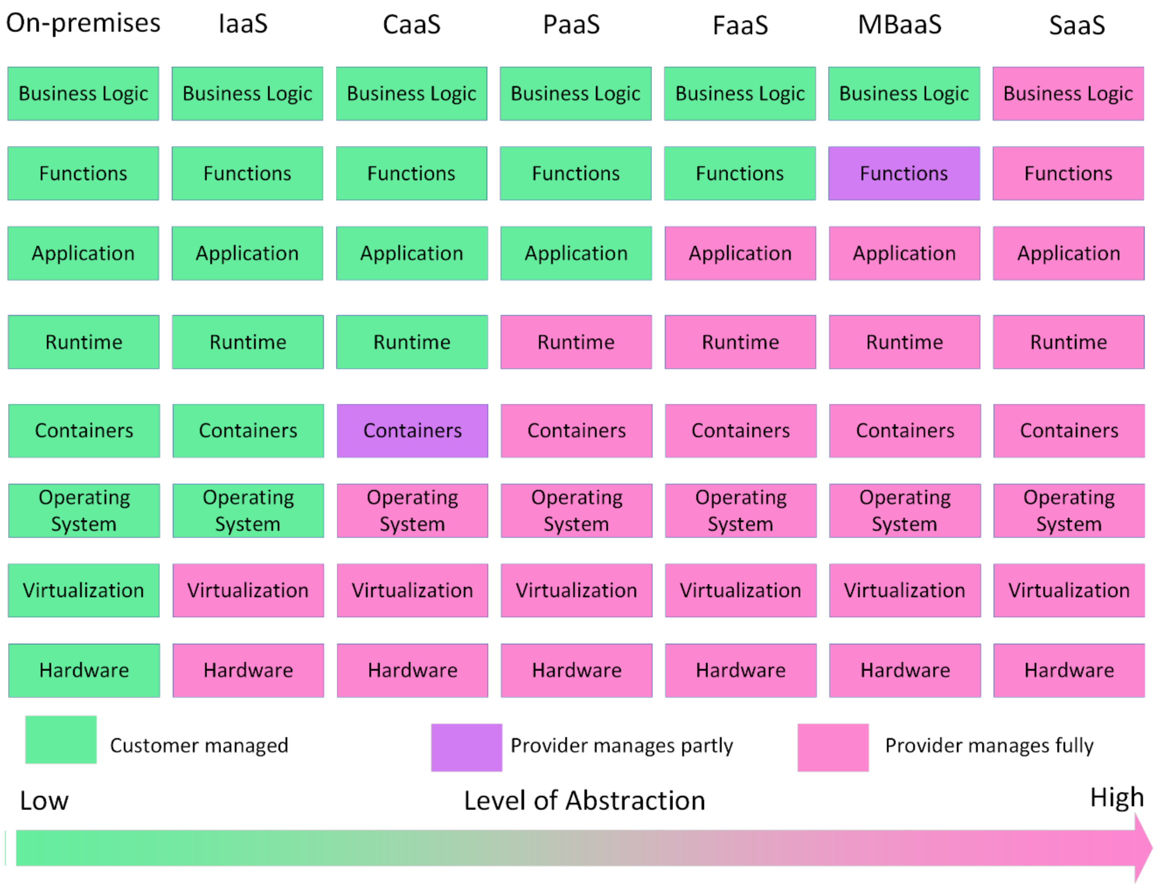
4) *Platform as a Service (PaaS)*. The consumer of the service is provided with certain core components such as databases, storage, and network as services for building their own applications. The consumer is presented with a capability to deploy on to the cloud provider's infrastructure by utilizing the supported services, tools, and libraries. Application developers often use this service model because it allows them to focus more on the functionality of the application instead of managing all the services they need, such as databases and storage, themselves. Examples of PaaS products are Amazon Elastic Beanstalk, Google App Engine and Azure Web Apps. (Mell et al., 2011, 2-3; Sarkar & Shah, 2018, 9)

5) *Containers as a Service (CaaS)*. The consumer of the service is provided the capability to use the provider's framework or orchestration platform, and infrastructure for deploying and managing container-based applications. The service automates the deployment, management and scaling for containers as well as allows the consumer to manage the entire life cycles of the container applications. The service enables the management of hundreds and thousands of containers and is suitable for use cases where the consumer has used a microservice architecture in their server-side application, uses

containers and need the service to be scalable, secure, and resilient. Examples of CaaS products are Amazon Elastic Container Service (ECS) and Azure Kubernetes Service (AKS). (Hussein et al., 2019, 1-3; Red Hat, 2019b; 2020)

6) *Infrastructure as a Service (IaaS)*. The service allows consumers to manage processing, storage, networking, and other computer resources to deploy and operate their own applications. This service model is the closest to running on-premises, but without the requirement for fixed capital investments in physical infrastructure because the client rents cloud service providers' infrastructure and only pays for what they use. Examples of IaaS products are Amazon EC2 and Azure Virtual Machine. (Mell et al., 2011, 2; Sarkar & Shah, 2018, 9)

The following figure (**Figure 2**) illustrates the individual and shared responsibilities of the customer and the cloud service provider between the afore mentioned service models in a typical application provisioned in the service. The figure presents the services ordered by the level of abstraction. The service models are contrasted by on-premises example, where the consumer is responsible for managing the entire infrastructure and its layers. The level of abstraction varies throughout the services, with PaaS having the highest level of abstraction and therefore being the quickest and easiest to configure and use, and IaaS having the lowest level and providing the highest level of configuration available. Regardless of used service model, the consumer has no access to the underlying cloud infrastructure, but increasingly manages the components and layers accessible in the stack, such as operating systems, runtime, and installed applications. In addition, in some service models, although primary managed by a service provider, the consumer may have limited control over some components, for example, in an IaaS, the consumer may have control over network components such as the host firewall (Mell et al., 2011, 3). Furthermore, in some service models, such as CaaS and MBaaS, the consumer and provider share the management of specific layer components. For example, in MBaaS, the service provider partially controls the "Functions" layer, which means that although the consumer can store and execute the function code, they are limited, for example, to use only the provided cloud APIs or restricted to do certain actions only. (Oulevay, 2021)



	On-premises	IaaS	CaaS	PaaS	FaaS	MBaaS	SaaS
Business Logic	🟩	🟩	🟩	🟩	🟩	🟩	🟥
Functions	🟩	🟩	🟩	🟩	🟩	🟪	🟥
Application	🟩	🟩	🟩	🟩	🟥	🟥	🟥
Runtime	🟩	🟩	🟩	🟥	🟥	🟥	🟥
Containers	🟩	🟩	🟪	🟥	🟥	🟥	🟥
Operating System	🟩	🟩	🟥	🟥	🟥	🟥	🟥
Virtualization	🟩	🟥	🟥	🟥	🟥	🟥	🟥
Hardware	🟥	🟥	🟥	🟥	🟥	🟥	🟥

🟩 Customer managed 🟪 Provider manages partly 🟥 Provider manages fully

Low
Level of Abstraction
High

Figure 2. Responsibilities across service models (Onrego, 2020; Oulevay, 2021)

2.1.3 Deployment Models

The four different deployment models defined by Mell et al. (2010) are Public, Private, Hybrid and Community cloud. One more type of deployment model is recognized by Sehgal and Bhatt (2018), Virtual Private Clouds (VPC), and will be expanded upon here as well (Sehgal & Bhatt, 2018, 43).

1) *Public Cloud.* The cloud infrastructure is accessible to the public. It may be accessed over the Internet from anywhere in the world. The computing resources are shared with other users typically through virtualization and the multi-tenant model. As the resources are shared among other tenants, the public nature of the deployment model, and the fact that the infrastructure is physically owned by the cloud provider, Public Cloud

encounters the greatest security issues. The cloud infrastructure exists on the cloud providers premises and can be owned and operated by organizations. Examples of providers that offer Public Cloud are Amazon Web Services, Microsoft Azure, and Google Cloud Platform. (Mell et al., 2011, 3; Oulevay, 2021; Sehgal & Bhatt, 2018, 3)

2) *Private Cloud*. The cloud infrastructure operated exclusively for a single enterprise. This deployment strategy is particularly capital expensive because it necessitates physical space and hardware investments, but it ensures security because it exists purely for the benefit of the organization and the services are offered internally through the organization's internal network. The cloud infrastructure can be owned and maintained by a company or a third party, and it can be located on or off-site. Examples of providers that offer this type of deployment are Microsoft and Red Hat. (Mell et al., 2011, 3; Oulevay, 2021; Sehgal & Bhatt, 2018, 3)

3) *Hybrid Cloud.* A “hybrid cloud” consists of two different cloud deployment models (i.e., public, private or community). The cloud infrastructures remain as separate entities and the data and application portability between the infrastructures are enabled by connecting them using standardized or proprietary technology. Typical example of usage is storing sensitive data on the private cloud and utilizing public cloud for other services. Organizations also often migrate parts of their infrastructure at a time to the public cloud, creating hybrid clouds. (Mell et al., 2011, 3; Oulevay, 2021; Sehgal & Bhatt, 2018, 43)

4) *Community Cloud.* In a Community Cloud, multiple businesses share the same cloud infrastructure. It is similar to Private Cloud, except instead of being owned, managed, and operated by a single organization, it can be owned, managed, and operated by several different organizations. Organizations use and share the costs of the services. The cloud infrastructure can exist on or off the premises. (Sehgal & Bhatt, 2018, 43; Mell et al., 2011, 3)

5) *Virtual Private Clouds (VPC)*. VPC is like an on-premises data center or Private Cloud, except it is housed in a Public Cloud environment. It can be seen as a virtual private cloud hosted inside a public cloud which is logically isolated. Cloud providers offer physical isolation for some of the infrastructure layers such as servers, storage, and networking but these often come with a fixed cost on top of the usage costs. An example of a VPC service is Amazon VPC. (Sehgal & Bhatt, 2018, 43-44)

As seen in the figure below (**Figure 3**), the economies of scale, control, and governance accessible to each deployment type vary. The public cloud has the greatest economies of scale and the lowest control and governance over the infrastructure, whereas the private cloud has the lowest economies of scale and the greatest control and governance over the infrastructure. To benefit from the benefits of greater economies of scale one must tradeoff some control and governance over the infrastructure. Greater economies of scale mean lower variable costs for the consumer (Amazon Web Services, 2022w). This has led many organizations to take advantage of economies of scale and lower prices in the public cloud,

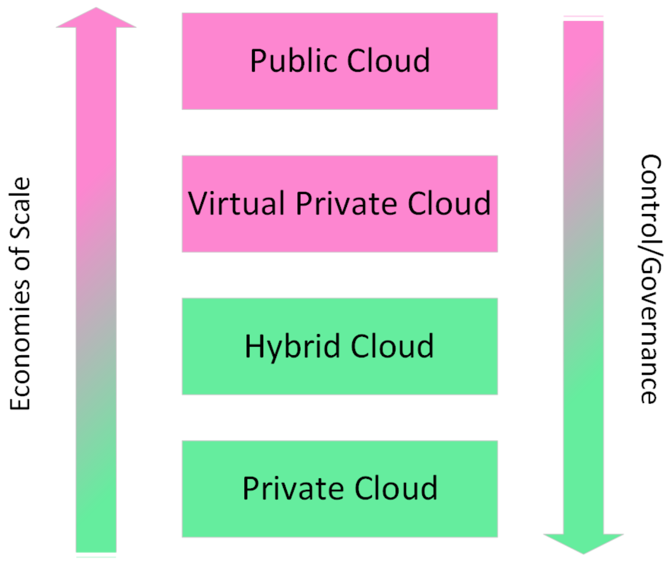


Figure 3. Economies of scale and control in the deployment models (Luo et al., 2019)

while maintaining a high level of governance and control over the sensitive data and operations by using the private cloud. In this way, the use of the hybrid cloud also solves some of the previously mentioned security issues that the public cloud faces and that many organizations are concerned about. (Oulevay, 2021)

2.2 Serverless computing

Serverless computing is a concept that allows consumers to develop and deploy applications without having to worry about the underlying servers and infrastructure. The launch of AWS

Lambda, a FaaS-product by Amazon in 2014, popularized the paradigm of serverless. The term itself can be misleading because servers are still required, but are abstracted by the service provider, so it seems to the consumer that there are no servers at all. Serverless is commonly referred to as FaaS but covers a wide range of technologies which can be grouped into FaaS and MBaaS (Roberts, 2017, 6). The concept of serverless computing is related to the study as it introduces an alternative paradigm to the traditional way of using constantly running servers, which can reduce costs and save developers time, but also includes some limitations. (Jonas et al., 2019, 7; CNCF, 2018)

2.2.1 Serverless vs Serverful

The figure below (**Figure 4**) illustrates how the deployment of server-side software is different in the traditional serverful (e.g., IaaS) and serverless context (e.g., FaaS). Deploying server-side applications in the traditional way requires a host instance, typically a container or a virtual machine (VM). The actual application is then deployed to the host instance and run in an operating system process. The application itself may contain various

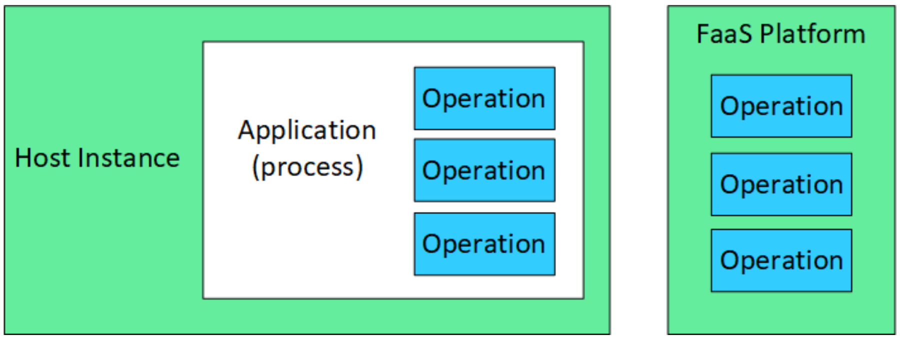


Figure 4. Traditional server-side software deployment vs serverless (Roberts, 2017, 8)

operations such as creating, reading, updating, and deleting resources, and connecting to other services, such as databases. Serverless strips away the requirements for a host instance and the application process, as the application's operations are individually uploaded to the FaaS platform and run when the specified events trigger them. The user's responsibility for numerous administrative chores such as selecting operating systems, libraries, and server instances, as well as managing the application's scaling, deployment, fault tolerance, monitoring, and logging, is transferred to the cloud provider. As a result, the application developers are left with more time to spend on creating business application logic. (Jonas et al., 2019, 5; Roberts, 2017, 8)

Finally, Jonas et al, (2019) recognize and conclude three critical distinctions between the serverless and serverful computing, which are referred to in the following sub-section:

1) *Decoupled computation and storage.* Computation and storage are separated, typically provided by separate cloud services.

2) *Executing code without managing resource allocation.* The cloud service automatically provides the required resources instead of the user having to request resources.

3) *Paying in proportion to resource used instead of for resources allocated.* Customer receives a variable bill based on the resource usage instead of a fixed one based on the allocated resources. (Jonas et al., 2019, 6)

2.2.2 Serverless architecture

The architecture of a serverless cloud is depicted in Figure 5 below (**Figure 5**). The serverless layer sits between the application and the main cloud platform layer, adding an additional degree of abstraction and making cloud services more accessible. Cloud Functions provide computing for the consumers and are complemented by an ecosystem of MBaaS-services to provide databases, user management and authentication for instance. The distinction made by Jonas et al., *Decoupled computation and storage* is observable here. The serverless architecture consists of numerous services decoupled into separate components in addition to the base cloud platform and its infrastructure. (Jonas et al., 2019, 5-6)

The serverless cloud works as an event-driven architecture, an example of this is how a FaaS platform operates. FaaS platform listens for user-defined events, instantiates a function

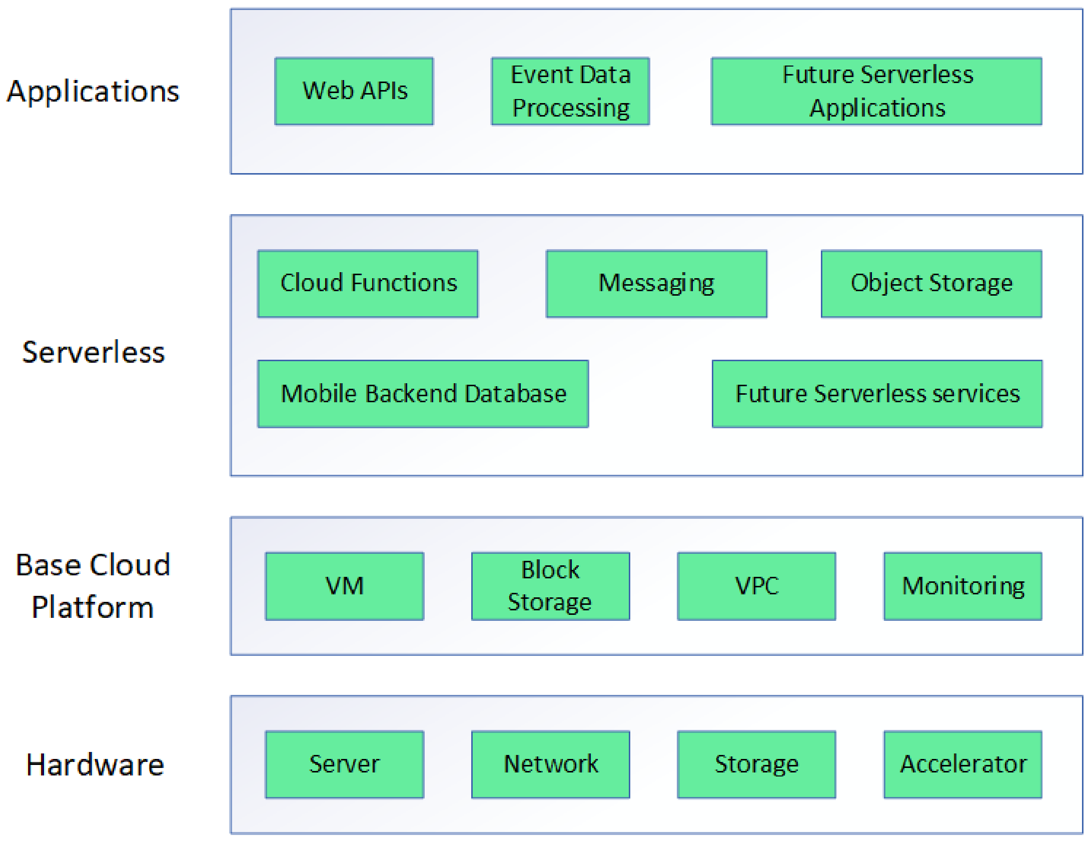


Figure 5. Serverless cloud architecture (Jonas et al., 2019, 6)

instance, and invokes it when the defined events occur (Roberts, 2017, 8). The different services can be accessed directly from the cloud functions through various policies and configurations, depending on the service provider (Amazon Web Services, 2022x) as the figure below presents (**Figure 6**). Furthermore, the user can select triggers for each function from a variety of event sources supplied by the cloud provider. In the figure, the authentication for each request is handled by Amazon Cognito. API Gateway distributes API calls to the Lambda functions, which have access to variety of services such databases (Amazon RDS), cloud storage (Amazon S3) and Amazon Simple Notification Service (Amazon SNS). Amazon SNS can be used to send notifications and trigger events from other AWS services, such as Amazon S3. The resources required to run the functions are automatically allocated when instantiated, and after execution, the resources are released as the function goes idle. This allows the user to execute code without managing resource allocation as well as for paying in proportion to the resources used using cost models such as the “pay-as-you-go” model, as mentioned by Jonas et al., 2019. The accessed cloud servi-

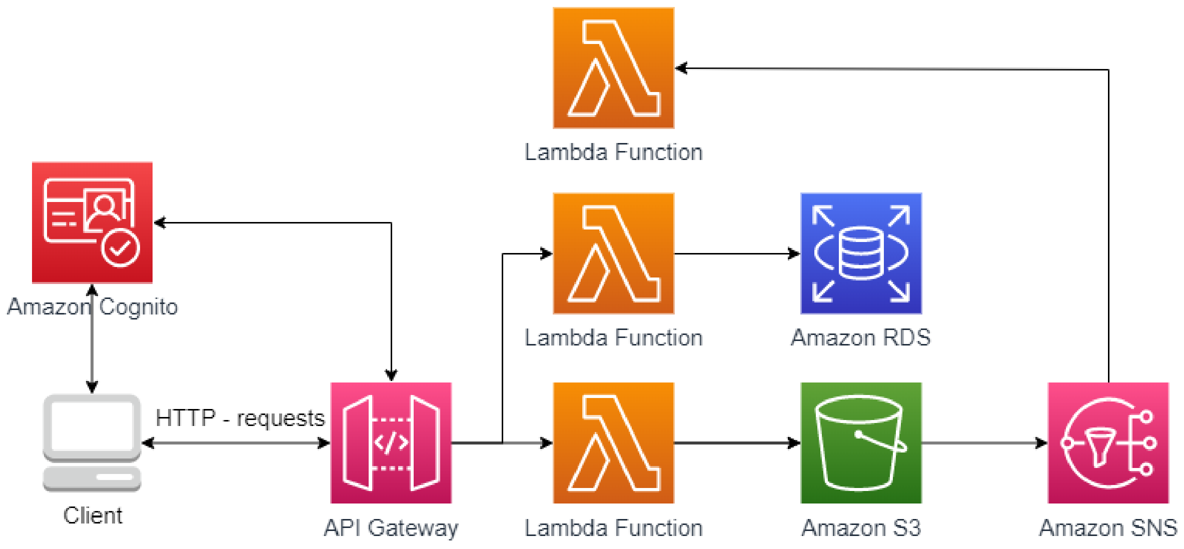


Figure 6. Serverless application architecture example (Amazon Web Services, 2022q)

ces are billed according to service-specific cost models, which can be for example, per-API call, per update or read requests, and storage used. (Jonas et al, 2019, 6-8; Roberts, 2017, 7-10)

2.2.3 Benefits and drawbacks

In general, a serverless approach should be chosen over a traditional approach when the workload is asynchronous, concurrent, and parallelizable into separate autonomous units, is infrequent, or has irregular demand, is stateless, and short-lived without the need for immediate start-up times (CNCF, 2018). Common use cases include (CNCF, 2018; Jonas et al., 2019, 9):

- HTTP REST APIs and web applications
- Data processing (batch jobs or scheduled tasks)
- Internet of Things (IoT)
- Chat bots
- Stream processing

- Change Data Capture (CDC)

Choosing a serverless approach requires considering the benefits and drawbacks of the computing model in relation to the computing needs of the project (CNCF, 2018). Benefits of a serverless approach are presented below (CNCF, 2018; Roberts, 2017, 19-26):

- *Reduced labor cost,* high degree of abstraction, the service provider handles most of the operations related work. A developer can focus on business logic and state, not infrastructure management.
- *Reduced risk.* Most technologies are operated and managed directly by the service provider. The service provider has its own teams per service that can quickly diagnose and fix problems, shortening expected downtime.
- *Reduced resource cost.* The pay-as-you-go concept allows the consumer to just pay for the computing time spent rather than having to plan, allocate, and provide resources and pay a fixed fee regardless of whether the resources are used at all.
- *Increased Flexibility of Scaling,* scaling is done automatically based on the demand or workload behalf of the service provider.
- *Shorter lead time,* complexity of building, deploying and operating applications is heavily reduced, resulting in shorter lead times, meaning it takes from conceptualization a feature to deploying it in a production environment.

The drawbacks from a serverless approach are presented below (CNCF, 2018; Jonas et al., 2019, 20; Roberts, 2017, 28-36):

- *Latency,* most of inter-component communication is done via HTTP protocol, which compared to other transports can be slower. The on-demand structure includes a “cold start” issue, where the function instance needs to be instantiated before invoking, adding latency for the first request.

- *Unpredictable costs*, comes with the pay-as-you-go model as the costs grow according to the demand – which can be difficult to forecast.
- *Loss of control*. A third party maintains and operates the platform. In contrast to the IaaS paradigm, where the user has full control over the software stack, adopting serverless implies giving up full management of the software stack.
- *Vendor lock-in*, not necessary from the components itself, but from the ecosystem provided by provider – how well the various components and services integrate.
- *Immaturity*, as a new and emerging computing model, it has less comprehensive and stable documentation, tools and best practices, lack of standardization, and is still an immature ecosystem.
- *State,* by design the components are stateless and should not persist information past their lifespan which can make state management tricky in some use cases.
- *Debugging,* dynamic runtime introduces challenges for debugging the application.

Using serverless provides monetary benefits not just by lowering resource costs, but also by reducing developer time because the service provider oversees operations. The provider is also responsible for ensuring that the used services operate and adhere to the agreed-upon SLAs, which reduces risk but can also increase risk because the consumer has no control over when and how the fault is resolved, as opposed to on-premises where the consumer would fix the issue themselves. Using a certain cloud service provider’s service can result in vendor lock-in that originates from the cloud ecosystem itself, and by using the services and the ecosystem, the consumer becomes completely reliant on the cloud service provider's roadmaps. The risk of vendor lock-in combined with the cloud service provider's ability to terminate specific services at any time can result in costly migration labors, as switching to a different service, service provider, or a on-premises solution in most cases is very time consuming. (CNCF, 2018; Jonas et al., 2019, 20; Roberts, 2017, 19-36)

3 Web and mobile applications

The first research question is partially answered in this chapter: *RQ:1. What requirements and quality factors should be considered when choosing cloud computing services for web and mobile application projects from a technical viewpoint?* by identifying the relevant needs and technical aspects related to web and mobile applications. To achieve this, a literature review on web and mobile applications and their architectures is provided, as well as a review of the characteristics and related concepts of server-side components.

3.1 Application architecture

Web applications consist of client- and server-side components that are combined into a complete application that uses the Web as an execution platform and runs over HTTP (Casteleyn et al., 2009, 2; Kuuskeri, 2014, 5). Complexity of these applications range from simple static HTML pages to dynamic applications that resemble installable desktop software (Casteleyn et al., 2009, 2). Web applications are typically accessed through a browser URL and do not require installation but can also be accessed through web services such as APIs (Kuuskeri, 2014, 10). Mobile applications, on the other hand, are software that can be installed on the device and often include offline features and resemble complex web applications (Amazon Web Services, 2022y).

Web and mobile application architecture is often divided into three layers, as shown in the diagram below: presentation tier, application tier, and data tier (**Figure 7**) (Casteleyn et al., 2009, 4; IBM Cloud Education, 2020). According to IBM and AWS, three-tier architecture is the most used software architecture for client-server applications like the study's key points of interest, mobile and web applications. The presentation tier is responsible for providing the user with a GUI (Graphical User Interface) to interact with. It displays and collects information from the user. The application tier is responsible for handling the business logic. It involves adding and modifying data in the data tier as well as processing information from the presentation tier. API calls are commonly used to communicate between the presentation and application levels. The data tier stores and manages the information processed by the application tier. The application tier mediates communication between the presentation and data tiers, so they don't connect directly. (IBM Cloud Education, 2020)

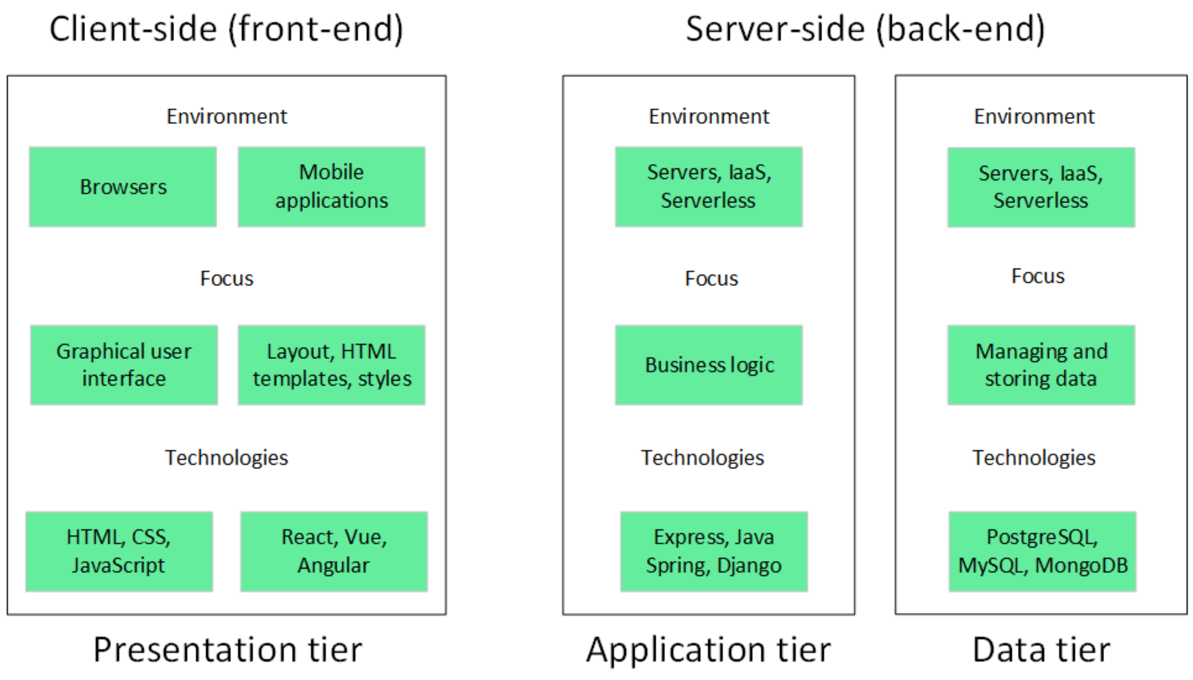


Figure 7. Three tier architecture (Casteleyn et al., 2009, 4; IBM Cloud Education, 2020)

Casteleyn et al. (2009) refer to multiple layers, while IBM refers to tiers; the definitions are different, as IBM puts it: "*A 'layer' refers to a functional division of the software, but a 'tier' refers to a functional division of the software that runs on infrastructure separate from the other divisions*". Three-layer applications handle presentation, application, and data layer in a single software, but layers are logically separated, and as a single tier; all layers run on the same infrastructure such as a calculator application on a mobile phone. Three-tier applications have separate layers of infrastructure that operate separately from each other. The decoupling of the tiers comes with several advantages such as reduction of network traffic, increased scalability, performance, and higher availability when compared to one- and two-tier architectures. However, the different N-tier architectures have limitations on how they can be operated and deployed in the cloud using different deployment models. Furthermore, multi-tier application architectures are fully supported by AWS (Amazon Web

Services, 2022a). (Casteleyn et al., 2009, 4; Hashida & Sakata, 2001, 715-716; IBM Cloud Education, 2020)

In this study, the presentation tier is referred as the front-end and the application and data tiers as the back-end of the application. The back-end's provided services that are accessed over the network are referred as web services. The case company's typical web-projects are built using three-tier architecture but implementations of two- and one-tier architectures exist as well. As illustrated in the figure (**Figure 7**) the front-end consists of environments such as browsers and mobile applications. The applications communicate with the back-end, e.g., servers and web services or MBaaS services through API-endpoints, to handle various tasks such as authentication and authorization, data management and user management (Amazon Web Services, 2022r). From the back-end perspective, it is trivial whether it is a mobile or a web application establishing connection as they both represent the presentation tier and typically communicate with the back-end via APIs over HTTP. Both may, however, have different back-end requirements; for example, mobile applications frequently use push notifications, whereas web applications rarely do (Amazon Web Services, 2022y), which must be factored into backend system design.

3.2 Back-end

The server-side components of a web and mobile applications are referred as the "back-end". The back-end consists of the application and data layers and is thus responsible for processing the business logic and data management of the applications. Back-end services can operate in a serverful or a serverless context such as traditional VMs, IaaS, FaaS and MBaaS. Application layer is often separated from the data layer to distinct tiers e.g., application logic and data logic running on separate servers or services. Common technologies used in the application tier are Express, Java Spring, Django and ASP.NET. The back-end communicates with the front-end, the presentation tier, via APIs through HTTP using architectural styles such as REST and GraphQL. Communication with the data tier and other services e.g., third-party authentication services and user management services, is operated via APIs. Furthermore, parts of a back-end system's functionality can be replaced with MBaaS services to save time developing and maintaining such features, but this exposes the development team at risk of vendor lock-in (Oulevay, 2021; Roberts, 2017, 6). This subchapter focuses on explaining the essential concepts and architectures related with back-end development. (IBM Cloud Education, 2020)

3.2.1 Application Programming Interfaces

Application Programming Interfaces (APIs) are used to enable communication between two software components using set of predefined protocols and definitions (Amazon Web Services, 2022r), e.g., front-end querying data from the back-end, which fetches data from the data tier, processes data and returns response back to the front-end (IBM Cloud Education, 2020). The application part of the acronym refers to a software with any function and interface defines the way one can communicate with the application. Web APIs typically communicate through HTTP, typically from a web browser to a web server and use formats such as XML (Extensible Markup Language) and JSON (JavaScript Object Notation). (Amazon Web Services, 2022r; Red Hat, 2019a)

Web APIs follow a client-server architecture and can be divided into several protocols and architecture styles, the most important for Web APIs which, according to Amazon Web Services, are: (Amazon Web Services, 2022r):

1) *Simple Object Access Protocol* (SOAP) is lightweight protocol that is used to exchange structured information. SOAP uses XML to exchange messages between client and server and is independent of the used transport layer and hence supports protocols such as HTTP and TCP (Transmission Control Protocol. SOAP was a popular API in the past but is less flexible compared to more modern APIs and is used less nowadays. (Amazon Web Services, 2022f; Kuuskeri, 2014, 13-14; W3C, 2007)

2) *Remote Procedure Calls* (RPC) is a protocol that allows client to run procedures remotely on the server. The client runs a procedure on the server, and the server replies with the output. RPC is designed specifically to support network applications

but can also be used for communication between different processes on same host. RPC is used to write network applications and requires the users to have knowledge over network theory, and hence is not often used in web and mobile applications. (Amazon Web Services, 2022r; Birrell & Nelson, 1984, 39-42; IBM, 2022a)

3) *WebSocket* is a protocol that enables a full duplex, two-way, connection between two applications, typically a web browser and a web server. WebSockets allow the client to have a long-lived and real-time duplex connection with the web server. It uses JSON objects to pass data. The protocol consists of two parts, a handshake and data transfer and it uses HTTP to establish connection. It is often used when there is a need for real-time duplex connection between the application and the server, such as instant messaging and gaming applications. (Amazon Web Services, 2022r; Fette & Melnikov, 2011; IBM, 2022b)

4) *Representational state transfer (REST)*. Previously mentioned SOAP, RPC and WebSocket are protocols, whereas REST is an architectural style for developing APIs. REST is protocol agnostic, but it is most used with HTTP. In REST, the client sends a request to the webserver, which according to the query can execute internal functions, and finally returns the data back to the client, usually in JSON or XML (see **Figure 8**). REST considers everything as a resource, everything that can be presented as an URL can be a resource and all resources must be addressable by an URL. It uses HTTP verbs, such as POST, GET, PUT and DELETE to define operations for the resources. The main principle of REST is being stateless. The web server does not store state, application state should only be stored on the client. REST is the most popular and flexible web API used nowadays. (Amazon Web Services, 2022r; Kuuskeri, 2014, 16-18; Microsoft Azure 2022)

5) *GraphQL* is a query language designed by Facebook for building APIs and is said to replace REST as it solves many of the issues present in the REST architecture. GraphQL uses HTTP as the protocol, and it only uses one endpoint, HTTP POST to query and mutate data. The client queries the server with a concrete structured JSON, telling the server exactly what resources to query or mutate. As seen from the figure below (**Figure 8**), REST requires each query and mutation to have its own endpoint, while GraphQL does everything through the one single endpoint. As the client can query just for the data they need, this solves problems such as under and over fetching

as in REST the client cannot explicitly tell the web server what data fields etc. to return. (Nusairat, 2020, 133-135; Lawi et al., 2021)

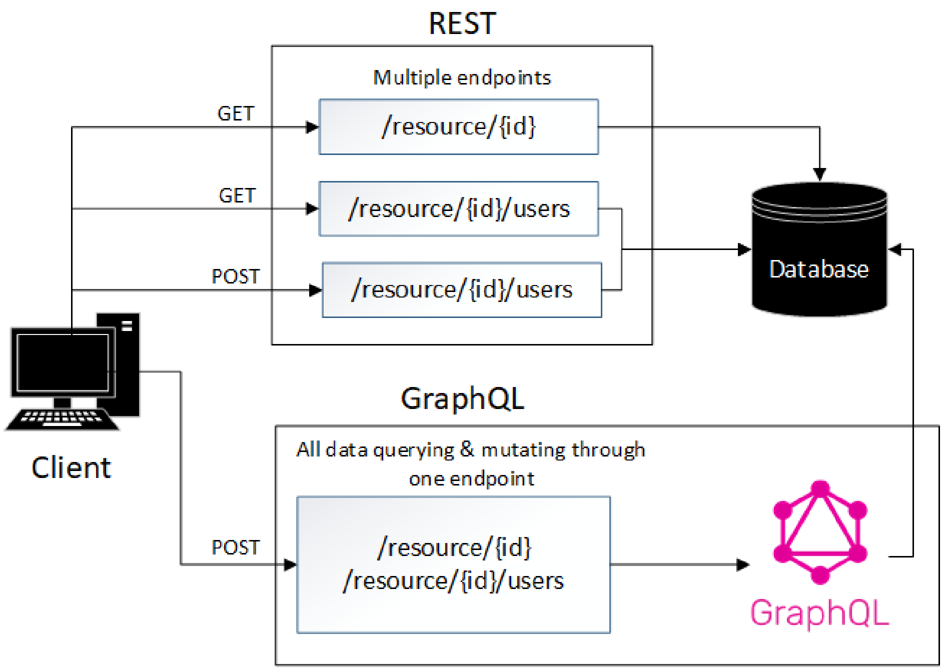


Figure 8. REST vs GraphQL example (Lawi et al., 2021)

The choice of different protocols and architecture styles for Web APIs may limit the choice of cloud services, for example, Amazon S3 currently supports SOAP, but new features are not supported, and the Amazon API Gateway does not support SOAP at all (Amazon Web Services, 2022[b,t]). However, it should be noted that this only applies to invoking cloud services, as when building their own services, they can reveal APIs that follow one or more of several protocols and architecture styles and still invoke web APIs using, for example, RPC, REST, and WebSockets (Microsoft, 2021).

3.2.2 Main back-end architectures

Back-end applications consists of numerous ways to architecture the provided services. Microservices and monolithic architectural techniques are two of the most used architectural designs, according to multiple sources (Amazon Web Services, 2022g; IBM, 2011).

Applications generally start with the traditional monolithic architecture and then migrate to microservices to enable better scaling (Atlassian n.d.). Microservices is a cloud native architectural paradigm used by large firms like Netflix and Amazon to power their applications and systems (Al-Debagy & Martinek, 2018; IBM, 2011). Microservices are frequently compared to the monolithic architectural paradigm, in which applications are made up of a single huge code base with several tightly coupled services rather than several smaller loosely coupled services (Al-Debagy & Martinek, 2018; IBM, 2011). Being a cloud native architecture, microservices paradigm is widely supported by cloud service providers such as AWS (Amazon Web Services, 2022g). Furthermore, Al-Debagy and Martinek (2018) list the following five benefits of employing microservices versus monolithic applications: 1) technology heterogeneity; each service can use different technologies, 2) failure of one component does not bring the entire system down, 3) scaling is achieved by scaling components rather than the entire monolithic application, 4) deployment is easier, as each component is deployed independently, and 5) the microservices architecture helps on reducing number of people working on same codebase (Al-Debagy & Martinek, 2018). Although it offers many advantages over the monolithic model, it also has limitations due to the complexity introduced by dividing business logic among multiple independent and asynchronous services and the complexity brought by microservices makes development, and operation more difficult (Soldani et al., 2018).

Monolithic backend applications, in contrast to the previously discussed N-tier application architectures, use a 1, 2 or 3-tier application architecture, which means that the backend application's business logic is contained in a single unit. CaaS, IaaS, and PaaS solutions make it simple to host these types of applications in the cloud (Amazon Web Services, 2022g). Microservice backends are made up of several services, each of which handles its own business logic. AWS offers container services like Amazon Elastic Container Service (Amazon ECS) and serverless services like Lambda to provide processing power for microservices applications, as well as a slew of other services to help with networking, messaging, storage, and logging. (Amazon Web Services, 2019; IBM, 2011)

4 Identifying project factors

This chapter continues answering to the first research question but focuses on identifying the relevant factors related to web and mobile application projects from a technical standpoint. A conceptual model is constructed to determine relevant project factors for web and mobile applications. The proposed model is heavily based on a conceptual model for web-based applications (WBA) presented in research by Nabil et al. (2011). The proposed conceptual model extends the model presented by Nabil to cover and reflect the views of the case company's key stakeholders and quality factors present in web services. This chapter presents the Nabil concept model, a literature review of the quality factors of web services, and the proposed conceptual model.

4.1 Supportive research

This subsection includes the supportive research to develop a conceptual model for identifying key quality aspects and factors over the lifecycle of web and mobile applications, focusing on cloud services.

4.1.1 WBA Conceptual model

In a study by Nabil et al. (2011) a conceptual model is proposed for WBA based on the six-quality characteristics of ISO 9126. The presented conceptual model attempts to identify, categorize, and model WBA quality factors. WBA-related quality factors and sub-factors can be systematically determined using the model. The model is built according to the views and use cases of the WBA. It focuses on three distinct levels that constitute the model: 1) identification of WBA quality views and usages through identification of relevant stakeholders and their views and concerns about the WBA, 2) classification of quality factors for each stakeholder, and 3) mapping of sub-factors for each quality factor. (Nabil et al., 2011, 214)

1) *Identifying quality views and usages.* Identifying quality views and usages is achieved through understanding the purpose of WBA and the stakeholders involved in designing, building, and using the WBA. The quality views can be identified by understanding stakeholder concerns and views on the WBA. The three main stakeholders identified in the model are developers, owners, and visitors. The developers and owners can be seen as the internal stakeholders, and visitors as the external stakeholders. WBA quality assessment is often based on internal stakeholders, but the needs of external stakeholders must also be considered. (Nabil et al., 2011, 214)

2) *Categorizing quality factors*. The International Standard ISO 9126 defines the six quality characteristics for software quality: functionality, reliability, usability, efficiency, maintainability, and portability (ISO/IEC, 2000). The classification of quality factors is done according to the concerns and needs of stakeholders and based on the quality characteristics of ISO 9126. For example, developers are concerned about factors such as maintainability and portability, whereas visitors are more interested in usability. (Nabil et al., 2011, 212-215)

3) *Mapping sub factors.* The last step in the model is mapping the sub factors to each quality factor based on the concerns of the identified stakeholders. The sub factors presented in ISO 9126 are extended by new WBA quality subfactors because according to Nabil, the definition of ISO-9126 is limited as it lacks specification of key factors for WBA and differences in objectives and quality views compared to traditional software applications. (Nabil et al., 2011, 212-216)

4.1.2 Web services

A standard for quality factors of web services was presented in a paper published by OASIS, the Organization for the Advancement of Structured Information Standards, in 2012. Web services are services that are accessible through an access point in the web and operate over HTTP (Kuuskeri, 2014, 5), for example back-end services that provide REST or GraphQL interfaces for the client to use. The standard includes quality factors and sub factors for the development, usage, and management of web services. Both OASIS and Nabil recognize that web-based applications and services have distinguished characteristics compared to traditional installation-based software which is why they require their own quality factors. (OASIS Open, 2012)

The figure below (**Figure 9**) presents the main characteristics, quality characteristics and quality factors of web services as according to OASIS. Characteristics of the web services are derived into quality characteristics that are derived into separate quality factors. OASIS lists the following as the main characteristics of web services: Different users and providers, network-based, loosely-coupled and various binding, platform independent, and standard-protocol based (OASIS Open, 2012). They can be further divided into the following quality characteristics:

1) *High sensitivity of quality*. Web services have a high sensitivity of quality. This is especially emphasized regarding the performance and business. Consumers using the service are highly interested in the quality of the service and will change the service if it does not fulfil their performance or business requirements. (OASIS Open, 2012)

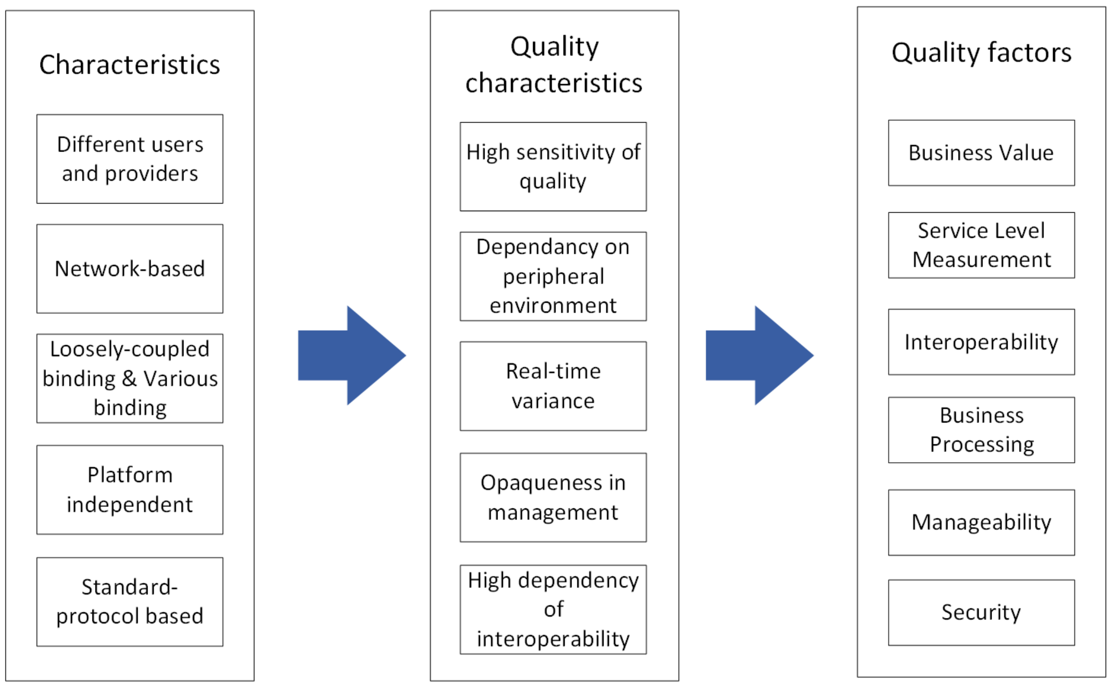


Figure 9. Characteristics of web services (OASIS Open, 2012)

2) *Dependency on peripheral environment.* Web services are operated closely within other systems. Hence the quality of web services is influenced by the technical environment of those systems as well such as the network system and resource system. For example, the service provider’s processing capability or a narrow bandwidth of the transport network affect the quality of the web service regardless of how well it has been implemented. (OASIS Open, 2012)

3) *Real-time variance.* Real-time variance suggests that the consumer’s customer and service provider’s service are loosely and variably tied. There is variation in the quality of the web service because the consumer can change the service in real time. (OASIS Open, 2012)

4) *Opaqueness in management.* The consumer often has limited or no access for controlling and managing the web service, thus having to rely for the service provider to deliver agreed level of quality. For this reason, consumers often require a prominent level of quality in the web service. (OASIS Open, 2012)

5) *High dependency of interoperability.* Ensuring high interoperability in web services requires effort because web service clients and servers can be deployed in several ways and on different platforms, and web services often have several distinct clients who consume them. With several clients there is also a risk that the standards will be misunderstood. (OASIS Open, 2012)

Finally, the described quality characteristics can be derived into the following quality factors and sub factors as seen on the figure below (**Figure 10**):

1) *Business Value*. The economic value that a web service delivers to a company's business is referred to as its business value. It depends on subfactors such as the price, the policy of penalties and incentives, the business performance, the recognition of the service and the reputation of the service and the service provider. The business value quality factors provide a business perspective that the consumers interested in the monetary value and trust of the web services can refer to when choosing web services. (OASIS Open, 2012)

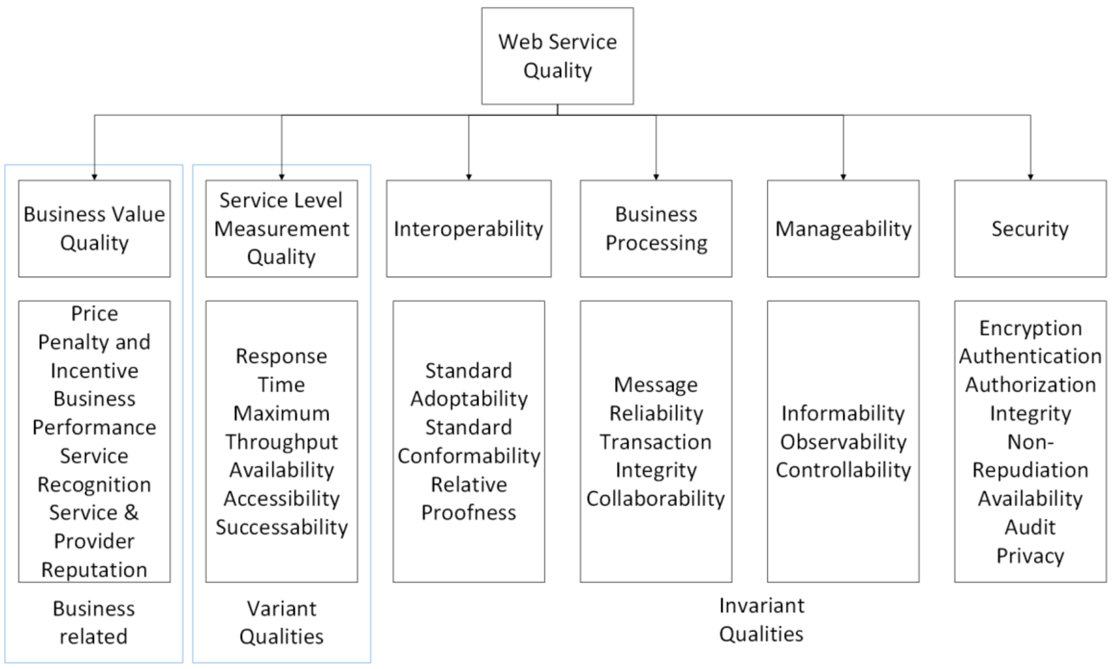


Figure 10. Web service quality factors (OASIS Open, 2012)

2) *Service Level Measurement*. The service level measurement sets several quantitative attributes for measuring the performance of a web service. It depends on subfactors such as response time, maximum throughput, availability, accessibility and successability. These qualities are variant qualities, which means that they can change, intentionally or not, in real-time during the use of the web service. (OASIS Open, 2012)

3) *Interoperability*. Interoperable means that two independent systems may share and use data as if they were running on the same platform. To ensure interoperability for web services OSASIS identifies three different sub factors:

   a. Standard Adoptability refers to the number of functions implemented by the web service, where the related standards have been adopted.

   b. Standard Conformability refers to the number of functions in which the web service fully conforms all the adopted related standards.

   c. Relative Proofness indicates whether the consumer and service can successfully exchange and use information. (OASIS Open, 2012)

4) *Business Processing*. Business Processing quality ensures that the service units operate in a reliable and stabile way. It includes sub factors such as messaging reliability, transaction integrity and collaborability, and seeks to ensure correctness and reliability in business processing units. (OASIS Open, 2012)

5) *Manageability*. Manageability quality factor refers to how manageable the web service and its resources are. It is achieved by implementation of manageability capabilities which can be categorized into sub factors: informability, observability and controllability. The consumers access the manageability capabilities through an exposed access point, where they can manage the web service's software and hardware components. (OASIS Open, 2012)

6) *Security*. Security quality factor refers to the safety of web services. Web services are available over the internet, so they have a wide attack surface area. Typical threats include unauthorized access and destruction of services. It includes subfactors such as encryption, authentication, and privacy. (OASIS Open, 2012)

4.1.3 Software Development Life Cycle (SDLC)

When evaluating cloud or web services for a software project, the full software lifecycle must be examined in addition to the presented WBA and web service quality factors offered by Nabil and OASIS Open. Furthermore, the development approach should be adjusted to cloud-based projects (Alshazly et al., 2021). The software development life cycle (SDLC) encompasses the full software development process, from concept to operation. When deciding whether to use specific cloud computing services, SDLC activities such as requirements gathering, design, implementation, deployment, maintenance, and retirement should be addressed. The flaws in requirements gathering and designing the software system spread to other tasks and become more expensive to correct in later stages. (Mahmood & Saeed, 2013, 12-30)

Alshazly et al. (2021) suggest a cloud software life cycle process (CSLCP) model to help small and medium-sized enterprises (SMEs) analyze the benefits and drawbacks of cloud computing and support creating quality software (Alshazly et al., 2021). The process model consists of nine phases, which of each have eight different process areas. The phases and process areas involved in this study are briefly presented:

1) *Exploration*. This includes defining high-level project and business objectives, project scope, the nature and boundaries of the software, and the initial requirements. Decisions on the employed deployment model and prospective service models can be made based on the information obtained. Service Level Agreements (SLAs), which describe the agreed-upon level of aspects of services (availability, quality, and so on), are investigated. (Alshazly et al., 2021)
2) *Alternatives and the decision-making*. Identification of development alternatives, such as usage of the service model, whether to utilize ready components (SaaS and MBaaS), build everything from scratch (on-premises or IaaS) or use service models that offer abstraction and managed services to speed up development (FaaS, PaaS, CaaS). (Alshazly et al., 2021)
3) *Planning*. Planning of the schedule, budget, and scope for the development. Includes estimating resources needed, budget, skills, and size of the development team. Requirements are prioritized, and development artifacts are identified. (Alshazly et al., 2021)
4) *Analysis*. Further examination of the specified functional and non-functional requirements, as well as a high-level architecture study to identify required software components. (Alshazly et al., 2021)
5) *Design*. Defining the interface and integration needs. A thorough architecture design is built based on the high-level architecture study, illustrating interactions between software components and their interfaces. The supported designs and programming languages of the chosen cloud service provider should be used in the design architecture. (Alshazly et al., 2021)
6) *Implementation*. The required services are obtained, SLAs are agreed upon, and development of the software artifacts begins. Quality of Service (QoS) is monitored

to determine whether to switch providers or stay with them. Component and integration testing is carried out. (Alshazly et al., 2021)

7) *Deployment*. When the software artifacts are deployed, the QoS terms and SLAs are reviewed. Interoperability, usability, and performance testing are carried out. End-user feedback is analyzed to improve the software. (Alshazly et al., 2021)

The model proposed by Alshazly et al. (2021) is a cyclic iterative prototyping model (Alshazly et al., 2021), whereas the case company's primary software development methodology is Agile. Both of which, by definition, are iterative (Alshazly et al., 2021; Atlassian, 2019). The CLSCP is iterative in nature, allowing for ongoing examination of work products and enhancement of software, due to found defects and/or changing requirements (Alshazly et al., 2021). In prototyping models, the prototype is produced, tested, and improved until it is accepted, after which the final software can be built based on it (Pomberge et al., 1998). Agile on the other hand is an iterative approach for software development and management where the work is done and delivered to the customer in small increments (Atlassian, 2019). Furthermore, in agile a Minimum Viable Product (MVP) is created and improved over multiple cycles as opposed to the prototyping model, where the prototype is developed separate from final product (Pomberge et al., 1998). When using CSLCP in an Agile framework, development may begin with the creation of the MVP rather than the proposed creation of the prototype.

4.2 Proposed model

The structure of the proposed bottom-up model is seen on the figure below (**Figure 11**). The conceptual model is based on the quality factors and sub factors identified based on the conceptual model of Nabil and the paper by OASIS Open. The proposed model extends the perspective of internal stakeholders beyond developers to development team and project management to form a theoretical basis for the subsequent survey of relevant project factors for the case company's projects when choosing cloud computing services. Although this study's focus is on technical quality and the perspectives of the case company's internal

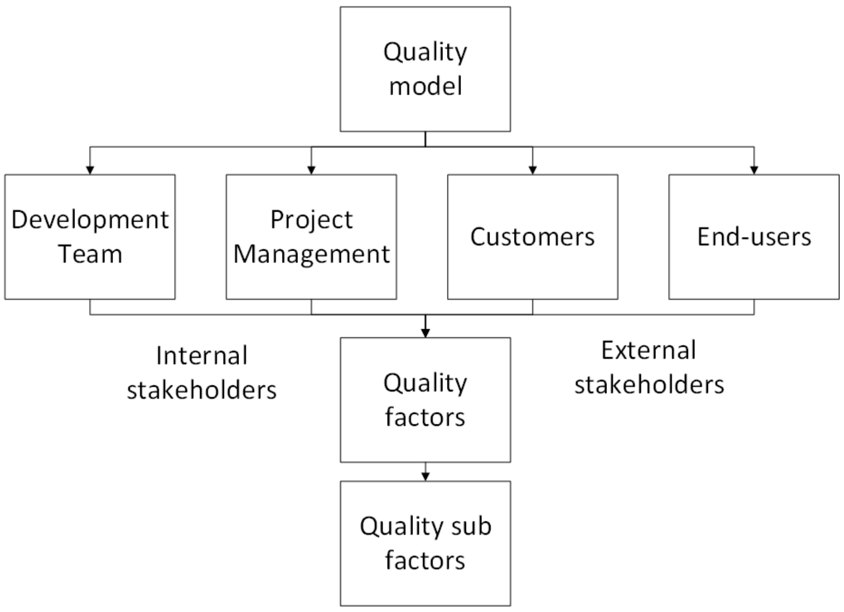


Figure 11. The extended conceptual model structure (Nabil et al., 2011, 214)

stakeholders, e.g., development team and project management, external stakeholders such as customers and end-users are considered to pro-duce an appropriate quality assessment of the web-based applications and services (Nabil et al., 2011, 214). Furthermore, the cloud SDLC phases and related factors from the process areas are incorporated into the recognized quality factors and sub-factors if deemed appropriate.

4.2.1 Identifying quality views and usages

To make an accurate quality assessment of the WBA, both internal and external criteria must be considered (Nabil et al., 2011, 214). Internal criteria come from internal stakeholders and external criteria from external stakeholders. In the case company's typical projects, the case company acts as internal stakeholders and the customer and the applications users as external stakeholders. Nabil’s view of the WBA sees developers and owners as a source of internal criteria and visitors as a source of external criteria. Given the nature of case company projects where the software is owned by the customer but developed by the case company, the stakeholder list can be modified to include development team and project management as internal stakeholders, and customers (former owners) and end users (former visitors) as external stakeholders.

End-users. End-users or visitors are the actual users of the application. According to Nabil, the users have substantial “need to include” features and requirements such as that the application should be easy to understand, easy to use and easy to find. They often also come from a variety of backgrounds, meaning they have a diverse background in terms of knowledge, expertise, and needs. (Nabil et al, 2011, 214)

Customers. Customers or owners are the party that owns rights to the application. Nabil recognizes that for the customers the applications intend to communicate an image of the organization, and support access to information and knowledge to the end-users in addition to the actual service being provided (Nabil et al, 2011, 214-215). Consumers frequently purchase an application's development and maintenance from a specialist organization to deliver the application to their own customers i.e., end-users. When purchasing software for internal use, the customer may also be the product's end-user.

Development team. The case company’s projects primarily follow the agile project methodology. Development teams in agile methodology are cross-functional and self-organizing. Cross-functional teams consist of all the necessary expertise in one team instead of having separate teams of, for example, developers, designers, and testers. Self-organizing teams do not have a separate project manager who assigns tasks to the development team, but the team assigns tasks to the project members, and takes responsibility for each activity and decision as a team. A typical agile development team in the case company consists of software developers, UX-designers, and testers. (Crowder & Friess, 2015, 19-35)

1) *Software developers*. Software developers design, build, and maintain software systems. Developers are involved in the entire software development lifecycle, starting from the process of transforming requirements to features, architecture design and product development and eventually software maintenance. The development process involves steps such as designing and building software architecture, managing configuration and quality assurance, deploying the product artefacts and software maintenance. (IBM, 2019b)

2) *UX-designers*. UX designers strive to create an excellent user experience for application users. They are concerned with the branding, design, usability, and function of the product. As software developers build applications, UX designers design what the interface and features should look and feel like. (Interaction Design Foundation, 2019)

3) *Testers*. Software testers try to find bugs and make sure that the software system or application works as intended and meets the requirements. Testing can reveal problems such as architectural flaws, scalability issues, and security vulnerabilities in the software system. (IBM, 2019a)

Project management. In the case company, at least one project manager is assigned to each project. According to Banerjee, project managers are concerned about the project's scope, cost, schedule, and overall quality (Banerjee, 2016, 3). They manage budget and risks, prepare project plans, communicate with the stakeholders, and ensure project progress by resolving issues and making sure that the development team is adequately resourced (Banerjee, 2016, 2-4). Furthermore, when it comes to choosing cloud computing services, they must be able to assess them throughout the software lifecycle in partnership with the development team to make suitable, long-term, and cost-effective judgments about cloud services in software lifecycle planning. In case company's projects that follow the Agile methodology, especially those that use Scrum, project managers often work in the role of Scrum Master, which expands the role of the project manager in establishing and managing the Scrum process (Banerjee, 2016, 2-4, Crowder & Friess, 2015, 19-35).

4.2.2 Categorizing quality factors and mapping sub factors

Using the six quality characteristics from the ISO 9126 standard, Nabil's conceptual model categorizes each of the quality factors for the identified stakeholders' quality views (Nabil et al., 2011, 215). In this study, Nabil's categorized quality factors from the ISO 9126 standard are used for end-users, customers, and partly for the development team, whereas factors identified from the OASIS Open paper are emphasized for the development team and project management as the identified factors from the paper are technology-oriented and their target audience are architects, developers, and quality managers (OASIS Open, 2012). Furthermore, because the study's focus is on technical quality from the perspectives of the development team and project management, both of which are emphasized in this research, end-users and customers receive less attention, even though they are included in the research because their concerns drive project requirements and thus push the development team and management to make decisions on cloud services the overall quality.

End-users. Visitors' concerns are classified as domain-independent or domain-dependent quality factors by Nabil et al. (2011). Domain-independent quality factors are consistent across all WBA domains, but domain-dependent quality factors differ from domain to domain (Nabil et al., 2011, 215). Usability, accessibility, content, and credibility are domain-independent quality factors, while functionality, security, and internationality are domain-dependent quality factors (Nabil et al., 2011, 215). The stated factors mostly address whether the WBA provides users with the necessary features, is useable and accessible, and is secure to use. When it comes to cloud computing services, none of the factors can be directly related, albeit security may be claimed as one because the public cloud faces security risks owing to its public nature (Mell et al., 2011, 3). The table below shows the discovered quality factors and sub factors from Nabil's model and the OASIS Open document that are related to cloud computing services from the viewpoint of the end-user, as seen on the table below (**Table 1**).

Table 1. End-user quality factors and sub factors (Nabil et al., 2011, 216; OASIS Open, 2012)

Quality factor	Quality sub factor	Description
Accessibility	Application speed / Response Time	Refers to how quickly the application responds to user inputs and how quickly the application is downloaded for usage by the user.
Content	Accuracy	Information presented by the application is correct.
Security	Authentication	The application is protected from unauthorized access.
Security	Authorization	Users have varying rights to the application resources.
Security	Integrity	Unauthorized alteration of resources and applications is prevented.
Security	Availability	Authorized consumers can access the application when needed.
Security	Privacy	The protection of sensitive information of users and suppliers.

Customers. The customer viewpoint, according to Nabil, comprises of three quality factors: differentiation, popularity, and profitability (Nabil et al., 2011, 215). Differentiation refers to how the WBA competes with others while promoting the owner's identity; popularity is strongly linked to addressing end-user problems; and profitability simply refers to how successfully the application accomplishes its goal (Nabil et al., 2011, 215). Customers pay the case company to produce the product, therefore the OASIS Open paper's business value quality factor may also be employed by customers, since it provides a business viewpoint to which clients interested in monetary value can refer when picking web services (OASIS Open, 2012). The table below shows the identified quality factors and sub-factors for customers (**Table 2**). Aside from the factors listed on the table, the customer also shares most of the end-user-related factors (**Table 1**).

Table 2. Customer quality factors and sub factors (Nabil et al., 2011, 216; OASIS Open, 2012)

Quality factor	Quality sub factor	Description
Business Value	Price	Monetary value paid for the service to operate. Service cost affects the cost of the project.
Security	Encryption	Confidential and sensitive data are protected by using cryptographic capabilities to prevent unauthorized users from accessing them.

Development team. Nabil recognizes three quality factors for developers: portability, maintainability, and reusability (Nabil et al., 2011, 214), none of which directly can be related to choosing a computing service. In addition, since the development team oversees a software application's full development cycle (IBM, 2019), including the bulk of technical decisions linked to cloud computing services, they must be supplemented with a more technical set of quality factors. Furthermore, the development team's actions are influenced by customer requirements, and the customer's requirements are influenced by end-user needs. As a result, the development team should also be concerned about the quality factors that affect both end-users and consumers. The table above outlines the quality factors and sub-factors for the development team when it comes to choosing cloud computing services (**Table 3**). In addition to the previously mentioned quality factors and sub-factors, the follo-

Table 3. Development team quality factors and sub factors (OASIS Open, 2012)

Quality factor	Quality sub factor	Description
Service Level Measurement (SLM)	Response Time	The time between sending a request and receiving a response.
SLM	Maximum Throughput	The maximum number of requests the service can handle in a given amount of time.
SLM	Availability	Refers to the amount of time the service is up and ready to serve requests.
Manageability	Informability	The information that is available about the service and if it is sufficient to aid in its management.
Manageability	Observability	Refers to how effectively the status information for the service is communicated through monitoring and notifications.
Manageability	Controllability	Refers to the control functions available to keep the web service in a manageable state. Includes operational and configuration functions.
Interoperability	No specific sub-factors	Refers to how well the service integrates with other services.

wing factors must be considered when selecting cloud computing services for the development team, as outlined in the previous chapters:

- The required level of configuration over the service. For example, IaaS for the highest level of configuration available.
- The required level of control and governance over the cloud infrastructure. For example, a public or private cloud.
- The architecture of the server-application. Microservices or monolithic, for example.
- The required functionalities of the server-application and whether any of them are implemented using a third-party provider, such as authentication through AWS Amplify.
- The chosen technology stack. E.g., programming languages, libraries and frameworks planned to use.

- The security and performance requirements established by the customer. For example, requirements for authentication, privacy, availability, latency, and scalability.

Project management. Project management is concerned with the project's scope, cost, timeline, and overall quality, and focuses on that the project completes within set budget and time, and produces the desired outcome (Banerjee, 2016, 3). To complete a project with the expected outcomes, it is necessary to listen to the customer's goals and values, as well as keep the development team engaged, well resourced, and handle communicating changes to stakeholders. (Banerjee, 2016, 2). The table below lists the identified quality factors and sub-factors for project management (**Table 4**).

Table 4. Project management's extended quality factors and subfactors (OASIS Open, 2012)

Quality factor	Quality sub factor	Description
Business Value	Price	Monetary value paid for the service to operate. Service cost affects the cost of the project.
Business Value	Penalty	A penalty is a monetary compensation for business losses caused by non-fulfillment of a contract or failing to achieve established quality standards. For instance, a customer may specify an availability requirement, but the service provider fails to provide it.
Business Value	Service fitness	Refers to well the service fits to the requirements of the application. For example, the alignment of the service provider's roadmap for selected service versus the application lifecycle, the available SLAs and QoS.
Manageability	Controllability	Refers to the control functions available to keep the service in a manageable state. Time spent on service management is time not spent on development.
Security	No specific sub-factors	Security quality refers to the ability to protect web services from different threats such as attackers attempting to obtain unauthorized access, falsify or destroy online services.

Both the developer team and the project management inherit quality factors from customers and end-users that they must consider when choosing cloud computing services. These factors are not presented separately in Tables 3 and 4, as they are case-specific and are based on the project's needs and requirements as provided by the customers. Manageability is crucial quality factor for both the developer team and project management. When a service is hard to manage, more time is spent configuring, debugging, and maintaining it, leaving less time for project work, which in turn can lead to delays and make future maintenance work time consuming. Security is becoming increasingly important and crucial for web services (OASIS Open, 2012), particularly in public cloud services (Mell et al., 2011, 3), and should thus also be a major concern for the project stakeholders, as most of the case projects operate on the cloud. Furthermore, issues related to privacy such as GDPR, law and local regulations should all be considered when adopting cloud services (OASIS Open, 2012). The business value of a service is also important because the cost of cloud services varies because it is calculated using service-specific cost models (Jonas et al, 2019, 6-8) and there may be precise limits on the performance of the application depending on the customer's requirements and agreements, and if the used service fails to provide that, depending on the service used, the service provider can pay monetary compensation. The business value incorporates decisions related to the entire economics of the software lifecycle. In the case of cloud computing services, this means not only selecting the most cost-effective service at the time, but also one that will meet the stated needs in the future, such as one whose service roadmap supports the length of the application lifecycle.

4.2.3 Summary

The proposed model for presents the most essential quality factors to consider when selecting cloud computing services. The model includes internal stakeholders, the development team, and project management, as well as external stakeholders, customers, and end-users. For each of the stakeholders, quality factors and sub-factors are provided, with a focus on technical quality. Internal stakeholders inherit quality features from external stakeholders because they provide objectives and requirements for the project. Critical quality factors for the development team and project management are service level measurement capabilities, manageability, interoperability, security, and manageability.

5 A survey of cloud computing quality factors and server needs

This chapter provides an answer to the research question: *RQ:2 What are the most important quality-related technical factors and needs when choosing cloud computing services for the case company's typical web and mobile projects according to the case company's developers and management?* by outlining the concepts and theory behind creating a high-quality survey, as well as building one for the thesis, and presenting the results. The previous chapters establish the theoretical foundation for the survey's content, such as the knowledge base for cloud computing, web and mobile applications, and the relevant quality factors and other aspects coming from the cloud software development lifecycle.

5.1 Literature supporting the survey preparation

A survey is a research method that can be used to characterize and compare knowledge. (Kitchenham & Pfleeger, 2002d, 16). A survey collects data from a group of people by asking them questions (Check & Schutt, 2012, 160), and it can be used to perform both qualitative and quantitative research (Ponto, 2015). Kitchenham and Pfleeger (2002d) propose 10 activities for the construction of a survey process (Kitchenham & Pfleeger, 2002d, 16), which are presented next and replicated for building a survey for the thesis.

1) Setting specific, measurable objectives. The first step of the survey process is setting the objectives. The objectives dictate what questions will be asked, what demographic will be surveyed, and what information will be gathered. The objectives are often rephrased as research questions or derived from the research questions. The survey's desired outcomes are described in each objective. (Kitchenham & Pfleeger, 2002d, 17)

2) Planning and scheduling the survey & 3) Ensuring that appropriate resources are available. Kitchenham and Pfleeger (2002d) mention the processes for planning and scheduling the survey, as well as ensuring that adequate resources are available, but no precise directions are given. However, survey planning may be regarded as being carried out by following the presented survey process. On the other hand, determining the schedule and required resources is case specific to the research situation.

4) Designing the survey. The survey design must relate to the objectives for the survey data and analysis to answer the questions that are given (Kitchenham & Pfleeger, 2002e, 18). When selecting the correct survey design, it is also important to assess if the design provides for the most effective ways of getting the information required to fulfill the objectives, effective meaning that it’s resilient to bias, appropriate, and cost-effective (Kitchenham & Pfleeger, 2002e, 18). In addition, choosing the target population, and constructing the survey in such a way that the highest possible response rate is established must be considered (Kitchenham & Pfleeger, 2002e, 18).

5) Preparing the data collection instrument. The construction of a questionnaire consists of searching the relevant literature, constructing, valuating, and documenting the instrument (Kitchenham & Pfleeger, 2002b, 20). Kitchenham and Pfleeger make several suggestions for question building, including understanding the respondents, asking a suitable number of questions, standardizing response styles, purposeful and concrete questions, question categories, and the questionnaire structure. (Kitchenham & Pfleeger, 2002b, 21-23). In addition, topics such as motivating people to answer the questionnaire and how to avoid research bias should also be considered when constructing the questionnaire (Kitchenham & Pfleeger, 2002b, 21).

6) Validating the instrument. The purpose of instrument validation is to ensure that the questions are understandable, expected responses match the data analysis techniques and to assess the validity and reliability of the instrument. The validation of the instrument is often conducted through focus groups and pilot studies. Focus groups are formed by collecting a group of people who represent the target audience to identify any potential difficulties, while survey pilot studies are conducted in the same way as surveys, but with a lower sample size to detect problems with the questionnaire itself. (Kitchenham & Pfleeger, 2002c, 20)

7) Selecting participants & 8) Administering and scoring the instrument. The process of selecting participants begins with defining the target population, or the group to which the survey applies (Kitchenham & Pfleeger, 2002a, 17). Sampling is the process of selecting participants from a target population using probabilistic or non-probabilistic sampling procedures to get enough survey replies (Kitchenham & Pfleeger, 2002a, 18-20). Furthermore, when dealing with a small target population, it is best to try to acquire replies from the complete population instead of sampling the population into smaller samples (Kitchenham & Pfleeger, 2002, 20a). The execution and management of the data collection method, such as self-administered questionnaires or conducted interviews, is referred to as administering, and scoring the instrument refers to assigning point values to the questionnaire’s answers.

9) Analyzing the data & 10) Reporting the results. The responses should be validated for completeness and validity before in-depth analysis; Kitchenham and Pfleeger (2003) suggest having policies for inconsistent and incomplete answers to the questionnaire to address this issue. Another thing to consider before beginning data analysis is partitioning the sub-groups. The responses can be partitioned into homogeneous sub-groups based on demography, for example, to allow comparison with other sub-groups. Data coding can be used to convert both closed and open questions into numerical scores before they are entered into electronic data files; however, if the survey is administered electronically, such as via the Internet, this will not be an issue. Open questions require more effort and expertise to correctly categorize and code into numerical values. Simply put, the type of data analysis required is determined by the survey design, response scale, and sampling method used, and it usually involves data coding and a analyzing the numerical scores. After data analysis, the results from the survey can be presented. (Kitchenham & Pfleeger, 2003, 24-27)

5.2 Constructing the survey for the data collection work

The survey is used as a research method in this study to explore which of the earlier identified quality factors and technical requirements are the most important for the case company’s employees and what are the most typical computational requirements and characteristics for case company’s web and mobile projects. Questionnaires and interviews are the most used data collection instruments in survey research, and self-administered Internet-based questionnaires were chosen as the data collection instrument in this study because they are less time consuming than interviews and are more practical for large samples of respondents (Ponto, 2015). Following the steps mentioned by Kitchenham and Pfleeger the survey is constructed:

1) Setting specific, measurable objectives. The survey's objectives in this study are to 1) identify which of the previously identified quality-related technical factors are most important, 2) recognize the most common computational requirements and characteristics, and 3) fill any information gaps through open questions about the objectives 1 and 2 from the perspective of the case company's software developers and project management in web and mobile projects. The answers to objectives 1 and 2 are quantitative and directly measurable, whereas the answers to the open-ended questions in objective 3 are qualitative and require additional work to quantify.

2) Planning and scheduling the survey. The theoretical background for the survey's objectives is established through literature reviews and the development of a conceptual model, as discussed in the preceding chapters. Following the establishment of the theoretical background, survey design and other associated activities are carried out. The survey will take place over the course of two weeks in May 2022 as agreed on with the case company.

3) Ensuring that appropriate resources are available. The survey's resources are limited because it is conducted by a single individual during a short period of time. As a result, the data collection instrument and questions should be developed in such a way that the survey may be completed within the time range specified.

4) Designing the survey. Given the survey's objectives, a descriptive design known as case control is chosen as the design type in this study. A case control design is retrospective in nature, and it seeks to explain current occurrences by looking back at previous events

(Kitchenham & Pfleeger, 2002e, 18), in this study the previous events refer to the previous web and mobile projects. The target population of the survey are the developers and project managers that have partaken in web and mobile projects in the case company. Since the data collection instrument chosen for the survey is a self-administered Internet-based questionnaire, it is easy to reach a large sample of responders (Ponto, 2015) and since the target population at the case company is not particularly large and it is well defined, the questionnaire is carried to the complete target population.

5) Preparing the data collection instrument. The data collection instrument chosen is questionnaire. In 2020, a survey for a master's thesis was conducted in the case company regarding testing in agile projects, where questionnaire was conducted through Google Forms and links to the questionnaire distributed through emails and Slack (Hartikainen, 2020). Because Google Forms was previously utilized at the case company, the questionnaire in this study is likewise created in Google Forms, and similarly the links to the questionnaire is distributed through emails and Slack. Based on the remarks presented by Kitchenham and Pfleeger, the questionnaire's questions and format is built (see Appendix 1).

6) Validating the instrument. Because the survey's target audience is project managers and developers, the validity of the questionnaire is tested in this study through a pilot study with one project manager and one developer from the target population. In addition, while constructing the questionnaire, feedback from a member of the case company is gathered. The design of the questionnaire can be altered in response to comments and ideas to make it easier to comprehend and complete.

7) Selecting participants & 8) Administering and scoring the instrument. In this study, the target population is all developers and project managers that have partaken in web and mobile projects in the case, and since the target population is well-defined and small, the questionnaire is sent to the whole target population through internal email lists and a Slack channel. The data is collected with the selected questionnaire tool, Google Forms. Google

Forms provides several options for exporting the results of the survey, which can then be used for more in-depth analysis and scoring of the responses.

9) Analyzing the data & 10) Reporting the results. The questionnaire's design can be seen on Appendix 1. As mentioned, the questionnaire is conducted using Google Forms, and Google Forms supports exporting responses in CSV (comma-separated values) format from which the responses can be further analyzed and visualized using software such as Excel. The questionnaire consists of both closed and open questions and produces ordinal and nominal data. Hence, data analysis needs to be conducted to take full advantage of the data at hand. The survey results are reported in the next subchapter, followed by findings and discussions based on further analysis.

5.3 Survey results

The survey was conducted between 06.05.2022 and 20.05.2022 and received a total of 24 responses from the employees of the case company. The questionnaire was sent out via the company's official Slack channel as well as an email distribution list. There are three sections to the questionnaire: 1) background section, which identifies the responders' background and experience with web and mobile projects; 2) back-end section, which determines the case company's back-end needs and requirements; and 3) cloud service section, which establishes the background of cloud computing services currently used by the case company. The survey's findings are briefly summarized next. The subsections that follow give a more in-depth data analysis and explanation of the findings.

5.3.1 Survey Part 1: Background

The answers for the first question: "What is your role in the organization" are shown in the table below (**Table 5**). Software developers and architects were the most common respondents, accounting for 79 percent of all replies. In addition to the 14 software developers and five architects, three project managers, one service designer (UI / UX), and one salesperson responded to the survey. The bulk of the case company's staff are IT consultants, such as software architects and developers, therefore the distribution of responders' backgrounds is not surprising. In addition, perspectives gathered from other departments can benefit the study and provide fresh perspectives on the subject.

Table 5. Question 1: Role in the organization

Role in organization	n	~% of total
Software Developer	14	58%
Architect / Consultant	5	21%
Project manager	3	13%
UI / UX	1	4%
Sales	1	4%

The answers for the second question regarding total work experience are presented in the table below (**Table 6**). 42 percent of respondents have worked in the industry for more than ten years, while 50 percent have worked in the industry for less than five years. Therefore, it can be concluded that the responses emphasize both the skills acquired through work experience and the fresh perspective of less experienced experts, as the respondents are a mix of junior and senior IT consultants.

Table 6. Question 2: Work experience

Work experience (years)	n	~% of total
0	0	0%
Less than 2	5	21%
2-5	7	29%
5-10	2	8%
More than 10	10	42%

The answers for the third question regarding work experience in web and mobile projects is presented in the table below (**Table 7**). 63% of those responded to the survey have worked on web and mobile projects for less than 5 years, while 38% have worked on them for more than 5 years. The majority of respondents (80%) have experience working on web and mobile projects for more than two years. Similar to the previous question and table 6 (**Table 6**), respondents consist of both junior and senior experts, with a majority of respondents having 2-10 years of experience.

Table 7. Question 3: Work experience in web and mobile projects

Work experience in web and mobile projects (years)	n	~% of total
0	0	0%
Less than 2	5	21%
2-5	10	42%
5-10	4	17%
More than 10	5	21%

The fourth question inquiries about the number of web and mobile projects on which respondents have worked on. The responses are presented in the table below (**Table 8**). Most respondents (67%) have participated in 0-10 web and mobile projects, with 38% having only less than five projects and 29% between five and ten projects. 17% have worked on 10 to 25 projects and 17% on more than 25 projects. The goal of the question is to set the stage for the next sections 2 and 3, and their project related questions.

Table 8. Question 4: Number of web and mobile projects participated in

How many web and mobile projects have you been involved in?	n	~% of total
0	0	0%
0-5	9	38%
5-10	7	29%
10-25	4	17%
+25	4	17%

The fifth and last question of this section discusses the respondents' primary roles in web and mobile projects and is a multiple-choice question. The answers are presented in the table below (**Table 9**). The most frequent roles for the respondents are front-end, back-end, and architect / consultant. Front-end is recognized by 63 percent of respondents as one of their main roles in web and mobile projects, followed by back-end by 54 percent, architect/consultant by 29 percent, project management by 21 percent, service design (UI/UX) by 4 percent, and sales by 4 percent. The highly development-oriented background of the respondents supplements the objectives of the study.

Table 9. Question 5: Main roles in web and mobile projects

Main roles in web- and mobile projects	n	~% of total respondents
Front-end	15	63%
Back-end	13	54%
Architect / Consultant	7	29%
Project Management	5	21%
UI / UX	1	4%
Sales	1	4%

The questionnaire elicited answers from 24 experts of the case company. According to the responses, the respondents include both very experienced and moderately experienced experts with a focus on front-end and back-end development, as well as software architecture. The responses in this section indicate that the respondents have adequate experience, have worked on a considerable number of web and mobile projects, and are experienced and knowledgeable enough about the survey's subject and the case company's projects to answer and provide value to the research.

5.3.2 Survey Part 2: Back-end

The answers for the sixth question, and first question of the back-end section, related to the usage of container technologies in the case company’s projects are presented in the table below (**Table 10**). The question has a response scale of 1 (never) to 5 (almost always). The majority of respondents (75%) chose 3, 4, or 5, indicating that containers are used more frequently than not. Only 4% of respondents have never utilized containers in their projects, whereas 21% use them virtually every time.

Table 10. Question 6: Usage of container technologies in the case company’s projects

Containers have been used in projects I have been involved in (e.g., Docker)	n	~% of total
1 (never)	1	4%
2 (rarely)	5	21%
3 (sometimes)	7	29%
4 (often)	6	25%
5 (almost always)	5	21%

The seventh question addresses the usage of microservices architecture in the case company’s projects, and its answers are presented in the table below (**Table 11**). The question uses the same 1 to 5 answer scale as the previous question. The majority of respondents (63%) selected 1 or 2, showing that the microservices concept is not commonly used in the case company’s projects. This isn't surprising, given that most small to medium-sized web and mobile projects don't require a complicated service design for the server-side like microservices, and rather opt-in for simpler solutions such as monolithic architecture design.

Table 11. Question 7: Usage of microservices architecture in the case company’s projects

Microservices architecture has been used in projects I have been involved in	n	~% of total
1 (never)	4	17%
2 (rarely)	11	46%
3 (sometimes)	3	21%
4 (often)	4	17%
5 (almost always)	0	0%

The answers for the eighth question regarding quality factors related to back-end development are presented in the following table (**Table 12**). The response scale has five levels, ranging from "very important" to "not at all important”, and the factors included in the question are: Low response time, support for a large number of concurrent users, scalability, and the scalability of the application environment. All of the above factors are regarded as fairly important, according to the respondents. All of the factors are seen as about equal in importance, with configurability of the service being somewhat more significant than the others and scalability being slightly less so.

Table 12. Question 8: Important back-end quality factors in the case company's projects

Factors	Very important	Fairly important	Neutral	Not important	Not at all important
Low response time	4% (1)	50% (12)	29% (7)	13% (3)	4% (1)
Support for a large number of concurrent users	4% (1)	42% (10)	46% (11)	4% (1)	4% (1)
Scalability	4% (1)	46% (11)	29% (7)	13% (3)	8% (2)
Configurability of the application environment	8% (2)	42% (10)	33% (8)	13% (3)	4% (1)

The final question for the section, Question 9, "Have there been any other important factors in your projects in terms of server solutions?", is a follow-up open question to Question 8 to fill any information gaps related to the important quality factors for server solutions. The question elicited 11 responses, which are examined in depth in the next sub-chapter.

5.3.3 Survey Part 3: Cloud services

The answers for the tenth question, and first question of the cloud services section, are presented in the table below (**Table 13**). The question addresses the types of cloud computing services used in the case company's projects. According to the findings, PaaS is the most underutilized service, while IaaS and CaaS are the most commonly used services. FaaS is used less frequently than IaaS and CaaS, but more frequently than PaaS.

The eleventh and last question in this section covers through the most significant factors to consider while selecting cloud services. Security, manageability over the service and its resources, configurability, compatibility, and performance are among the factors cited in the

Table 13. Question 10: Types of cloud computing services used in the projects

Types of computing services	Almost always	Often	Sometimes	Rarely	Never
IaaS	8% (2)	29% (7)	29% (7)	25% (6)	9% (2)
CaaS	0	38% (9)	33% (8)	17% (4)	13% (3)
FaaS	0	21% (5)	38% (9)	13% (3)	30% (7)
PaaS	0	8% (2)	13% (3)	33% (8)	46% (11)

question. The question is a multiple-choice one, with responders instructed to pick three of the most critical factors to them. The results are shown in the table below (**Table 14**). According to the results, the two most critical factors for the respondents are security and manageability of the service and its resources, with both receiving over 80% of responses. Configurability is important to 50 percent of responders, performance to 39 percent, and compatibility to 35 percent.

Table 14. Question 11: Important cloud service quality factors in the projects

Factors	n	~% of total respondents
Manageability over the service and its resources	21	88%
Security	19	83%
Configurability	12	50%
Performance	9	39%
Compatibility	8	35%

The final question of the survey, Question 12, “Are there any other important factors that come to mind when choosing a cloud service?”, is a follow-up open question to Question 11 to fill any information gaps related to the important cloud service quality factors. The question elicited 7 responses, which are examined in depth in the next sub-chapter.

5.4 Analysis, findings, and discussions based on the results

This sub-chapter contains a more in-depth analysis and discussion of the findings. The chapter is divided into three sections to make the findings more comprehensible: 1) the current situation in the case company's projects, 2) the most important quality factors for respondents regarding server-side applications and cloud services, and 3) filling in information gaps related to sections 1) and 2) through open questions. As suggested by Kitchenham and Pfleeger (2003), data coding is applied to turn closed question replies into numerical scores, and the responses are partitioned into homogenous sub-groups based on demography to allow comparison with other sub-groups (Kitchenham & Pfleeger, 2003, 24-27).

5.4.1 Current situation in the projects

As previously stated, the responses are separated into sub-groups, as shown in the table below (**Table 15**). Respondents are sorted into four categories based on their work experience and the number of web and mobile projects they have worked on. There are eight respondents in Group 1 who are technically minded and highly experienced (33 percent of total). Two respondents belong to Group 2, which includes highly experienced but non-technically oriented employees (8 percent of total). Personnel with intermediate experience make up the largest Group 3, with 9 replies (38 percent of total). Group 4, which includes individuals with little experience, has five respondents (21 percent of total). The established sub-groups are utilized to compare the previously reported data and provide more in-depth analysis of the findings.

Table 15. Sub-groups for analysis

Group	n	~% of total
1. Technically oriented, highly experienced	8	33%
2. Not technically oriented, highly experienced	2	8%
3. Moderate experience	9	38%
4. Low experience	5	21%

The following figure (**Figure 12**) illustrates the relationship between total usage of containers and microservices as according to the total results. Containers are used far more frequently than microservices, which is logical given that microservices are based on containers. Containers should thus be utilized at least as frequently as microservices. In general, microservices are not broadly applied, whereas containers are used quite frequently.

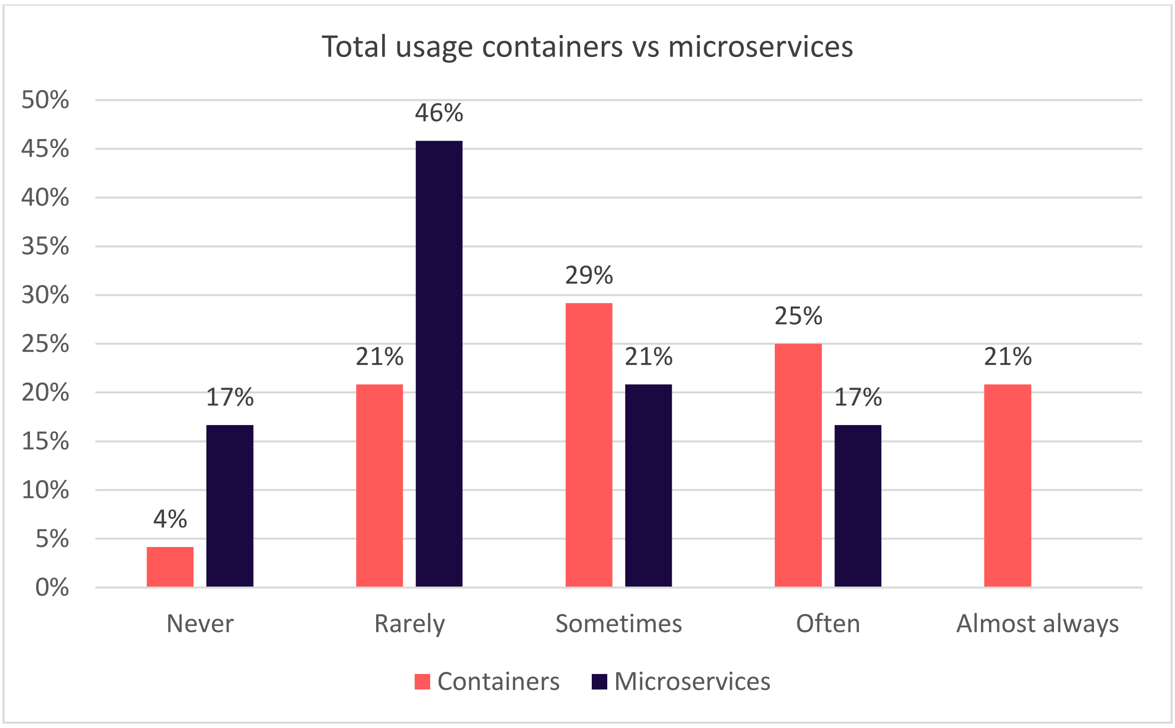


Figure 12. Relationship between the usage for containers and microservices

The following figure (**Figure 13**) shows the relationship between the usage of containers and microservices, but this time grouped by the previously formed sub-groups. There are no significant variations across the groups, except that Group 4, the less experienced responders, uses microservices and containers slightly less than the other groups. This makes sense, given that microservices are a complex concept that takes expertise and knowledge to properly implement. Furthermore, less experienced employees may not be allocated to projects with complicated design paradigms. Containers are utilized from “Sometimes” to “Often” across all subgroups, but certainly not in every project possible, and microservices from “Rarely” to “Sometimes” and is still quite an uncommon architecture paradigm in the case company’s projects.

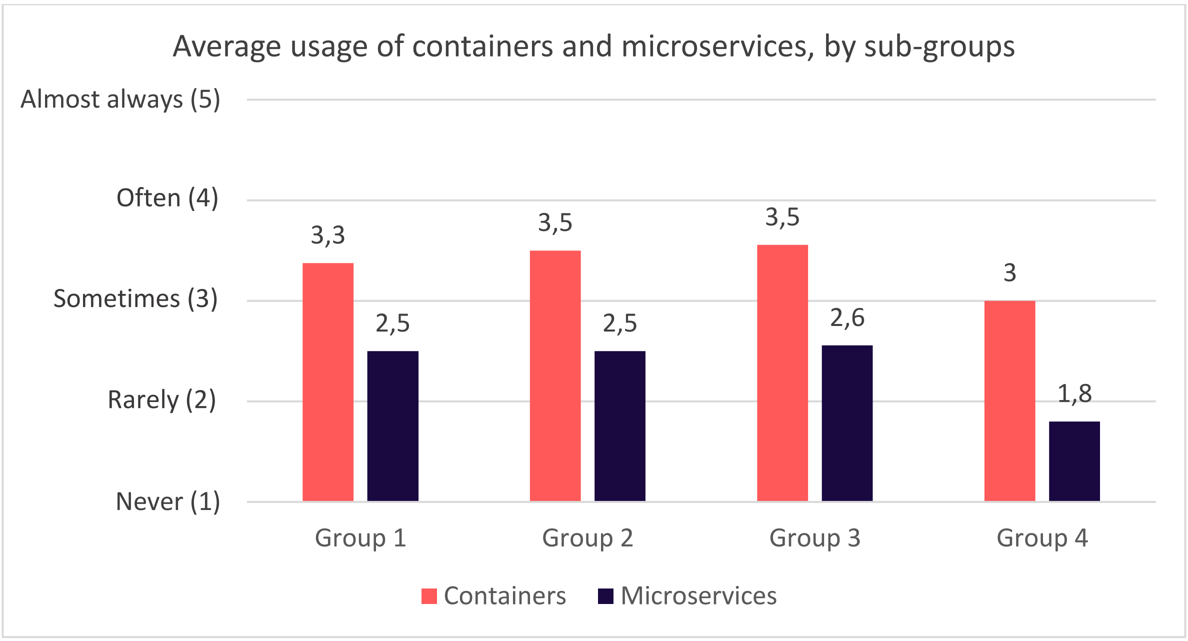


Figure 13. Usage of containers and microservices by sub-groups

The computing service models utilized are presented by sub-groups in the following figure (**Figure 14**). It is evident from the figure that most experienced responders, Group 1, the most experienced responders, prefer IaaS over the more abstracted service models, while the least experienced responders, Group 4, prefer CaaS and FaaS over IaaS. Respondents in Group 1 have the most years of experience and have so begun to use cloud services before other sub-groups. IaaS was the first service model introduced by major cloud service providers, which might explain Group 1's preference of IaaS over the other cloud service models. Groups with less experience and who are newer to the industry increasingly utilize the higher abstracted services, CaaS and FaaS over IaaS. Group 4 considers CaaS to be the most popular while having used containers the least as a sub-group (see **Figure 13**). Furthermore, one may think that less experienced professionals would find higher abstracted services, such as FaaS and PaaS, more appealing than lower abstracted services, however PaaS remains the least used even among the least experienced group, Group 4. CaaS and IaaS are undoubtedly the most extensively utilized service models in the case company among the sub-groups, followed by FaaS and PaaS. Surprisingly, this is also the order of abstraction in service models, from lowest to highest, implying that respondents prefer lower abstracted services, where developers have a higher level of configuration available, to highly abstracted services.

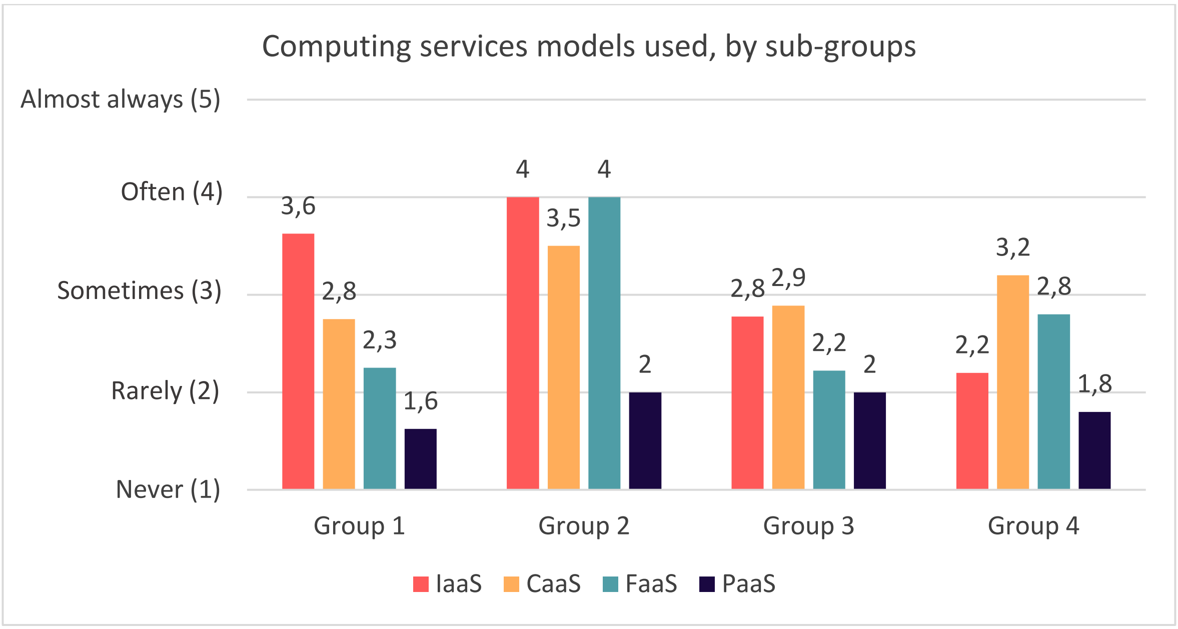


Figure 14. Cloud computing service models used by sub-groups

In comparison with the findings established about computing service models utilized by sub-groups (**Figure 14**), the next figure (**Figure 15**) depicts computing service models used by work experience. Although the most experienced sub-group, Group 1, preferred IaaS over CaaS in the previous figure (**Figure 14**), the professionals with the most work experience

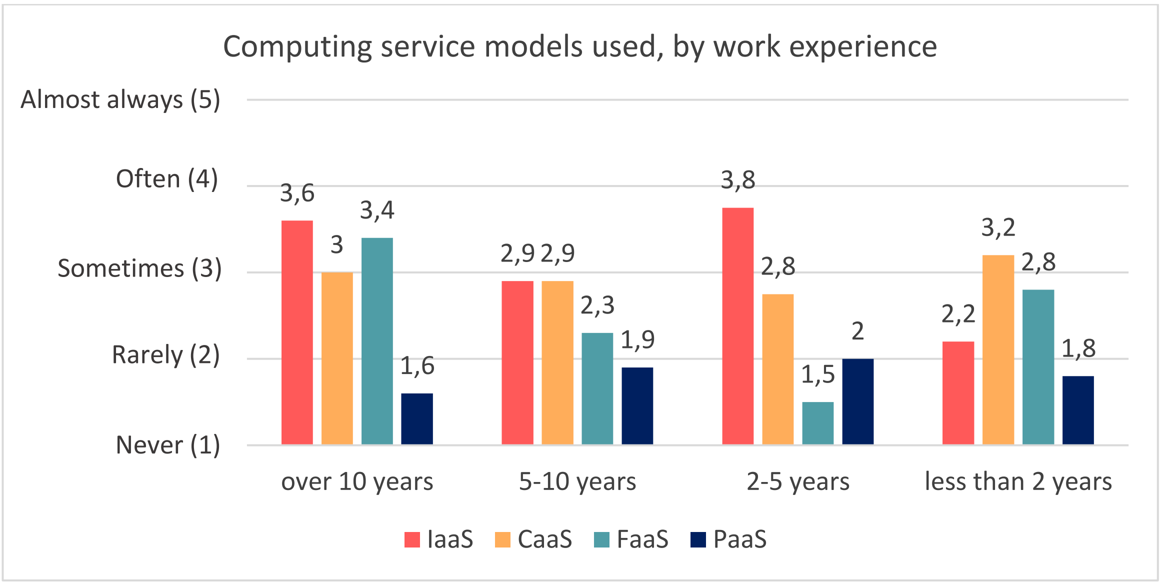


Figure 15. Computing service models used by work experience

preferred IaaS and FaaS. Professionals with the least work experience, on the other hand, tend to use PaaS and IaaS less frequently than other service models. As can be seen in the figure (**Figure 15**), as work experience increases, so does the utilization of IaaS. Perhaps the pattern will continue in the future, but for higher abstracted services such as FaaS and PaaS; that as professionals gain experience, those services will be used increasingly, and direct usage of IaaS will begin to wane. In conclusion, PaaS continues to be the least popular service across all respondents, while CaaS and IaaS are the most popular, followed by FaaS.

5.4.2 Significant quality factors

The next figure (**Figure 16**) depicts the most critical back-end quality variables by sub-groups, utilizing the same sub-groups where respondents are grouped based on their work experience and number of projects they have worked on (see **Table 15**). Group 4, which has the least experience and is newest to the industry, has the greatest averages for low response time, concurrent users, and scalability, as well as a significantly lower average for configurability than the other subgroups. This has some resemblance to the preceding figures on commonly used computer service models; as configurability isn't a top concern for this group, IaaS isn't a commonly utilized service model for them. The stated quality factors are

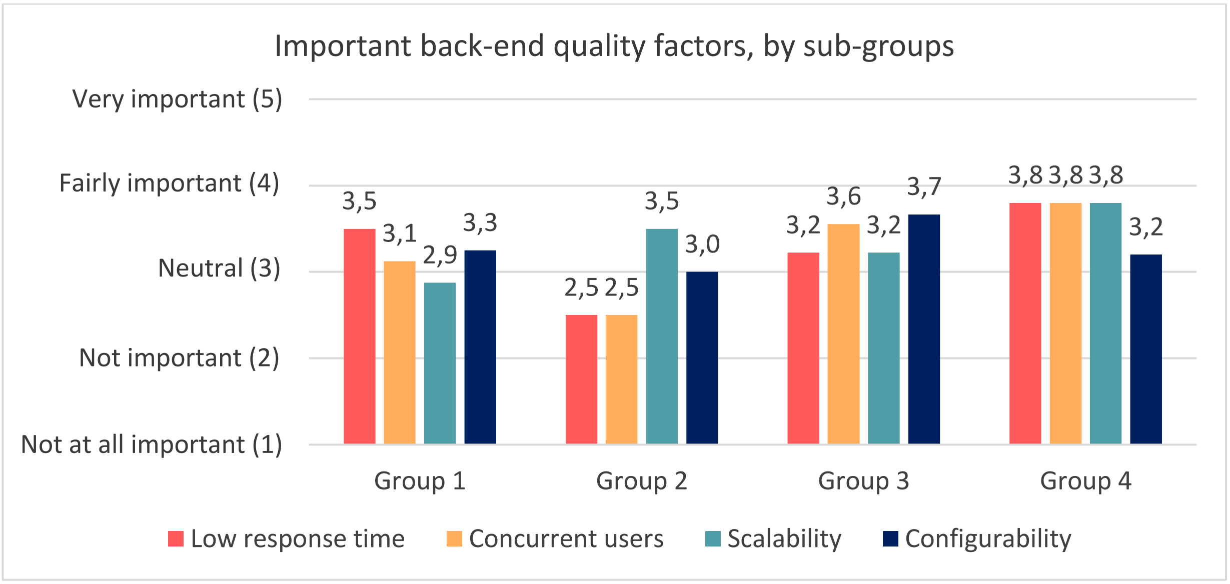


Figure 16. Important back-end quality factors by sub-groups

rated lower by more experienced experts than by less experienced experts. The stated quality factors are rated lower by more experienced experts than by less experienced experts. Each sub-groups' ranking of each quality factor appears to be quite even-handed. The importance of the listed quality factors tends to decrease as expertise grows. However, this does not imply that these issues will become less important as workers gain experience; rather, the more experience they gain, the better equipped they will be to determine when things like low response time and scalability are truly important and critical to operations, and when they are not. Estimating user numbers, scalability, and response times, for example, might be challenging for someone who has gone straight from education to employment. Choosing the correct architecture and computing services based on these criteria, therefore, necessitates purely hands-on experience with various implementations. Furthermore, the factors established by the question and their scales might imply various things to different people; for example, a low response time for some people may be in the range of 30ms, while it may be in the range of 80 ms for others.

The following figure (**Figure 17**) illustrates which cloud service quality factors the responders chose as important by sub-groups. The question is a multi-choice question, with responders allowed to pick three of the most important quality factors to them. The figure

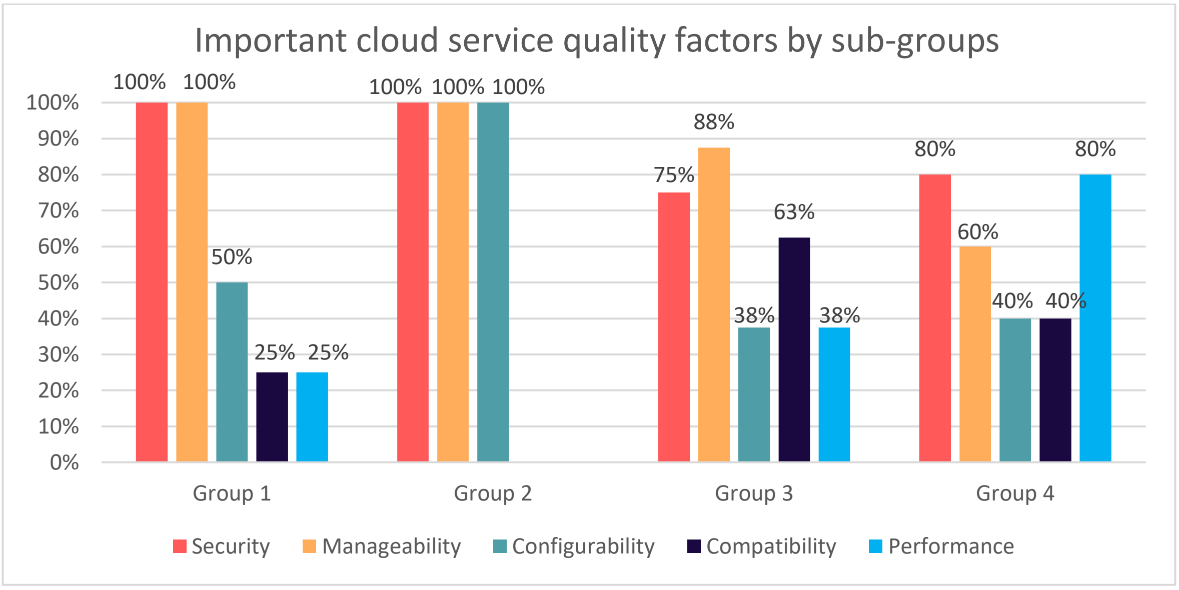


Figure 17. Important cloud service quality factors by sub-groups

presents the percent of responses each quality factor received per sub-group. In each subgroup, security and manageability have been highlighted as the most important factors. In Groups 1 and 2, 100 percent of respondents picked security and manageability as the most important quality factor. In Group 4, the only exception among the groups, performance was given precedence over security and manageability. As the third most important quality aspect, Group 1 chose configurability, Group 3 compatibility, and Group 4 manageability. Except for Group 4, performance was the least important quality criterion in general.

5.4.3 Filling information gaps

The open-question Question 9, “Have there been any other important factors in your projects in terms of server solutions?” received 11 responses from the 24 responders. As the table shows (**Table 16**), most responses are from groups 1 and 3. Four respondents indicated costs incurred as a result of selecting a server solution. The expenses of maintaining a server are based on the computing capabilities required to provide the service and are thus more closely

Table 16. Total responses for Question 9, by sub-groups

Group	n	~% of total responses
1. Technically oriented, highly experienced	4	36%
2. Not technically oriented, highly experienced	2	18%
3. Moderate experience	4	36%
4. Low experience	1	9%

linked to the cloud computing model chosen. Two respondents highlighted logging and GDPR, and two respondents also noted the techniques and technology chosen by the customer organizations’ IT departments. As previously mentioned, the case company is usually in charge of ensuring compliance with laws, rules, and GDPR, and is generally required to do so by law. When it comes to technology choices, customers often have their own preferences, and there may also be existing standards in place, such as from large organizations requiring that servers, and so on, be consistent (e.g., operating systems). One responder noted interoperability, stating that the new service must be compatible with the customers' existing solutions. Finally, for some projects, geographic location is critical; for example, healthcare solutions require services to be physically located in Finland.

The open-question Question 12, “Are there any other important factors that come to mind when choosing a cloud service?” received 7 responses from the 24 responders. As the table shows (**Table 17**), most responses are from groups 1 and 2. Four of the seven replies highlighted pricing or cost as a deciding factor in cloud service selection. The pricing was

Table 17. Total responses for question 12, by sub-groups

Group	n	~% of total responses
1. Technically oriented, highly experienced	3	43%
2. Not technically oriented, highly experienced	2	29%
3. Moderate experience	1	14%
4. Low experience	1	14%

purposefully left out of the factors since it is typically the client, not the developers, who makes decisions about cloud services based on their prices. Nevertheless, it is seen as a critical factor. The customer's technological choices, as well as GDPR compliance, are brought up again as crucial factors. As new subjects, developers, and maintainers' documentation, as well as the quality and speed of the service provider's assistance, are cited as key factors. Finally, one responder mentioned the cloud service's user interface, which allows customers to manage the service themselves. The documentation and support are more related to the choice of cloud service provider, rather than cloud services, e.g., AWS versus Azure.

5.4.4 Summary

According to the findings of the first section of the survey, the background section, the respondents have sufficient experience and have worked on a significant number of web and mobile projects, and thus are sufficiently informed about the survey's subject and the case company's projects to participate and contribute value to the study. The respondents were divided into sub-groups depending on their demographics to aid in the analysis of the following sections. Containers are widely used, however microservices are employed less frequently in the company's projects, according to the second section of the survey on back-end quality factors and requirements. The respondents rated the presented back-end quality factors as significant but not critical. The less experienced respondents, on the other hand, seemed to place a higher emphasis on the factors than the experts with greater work experience. The cloud service section of the survey found that the majority of experienced respondents favor IaaS as the computing service model, followed by CaaS and FaaS. PaaS, the fourth computer service model, received the least favorable response among all respondents. Security and manageability were clearly the most critical quality factors for all respondents when it came to cloud services and cloud computing. Configurability comes in third, followed by performance and compatibility. Finally, the most often stated issues in the open-ended questions in both back-end cloud services were cost structures, GDPR and logging, customer and customer organization preferences and requirements.

6 Analysis and comparison of AWS computing services

The computing services provided by Amazon Web Services are discussed, analyzed, and compared in this chapter. The goal is to collect data to enable reliable service analysis, analyze and compare selected computing services based on survey results, and determine which services are best suited to the case company, providing answer to the final research question, *RQ3*.

6.1 Introduction to AWS computing services

Amazon Web Services (AWS) is currently one of the leading cloud platforms (Stack Overflow, 2021) with over 200 fully featured services to choose from (Amazon Web Services, 2022z). The services provided range from computing, networking and storage to analytics, robotics and IoT (Amazon Web Services, 2022z). This study focuses on services that provide computing and are suitable for hosting server-side applications. Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), Elastic Kubernetes Service (EKS), AWS Fargate, AWS App Runner, AWS Lambda, and AWS Elastic Beanstalk were chosen for further investigation together with the case company. The chosen services come in a range of service types and management options, from fully managed to those that require more user administration.

6.1.1 Related services and terms

The terms and services mentioned in this section are reviewed so that they can be referred to while presenting the services: (Amazon Web Services, 2022[e,i,j,l,n,p,u])

Managed service. Parts of operational and administrative burdens that a customer would normally have to manage, such as scaling, provisioning, and patching servers, are abstracted away by managed services.

Availability Zone (AZ). AZ consists of more than one data center.

Region. A region is a geographical place where AWS clusters data centers throughout the world. There are several AZs in each region.

EC2 Instance Connect. AWS offers both a browser-based Amazon EC2 console and an installable application called EC2 Connect CLI for connecting to an EC2 instance.

Amazon EC2 Auto Scaling. By creating Auto Scaling Groups in Amazon EC2 Auto Scaling service, the consumer can control the minimum and maximum number of instances active on each group, as well as the scaling policies that control when new instances are created and terminated (scaling out and scaling in).

Elastic Load Balancing (ELB). ELB distributes the incoming traffic among many targets in one or more Availability Zones, such as EC2 instances, ECS clusters and AWS Lambda.

Security group. A virtual firewall that adds rules based on protocols and port numbers to govern inbound and outbound traffic.

Amazon Elastic Container Registry (ECR). ECR is a container image registry service managed by AWS.

Virtual Private Cloud (VPC). VPC is a logically isolated virtual networking environment.

6.1.2 Amazon Elastic Compute Cloud (EC2)

Amazon EC2 offers virtualized scalable computing resources in the cloud. AWS refers to the virtual computing environments as instances, however they can also be referred to as virtual machines. AMIs (Amazon Machine Images) are pre-configured templates for the instances that include the operating system, pre-installed software, and data. Customers can build their own AMIs, purchase third-party AMIs, or use images managed and given by AWS. Instance types are used to configure resources and processing capabilities such as CPU, memory, storage, and networking. Amazon Elastic Block Store (Amazon EBS) is used to provide persistent storage for virtual machines. The resources' physical location can also be selected from a variety of regions and availability zones. EC2 is billed with a per-second model, with a minimum of 60 seconds. The cost varies based on the chosen instance type, pricing strategy (e.g., on-demand and reserved instances), needed persistent storage and used data transfer. Furthermore, EC2 is classified as IaaS, which indicates that the customer is given the ability to manage their own processing, storage, network, and other computing resources for developing and operating their own applications rather than relying on the cloud provider to do so. (Amazon Web Services, 2019a; 2022i)

A simple Amazon EC2 instance running on the AWS cloud is depicted in the diagram below (**Figure 18**). The instance connects to a database, Amazon RDS and to an object storage service, Amazon S3. The end-users, or clients, make HTTP queries to the web server running

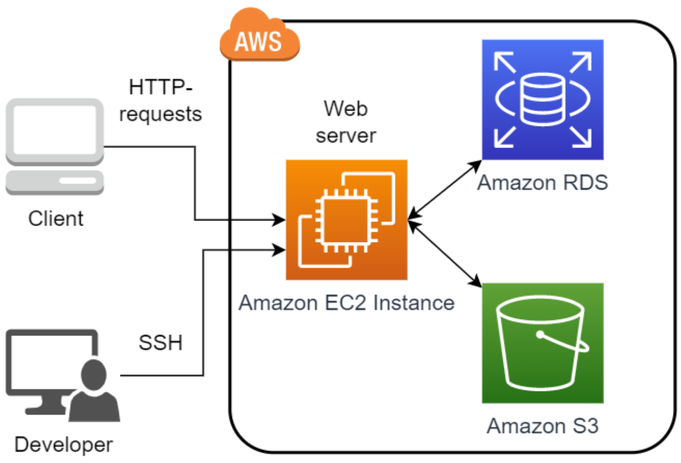


Figure 18. A simple architecture diagram for EC2 (Amazon Web Services, 2022i)

operating system, programming language runtime, and web server, among other things (Oulevay, 2021). AWS also offers other ways to connect to the EC2 instance, such as EC2 Instance Connect, and several ways and services to manage and deploy applications to the instances with integrations to popular version control systems such as GitHub. The example offered (**Figure 18**) does not consider issues like scalability, reliability, or security. Because EC2 is not a managed service, the customer is solely responsible for all configuration and administration, including the mentioned issues. Furthermore, EC2 can be combined with services like Amazon EC2 Auto Scaling and ELB, as well as deploying across different availability zones and/or regions, to achieve scalability and reliability. Security groups can be built to operate as virtual firewalls to control traffic and enforce instance level security. (Amazon Web Services, 2022i)

6.1.3 Amazon Elastic Container Service (ECS)

Amazon ECS is a fully managed "highly scalable and fast container management service" that can be used to run and manage containers on a cluster, as described by AWS. ECS offers a fully managed container orchestration service that can host monolithic containerized applications, microservices, and any other containerized workloads and applications in between. The containers are specified in Amazon ECS task specifications, which are used to perform specific tasks within a service. Each task can have one or more containers. Services are used to simultaneously operate and manage a set of tasks in a cluster, as illustrated on the figure below (**Figure 19**). Tasks and services can be run on AWS Fargate's serverless infrastructure or on a cluster of Amazon EC2 instances managed by the consumer. Furthermore, ECS provides integrations with Amazon Elastic Container Registry (ECR) and Docker to make deployment and utilization of CI/CD pipelines more convenient. (Amazon Web Services, 2022[c,o])

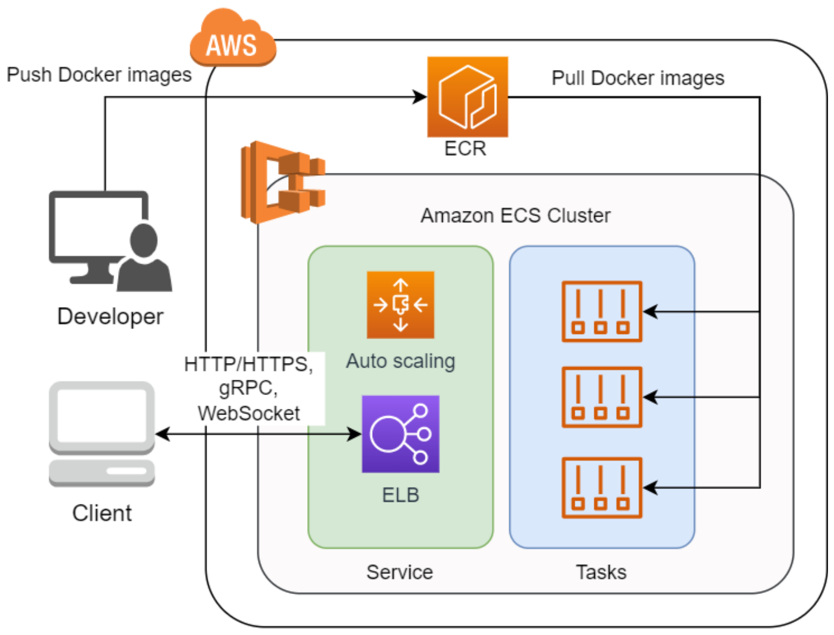


Figure 19 ECS architecture diagram (Amazon Web Services, 2022[c,o]; JFrog, 2019)

In the example above (**Figure 19**), the developer uploads a docker image to ECR, and the ECS services pull the new image for tasks that use the pulled images. Within the service, auto scaling may be utilized to scale tasks and container instances based on user-defined policies. ELB may be utilized within the service to distribute traffic between tasks and let clients connect to the application via protocols such as HTTP and WebSocket. There is no additional payment for utilizing ECS; the customer simply pays for the resources that are consumed (e.g., EC2 instances, persistent storage, used AWS Fargate resources). (Amazon Web Services, 2022o; JFrog, 2019)

6.1.4 Amazon Elastic Kubernetes Service (EKS)

Amazon EKS is a managed container management service for running and scaling Kubernetes applications on-premises or in the cloud. EKS manages container orchestration with Kubernetes and can be used to deploy monolithic and microservice architecture applications, but it excels at microservices and complex containerized applications that require reliability and scalability. (Amazon Web Services, [2022d,m])

Because the operation of Kubernetes is not the focus of this thesis, it will be discussed only briefly. Kubernetes is an open-source platform for containerized service management. The primary components of Kubernetes are clusters, nodes, pods, and control planes. A cluster is created once Kubernetes is deployed. The cluster is made up of worker machines known as nodes. The nodes run pods, which are groups of one or more application containers, such as Docker containers, that represent the application workload. A control plane is an orchestration layer that manages each node and pod in the cluster by defining, deploying, and controlling the lifecycle of containers. (Kubernetes, 2022)

AWS provides four alternative deployment choices for Amazon EKS, each requiring differing levels of management from the customer in terms of hardware, deployment location, Kubernetes control plane location, Kubernetes data location, and available technical support: (Amazon Web Services, 2022d)

1) *Amazon EKS*. A managed service that is operated and managed on AWS. AWS supplies hardware, and everything is deployed in AWS cloud.
2) *Amazon EKS on AWS Outposts.* A partially managed service, where EKS nodes are run on AWS Outposts. AWS supplies hardware, and the Kubernetes control plane is deployed in AWS cloud. Kubernetes deployment and data are in the consumer’s data center.
3) *Amazon EKS Anywhere*. Deployment option in which Kubernetes clusters are built and managed on-premises, with management via the Amazon EKS console. Customers run Kubernetes on their own infrastructure with AWS support, meaning that the hardware is provided by the customer, and everything is deployed on the customer's data center with AWS support.
4) *Amazon EKS Distro*. The Kubernetes software and dependencies used by Amazon EKS. Fully self-managed with no support from AWS.

As established, Amazon EKS uses the open-source Kubernetes software to operate, which means that existing Kubernetes-based applications are fully compatible with Amazon EKS. Amazon EKS, like Amazon ECS, can be used with AWS Fargate and EC2 instances. It is compatible with AWS services such as ECR, ELB, and VPC. EKS is billed based on the resources spent, like ECS, and each EKS cluster also has a fixed fee associated with running the control plane. (Amazon Web Services, 2022d)

6.1.5 AWS Fargate

AWS Fargate is used for serverless computing for containerized applications and workloads. It can be used with both the Amazon ECS and Amazon EKS as the computing engine. AWS Fargate features a serverless architecture that eliminates the need for customers to worry about scaling, patching, or managing servers. AWS provides Amazon Linux 2, Windows Server 2019 Full and Core versions as task operating systems. It uses a pay-as-you-go billing approach, like other serverless services, in which the user only pays for the resources used. (Amazon Web Services, 2022[h,k])

6.1.6 AWS App Runner

AWS App Runner is a fully managed service for running containerized applications. App Runner requires no prior infrastructure expertise; the customer simply sets up the service and submits the code or container image, and the service will take care of the rest, including developing, deploying, and scaling the app, as shown in the figure below (**Figure 20**). (Amazon Web Services, 2022[aa,ab])

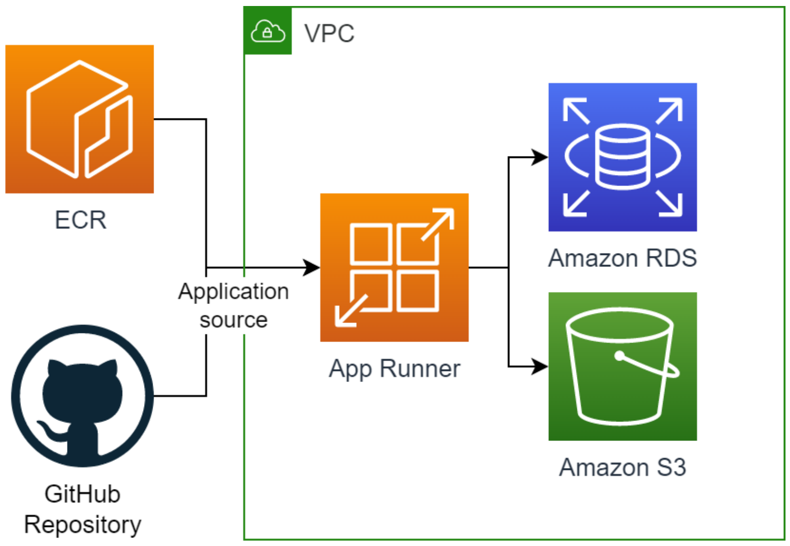


Figure 20. AWS App Runner architecture example (Amazon Web Services, 2022[ab])

App Runner currently integrates with GitHub repositories and ECR to enable automated deployments and provides easy access to other AWS services and applications operating in a private VPC such as databases, caches, and storage. Regardless of the application source (GitHub or ECR), The launch, operation, scalability, and load balancing of the service are all managed by App Runner. Although App Runner can run any containerized workload, including API services, backend web services, and microservices, it lacks the container orchestration features of ECS and EKS, therefore applications with a more complicated microservice architecture should utilize ECS or EKS instead. App Runner uses a pay-as-you-go billing model, in which the customer only pays for the resources used by the service. In addition, there are hourly fees for provisioned and active container instances. (Amazon Web Services, 2022[ab])

6.1.7 AWS Lambda

AWS Lambda offers serverless event-driven computing. The developers can run code on Lambda without provisioning or managing servers by organizing their application logic and code into Lambda Functions. Lambda Functions scale automatically and are only run when needed. Lambda manages all server and operating system administration, as well as capacity provisioning, scaling, and monitoring, so that the customer does not have to. Lambda Functions can be invoked from over 200 AWS services, e.g., in response to events. The figure below illustrates an example architecture of AWS Lambda (**Figure 21**). (Amazon Web Services, 2022[ac,ad])

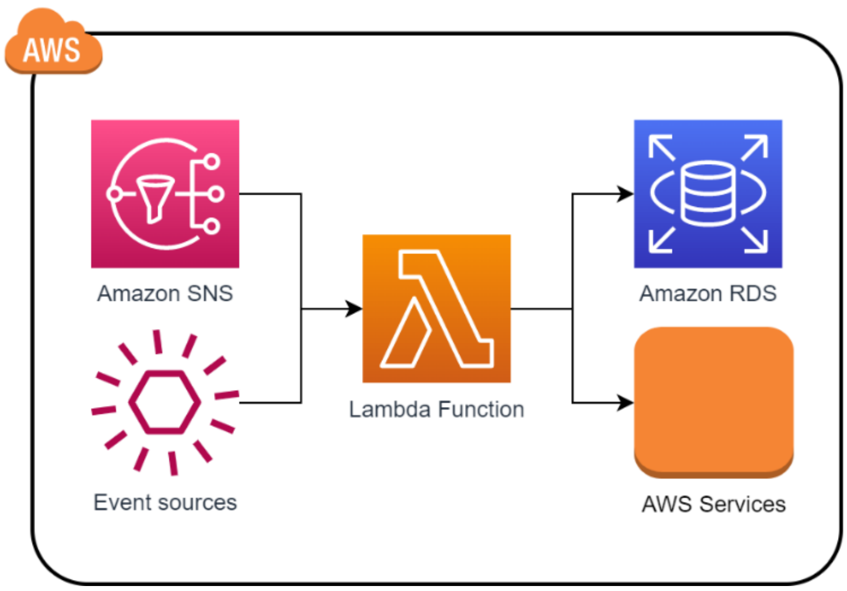


Figure 21. AWS Lambda architecture overview (Amazon Web Services, 2022[ac,ad])

As seen in the example above (**Figure 21**), the Lambda function is triggered by event sources and once executed, it can access other AWS services and resources. Other AWS services can provide event triggers to invoke the functions, such as uploading files to Amazon S3 or updating rows in an Amazon RDS database. Lambda functions can be configured to access other AWS services such as Amazon RDS and other Lambda functions via various policies. By using Lambda, the user only pays for the computing time consumed by the functions. (Amazon Web Services, 2022[ac,ad])

6.1.8 AWS Elastic Beanstalk

AWS Elastic Beanstalk offers a PaaS for deploying and managing applications on AWS. Elastic Beanstalk is the "fastest and simplest way to launch your application on AWS", according to AWS, while also giving the consumer complete control over the AWS resources used. AWS manages infrastructure and operational responsibilities such as server configuration, database maintenance, networking, and load balancing. Elastic Beanstalk offers several options for integrating AWS services such as logging, databases, storage, and content delivery networks (CDN) into the application environment, as well as the ability to automatically setup and create the required AWS resources. The figure below illustrates an example architecture of an AWS Elastic Beanstalk environment (**Figure 22**). (Amazon Web Services, 2022[ae,af])

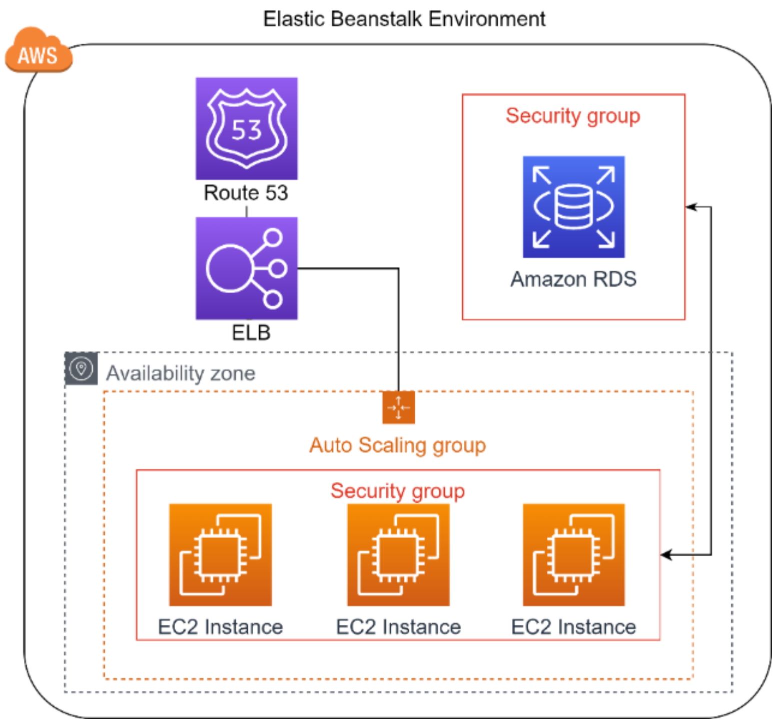


Figure 22. Elastic Beanstalk example (Amazon Web Services, 2022[af])

An example of Elastic Beanstalk environment architecture is shown in the figure above (**Figure 22**). An Auto Scaling Group is provisioned by default for each Elastic Beanstalk environment to automatically provision more instances and terminate them when the load decreases. In addition, each environment has an ELB to divide traffic among the Auto Scaling Group's instances. Finally, Amazon Route53, a Domain Name System (DNS) service, is used to alias an ELB URL. The URL is used to provide user access to the environment. Other resources necessary to operate the application are also automatically provisioned by Elastic Beanstalk, such as databases and storage services. Security groups are used to control access to these services. Customers who use Elastic Beanstalk only pay for the AWS resources that the application consumes; Elastic Beanstalk is not charged separately. (Amazon Web Services, 2022[af])

6.2 Analysis and comparison

Following the presentation of the 7 chosen computing services, a comparative study to uncover distinct use cases may start. The chosen services include IaaS, CaaS, FaaS, and PaaS solutions, serverful and serverless solutions, and AWS-managed and non-managed services. These characteristics alone have a significant impact on the computing solution chosen. Different service models determine how much real control the user has over the technology stack and resources they use, with IaaS having the most control and FaaS having the least. Serverless and managed services relieve the administrative and operational burdens associated with provisioning and managing servers.

Different characteristics of the chosen AWS services are shown in the table below (**Table 18**). The characteristics are chosen based on the findings of the questionnaire, the conceptual model, and the literature reviews on cloud computing, web, and mobile applications that have been presented. Microservices and containers are not widely used in the case company, and performance-related back-end quality factors and cloud service factors are not crucial in the case company's projects, whereas manageability, configurability, and security are. In the table, manageability is addressed by the service model, whether the service is managed by AWS, and if the service is serverless, and configurability by level of configurability available. Furthermore, the best-suited use cases based on the service's properties and AWS recommendations are presented to give context to the service's usual workloads. Security and performance are not included in this analysis because security is not particularly bound

Table 18. Chosen computing services and their characteristics (Amazon Web Services, 2022h)

Computing service	Description	Use cases	Configu-rability	Service model	Managed by AWS	Server-less
Amazon EC2	Provides virtualized scalable computing resources. In AWS, it is employed as a computing layer for more abstracted services (e.g., CaaS, PaaS)	Virtually any workload	High	IaaS	No	No
Amazon ECS	A fully managed orchestration service for running containerized workloads.	Container based applications and microservices	Medium-High	CaaS	Yes	Yes, with Fargate
Amazon EKS	A managed container management service for running and scaling Kubernetes applications on-premises or in the cloud.	Kubernetes applications	Medium-High	CaaS	Yes/No	Yes, with Fargate
AWS Fargate	Serverless computing engine for ECS and EKS.	Run containers serverless	Medium	CaaS	Yes	Yes
AWS App Runner	Fully managed serverless service for running containerized workloads.	Container based applications on a fully managed service	Low	CaaS	Yes	Yes
AWS Lambda	Serverless event-driven computing. Application logic and code are split into separate functions that run on a highly available computing infrastructure.	Virtually any workload	Low	FaaS	Yes	Yes
AWS Elastic Beanstalk	PaaS for deploying and managing applications on AWS. Provides complete control over used resources.	Web applications	Medium-High	PaaS	Yes	No

to individual computing services, but rather to the surrounding environment, and performance is not important to the case company's projects, as stated, and performance across the listed services is, after all, quite similar.

The services with the highest configurability available over the resources are Amazon EC2, Amazon ECS, Amazon EKS and AWS Elastic Beanstalk. Amazon EC2 is a highly configurable building block often utilized by other AWS services; ECS, EKS and Elastic Beanstalk all utilize Amazon EC2 under the hood to provide computing capabilities. In addition, ECS and EKS offer the use of AWS Fargate as an alternative to EC2, when the user does not need to directly manage their infrastructure. ECS and EKS are both used to run container-based applications that have a need for container orchestrations such as applications following a microservices architecture. Elastic Beanstalk on the other hand, is designed to provide an easy-to-use platform for building, running, and monitoring web applications. As mentioned, AWS Fargate provides serverless computing for ECS and EKS. AWS App Runner can be used for container-based applications instead of ECS and EKS when there is no need for complex container orchestration. AWS Lambda can be used for almost any workload that can run on Lambda's runtime and resources, such as Web APIs or custom data processing.

A generic use case guidance is provided in the picture below based on the presented analysis and comparison results (**Figure 23**). Choosing the computing service candidate starts with asking a question whether full control over the service and its resources is needed. As stated, Amazon EC2 offers the nearest bare-bone computing AWS has to offer and is the IaaS product in the chosen computing services. EC2 should be chosen when developers need a highly configurable service with access to the operating system configuration and runtime. Moreover, EC2 should be chosen as the computing building block for managed services (especially ECS and EKS) for the same reasons. The next question is about the backend architecture, specifically whether the workloads follow a microservices architecture. If not, it is advised that AWS Elastic Beanstalk is used. AWS Elastic Beanstalk uses highly configurable EC2 instances, but it's still an AWS managed service that can automatically manage and deploy services if needed. Elastic Beanstalk is most often used to deploy and

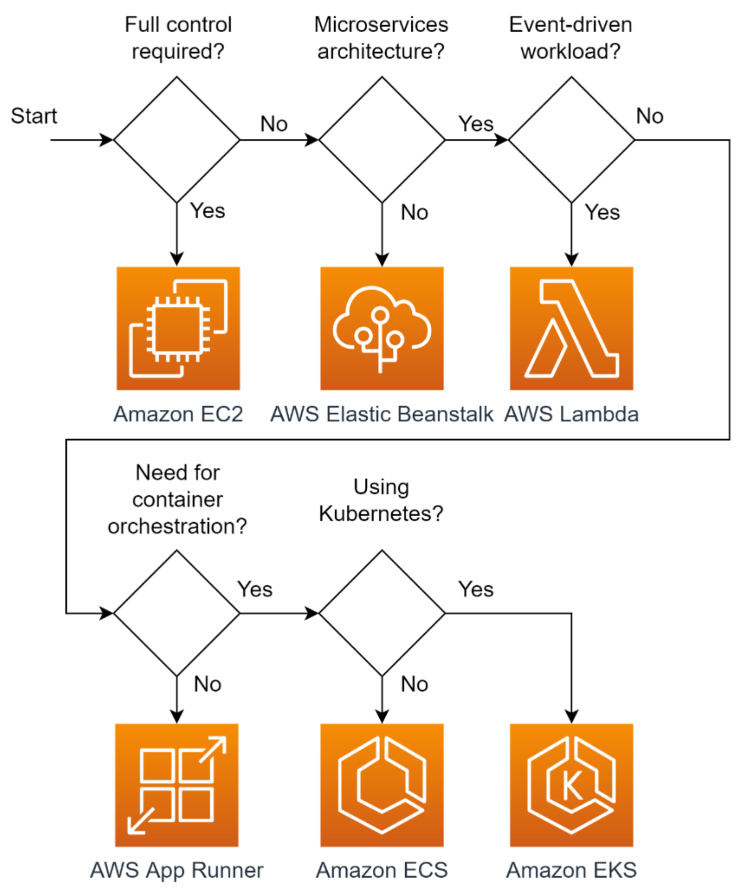


Figure 23. A generic use case guide for choosing an AWS computing candidate.

scale web applications. If the workloads are event-driven, AWS Lambda should be used since it provides event-driven serverless computing, which fits well with the nature of microservices, where an application's capabilities are divided down into numerous distinct logical entities, or services. Continuing the use of microservices architecture, AWS App Runner should be used for containerized workloads where extensive container orchestration is not required, and minimal software stack setup is required. App Runner should be utilized for simpler workloads than ECS and EKS, and when there is a need to get a simple containerized application or workload up and running quickly. Finally, if there is a need to run containerized applications with a fully-fledged container orchestration, either Amazon ECS or Amazon EKS can be chosen. Moreover, EKS should be used when Kubernetes is used or is planned to be used.

The pricing between the chosen services varies as seen on the table below (**Table 19**). Services that utilize EC2 (Elastic Beanstalk, ECS and EKS), and EC2 itself, are billed based on the pricing model, the instance specifications that determine the amount of computing power and memory the instance will have, and the amount and type of storage assigned to

Table 19. Computing services chosen and their associated pricing models

Computing service	Pricing model
Amazon EC2	Per-second, 60 second minimum. Cost is based on the pricing model, instance type (specifications) and persistent storage.
Amazon ECS	No additional fee. Billed for consumed resources (EC2, Fargate, S3, etc.)
Amazon EKS	Fixed cost per cluster. Billed for consumed resources (EC2, Fargate, S3, etc.)
AWS Fargate	Billed for consumed resources. Per second, 60 second minimum.
AWS App Runner	Billed for consumed resources. Hourly fees for provisioned and active container instances deployed to the service.
AWS Lambda	Billed based on the number of requests and the duration taken to execute the code rounded up to the nearest millisecond.
AWS Elastic Beanstalk	No additional fee. Billed for consumed resources (EC2, S3, etc.).

the instance. The usage of EC2 is billed with a per-second model, with a minimum of 60 seconds. ECS is only charged for the resources used, whether the launch type is Fargate or EC2. EKS on the other hand, has a fixed cost associated with each EKS cluster. AWS Fargate is billed similarly to EC2, as it is based on the specified computing resources (CPU and memory) as well as based on the chosen operating system, CPU architecture and storage resources consumed. Fargate charges on a per-second basis, with a minimum of 60 seconds (15 minutes for Windows containers) for the duration between the start of the container image download and the termination of the task. AWS Lambda is billed based on the number of requests and duration taken to execute the code, rounded up to the nearest millisecond. The cost associated with the execution time depends on the assigned amount of memory allocated to the function. AWS Elastic Beanstalk is also only billed based on the used resources, such as EC2 instances for computing, S3 for object storage and Amazon RDS for databases. Computing resources (EC2, Fargate), storage (EBS), and outbound data transfer are all examples of consumed resources. Serverless tends to be more expensive to use than serverful solutions, but they are faster to setup and require less work to maintain and manage.

7 Conclusions and discussion

This chapter provides observations on how well the research fulfilled its objectives by providing answers and discussion to the research questions. The study's implications for practitioners and academics in adjacent disciplines are examined, as well as its shortcomings and future projections in this area of research.

7.1 Research questions and related discussion

RQ:1 What requirements and quality factors should be considered when choosing cloud computing services for web and mobile application projects from a technical viewpoint?

The first research question, RQ1, is answered by a combination of the two chapters, 3 and 4. Chapter 3 identifies the relevant needs and technological aspects related with web and mobile applications by providing a literature review on the associated architecture, characteristics, and related notions of web and mobile apps as well as server-side components. Chapter 4 continues describing relevant quality factors related to web and mobile application projects from a technology standpoint by providing a conceptual model.

Cloud computing services are used to provide computing resources for back-end solutions, such as web and mobile application server-side components. Hence, in order to respond to RQ1, the server-side components of web and mobile applications must be reviewed. In Chapters 3 and 4, the following factors were identified: 1) the required degree of service configuration, 2) the required level of cloud infrastructure management and governance, 3) the architecture of the server-side application and the chosen technology stack, and 4) the customer's security and performance requirements. In addition, the conceptual model developed in Chapter 4 presents the most essential quality factors to consider when selecting cloud computing services, such as manageability, interoperability, security, and business value, as well as various sub-factors for each quality factor from the identified key stakeholders: development team, project management, customers, and end-users.

Furthermore, to produce quality software, one must be able to assess the customer's current wishes and values, predict how they may evolve, and make technological decisions that allow for future adjustments. Similar decisions, regarding aspects such as scalability and support for new features, must be made when choosing cloud services in addition to the listed requirements and quality factors. For example, when relying on MBaaS services to replace self-written server-side components, it’s solely up to the service provider to develop new features and manage scalability. Consequently, using MBaaS services can result in vendor lock-in (Roberts, 2017, 6), making it difficult to switch to a different vendor or service if the current one proves inadequate or decides to shut down the service. Building everything from the ground up and hosting the solutions with serverful context such as IaaS services, on the other hand, adds a lot of possibly unnecessary complexity and maintenance work, but gives the developers more flexibility in terms of what can and cannot be done.

During the review and analysis of AWS computing services, it became evident that the thesis was able to identify the main features of web and mobile projects related to choosing cloud computing services thanks to the thorough baseline research conducted in Chapter 2. It was difficult to identify the quality factors since there was inadequate supporting research. Fortunately, a WBA-related conceptual model was discovered and used to establish a new conceptual model that included web and mobile application development as well as cloud computing services. The established conceptual model proved to be quite comprehensive during the questionnaire construction.

RQ:2 What are the most important quality-related technical factors and needs when choosing cloud computing services for the case company’s typical web and mobile projects according to the case company’s developers and management?

The second research question is addressed in Chapter 5, which outlines the concepts and rationale behind conducting a survey, as well as how to create one for the thesis. The data collection tool was a questionnaire, which was used to collect responses from the case company's employees between 06.05.2022 and 20.05.2022, yielding a total of 24 replies. The questionnaire had three discrete sections: 1) background section, used to identify the responders' background and experience with web and mobile projects; 2) back-end section, to establish the case company's back-end needs and requirements, and 3) cloud service section, to establish the background of cloud computing services currently used in the case company.

Based on the findings of the first section, it is clear that the respondents have sufficient experience and have worked on a significant number of web and mobile projects and are thus sufficiently informed about the survey's subject and the case company's projects to participate and contribute value to the study. To further analyze findings from sections 2 and 3, the responders were categorized into four distinct sub-groups based on their experience. The responders are familiar with container technologies, and containers are often utilized in projects, but microservices are not. Microservices is a complicated architecture concept that needs a considerable number of resources and expertise to effectively implement. According to the findings, the majority of small to medium-sized web and mobile applications do not necessitate such a complicated architecture design and are therefore not used as often. More experienced respondents favor IaaS, followed by CaaS and FaaS, and lastly PaaS, which is the least preferred service model among all respondents and sub-groups. Less experienced respondents favored FaaS and CaaS over IaaS, implying that more abstracted services and serverless are becoming more popular. However, as each situation is different, it's impossible to say which option would be the best in every scenario. The survey results indicate which service models are being used in the case company, but it is not possible to draw firm conclusions about which model should be always used; rather, the best-fitting service model is highly use-case dependent on numerous factors such as application workload, development team skills and resources, and client and development team preferences. Furthermore, the choice of technologies like as containers and paradigms such as microservices is heavily impacted by the demands and preferences of the client, rather than being solely in the hands of the case company.

The influence of back-end and cloud service quality factors on cloud service model selection was also looked into more extensively. The backend quality factors listed in the thesis are important but not critical by the respondents. The less experienced respondents seemed to value the quality factors more than the more experienced respondents. The significance of these factors in projects varies greatly, and in the end, the workloads and uses of the application ultimately determine the cloud service and computing model that is used. Security and manageability are clearly the most important cloud service quality factors for all subgroups. At least in this context and with the related list of quality factors, compatibility and configuration are more important than performance. Cost structures, GDPR and logging, customer and customer organization preferences and requirements were the most mentioned topics in the open-ended questions in both back-end and cloud-services. When all of the questions were broken down by subgroups, there were differences in the responses, indicating that work experience and project involvement have an impact on the responses, as expected.

The questionnaire produced interesting insights how work experiences affect which quality factors were valued over the others, as well as how the usage of computing service models distributed between the sub-groups. However, if a different data collection method had been used with the survey, such as interviews or inspecting existing project artefacts such as environments and requirement specifications, the results could have led to a situation where more specific guidance on which computing service model to use and when could have been given. That is, if the projects could be approximately classified with comparable computing requirements, etc. Nevertheless, the questionnaire was successful in answering RQ1.1 and identifying the most relevant quality-related technical factors and needs on typical projects of the case company.

RQ:3 Which of the AWS cloud computing services designed for hosting server applications offer the greatest quality for the case company's typical web and mobile projects based on the previously identified factors and needs?

The third research question is addressed by Chapter 6, which presents the comparison and analyzing of the seven AWS computing services which were chosen for the comparison with the case company. These services are Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), Amazon Elastic Kubernetes Services (EKS), AWS Fargate, AWS App Runner, AWS Lambda, and AWS Elastic Beanstalk.

Following the analysis, a use-case guide (see **Figure 23**) was developed, which outlines the key decisions that must be addressed when selecting computing services for the project's application workloads. The use-case guide centers on subjects like how much control is required from the computing service, what architecture the application workload utilizes, if the workloads are event-driven, whether container orchestration is required, and whether Kubernetes is utilized. Services like Amazon EC2, Amazon ECS, AWS Lambda, and AWS Elastic Beanstalk are among the best suitable services for the case company's small to medium-sized web and mobile projects, according to the survey results and the use-case guide.

It should be noted that the constructed use-case guide only offers general guidelines, and that in real-world use cases, the developers from the case company will need to examine each workload of the application individually and will certainly use multiple computing services in their solutions, such as Lambdas for custom data processing and AWS Elastic Beanstalk for hosting web applications. In addition to the guidelines offered, there are various other variables to consider while selecting services. Some services are restricted by the communication protocols used; for example, Lambda does not support SOAP. In general, AWS favors RESTful service design. Furthermore, each computing service has its own set of supported programming languages and runtimes. All the chosen services support the use of containers and typically, the more managed the computing service is, the less configuration options and development options it offers, such as supported programming languages, runtimes etc.

Upon the analysis and comparison, it quickly became clear that recommending a single service or multiple services accurately for the case company's projects was impossible, because determining the best service for each project necessitates examining each workload of the application separately, and certainly using multiple computing services in single solutions. More data, especially on the currently existing projects, their specifications, and environments, may have resulted in more precise recommendations. Interviews as part of the questionnaire, as previously indicated, would have provided more data to work with. In conclusion, the case company is advised to use the constructed use-case guide as a guideline and select computing services after examining the client's requirements and preferences, the development team's preferences, and the types of application workloads. The case company should also incorporate planning for cloud services early on in the software project's lifecycle as detailed by Alshazly et al. (2021) in the CSLCP model.

7.2 Implications for researchers and practitioners

The thesis offers discussion of web and mobile projects needs and technical perspective, factor evaluation mainly from the development team perspective, as well as comparison and analysis of AWS services. There is a clear research gap in studies that address the quality aspects of web and mobile projects from the standpoint of cloud services. Nabil's WBA conceptual model focuses on the quality aspects of WBA projects, emphasizing how project stakeholders may be identified as well as the appropriate quality criteria. The CSLCP model by Alzhaszly et al. (2021) describes how the lifecycle model of a whole software project must be addressed when evaluating cloud or web services. In addition, OASIS Open (2021) published a web service quality standard in 2012. In this thesis, the CSLCP model and web service quality standard were utilized to augment the WBA conceptual model to meet the expectations of web and mobile applications from a cloud services perspective. The thesis serves as a starting point for further academic study in the field of quality in cloud computing in software projects.

In addition, theses like this one, which compile essential knowledge in the field and prioritize the usage of the corporate environment, are important sources of information for practitioners like students, software professionals, and IT organizations. In light of recent research, many students are not yet able to make clear personal branding (Happonen et al., 2021a) and systematic career planning that would meet the expectations put on the students (Happonen et al., 2021b). Hence, reading the core issues important to the software industry can be beneficial in competence development, during studies. The proposed conceptual model and use-case guide for cloud computing services will help software professionals and IT organizations make informed decisions when evaluating prospective services for their projects and application workloads.

7.3 Shortcomings and future research priorities

The main data collection approach was a questionnaire, and it would have been advantageous to the study to have another data collection method alongside it in order to make recommendations based on actual data from the projects. Supporting literature for the work was limited, however if more material had been available, more viewpoints may have been included in the literature assessment, making the conclusions more credible. Instead of peer-reviewed academic literature, which would have been more reliable, the majority of the AWS-services evaluations were dependent on documentation provided by AWS. The study examines themes from a technical standpoint and focuses on small to medium-sized online and mobile projects from the perspective of a development team. As a result, some details may have been ignored because the focus was solely on the listed stakeholders and technical details. Furthermore, the study only looked at AWS computing services that were chosen in collaboration with the case company, therefore no other cloud providers were considered.

Future study could focus on 1) determining the demands of web and mobile projects, 2) determining quality characteristics for web and mobile projects for all key stakeholders, and 3) comparing and analyzing AWS computing capabilities with those of other cloud service providers. A framework and a guidebook based on these themes could be developed in the future to help organizations evaluate the usage of cloud computing services in their web and mobile projects, as well as determine preferences and the most critical quality factors for their organization. For the case company, additional research into project needs, i.e., a systematic examination of projects and their specifications, might be done to enable data-based conclusions about project needs to make suggestions on the used computing services.

References

Al-Debagy, O. and Martinek, P. (2018). A Comparative Review of Microservices and Monolithic Architectures, IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), doi: 10.1109/CINTI.2018.8928192.

Alshazly, A.A., ElNainay, M.Y., El-Zoghabi, A.A. and Abougabal, M.S. (2021). A cloud software life cycle process (CSLCP) model. Ain Shams Engineering Journal, 12(2), doi: 10.1016/j.asej.2020.11.004, pp.1809–1822.

Amazon Web Service (2022a). Introduction - AWS Serverless Multi-Tier Architectures with Amazon API Gateway and AWS Lambda. Available at: https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/introduction.html [Accessed 12 Apr. 2022].

Amazon Web Service (2022ae). AWS Elastic Beanstalk – Deploy Web Applications. Amazon Web Services, Inc. Available at: https://aws.amazon.com/elasticbeanstalk/ [Accessed 2 May. 2022].

Amazon Web Service (2022af). What Is AWS Elastic Beanstalk? - AWS Elastic Beanstalk. Available at: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html [Accessed 2 May. 2022].

Amazon Web Services (2019a). Amazon EC2 Pricing - Amazon Web Services. Available at: https://aws.amazon.com/ec2/pricing/ [Accessed 22 Apr. 2022].

Amazon Web Services (2022aa). AWS App Runner – Fully managed container application service - Amazon Web Services. Available at: https://aws.amazon.com/apprunner/ [Accessed 29 Apr. 2022].

Amazon Web Services (2022ab). What is AWS App Runner? - AWS App Runner. Available at: https://docs.aws.amazon.com/apprunner/latest/dg/what-is-apprunner.html [Accessed 29 Apr. 2022].

Amazon Web Services (2022ac). What Is AWS Lambda? - AWS Lambda. Available at: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html [Accessed 29 Apr. 2022].

Amazon Web Services (2022ad). AWS Lambda – Serverless Compute - Amazon Web Services. Amazon Web Services, Inc. Available at: https://aws.amazon.com/lambda/

Amazon Web Services (2022b). Amazon API Gateway FAQs | API Management | Amazon Web Services. Available at: https://aws.amazon.com/api-gateway/faqs/ [Accessed 28 Mar. 2022].

Amazon Web Services (2022c). Amazon ECS - Run containerized applications in production. Available at: https://aws.amazon.com/ecs/ [Accessed 22 Apr. 2022].

Amazon Web Services (2022d). Amazon EKS - Managed Kubernetes Service. Available at: https://aws.amazon.com/eks/ [Accessed 22 Apr. 2022].

Amazon Web Services (2022e). Amazon Virtual Private Cloud (VPC). Amazon Web Services, Inc. Available at: https://aws.amazon.com/vpc/ [Accessed 22 Apr. 2022].

Amazon Web Services (2022f). Appendix: SOAP API - Amazon Simple Storage Service. Available at: https://docs.aws.amazon.com/AmazonS3/latest/API/APISoap.html [Accessed 28 Mar. 2022].

Amazon Web Services (2022g). Building Monoliths or Microservices - Developing and Deploying .NET Applications on AWS. Available at: https://docs.aws.amazon.com/whitepapers/latest/develop-deploy-dotnet-apps-on-aws/building-monoliths-or-microservices.html [Accessed 19 Apr. 2022].

Amazon Web Services (2022h). Compute Services - Overview of Amazon Web Services. Available at: https://docs.aws.amazon.com/whitepapers/latest/aws-overview/compute-services.html [Accessed 22 Apr. 2022].

Amazon Web Services (2022i). What Is Amazon EC2? - Amazon Elastic Compute Cloud. Amazon.com. Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html [Accessed 22 Apr. 2022].

Amazon Web Services (2022j). Global Infrastructure Regions & AZs. Available at: https://aws.amazon.com/about-aws/global-infrastructure/regions_az/ [Accessed 22 Apr. 2022].

Amazon Web Services (2022k). Serverless Compute Engine–AWS Fargate–Amazon Web Services. Available at: https://aws.amazon.com/fargate/?c=cn&sec=srv [Accessed 28 Apr. 2022].

Amazon Web Services (2022l). What Is Amazon EC2 Auto Scaling? - Amazon EC2 Auto Scaling. Available at: https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html [Accessed 22 Apr. 2022].

Amazon Web Services (2022m). What is Amazon EKS? - Amazon EKS. Available at: https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html [Accessed 22 Apr. 2022].

Amazon Web Services (2022n). What is Amazon Elastic Container Registry? - Amazon ECR. Available at: https://docs.aws.amazon.com/AmazonECR/latest/userguide/what-is-ecr.html [Accessed 22 Apr. 2022].

Amazon Web Services (2022o). What is Amazon Elastic Container Service? - Amazon Elastic Container Service. Available at: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html [Accessed 22 Apr. 2022].

Amazon Web Services (2022p). What Is Elastic Load Balancing? - Elastic Load Balancing. Available at: https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html [Accessed 22 Apr. 2022].

Amazon Web Services (2022r). What is an API? - API Beginner's Guide - AWS. Available at: https://aws.amazon.com/what-is/api/ [Accessed 7 Mar. 2022].

Amazon Web Services (2022s). AWS Lambda – FAQs. Available at: https://aws.amazon.com/lambda/faqs/ [Accessed 04 Feb 2022].

Amazon Web Services (2022t). Appendix - Amazon Simple Storage Service. [online] Available at: https://docs.aws.amazon.com/AmazonS3/latest/API/appendix.html [Accessed 29 Apr. 2022].

Amazon Web Services (2022u). Regions, Availability Zones, and Local Zones - Amazon Elastic Compute Cloud. Available at:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html [Accessed 12 Feb 2022].

Amazon Web Services (2022v). Shared Responsibility Model - Amazon Web Services (AWS). Amazon Web Services, Inc. Available at: https://aws.amazon.com/compliance/shared-responsibility-model/ [Accessed 04 Feb 2022].

Amazon Web Services (2022w). Six Advantages of Cloud Computing. Available at: https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroupLifecycle.html [Accessed 18 Feb 2022].

Amazon Web Services (2022x). Using resource-based policies for AWS Lambda. Available at: https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html [Accessed 22 Feb 2022].

Amazon Web Services (2022y). Mobile Application Development. Available at: https://aws.amazon.com/mobile/mobile-application-development/ [Accessed 4 Mar. 2022].

Amazon Web Services, (2022q). Building microservices on AWS Workshop. Available at: https://www.microservicesworkshop.com/usecase/architecture.html [Accessed 22 Feb 2022].

Amazon Web Services. (2019). What are Microservices? | AWS. Available at: https://aws.amazon.com/microservices/ [Accessed 19 Apr. 2022].

Amazon Web Services. (2022z). What is AWS? - Amazon Web Services. Available at: https://aws.amazon.com/what-is-aws/ [Accessed 03 Feb 2022].

Arutyunov, V.V. (2012). Cloud computing: Its history of development, modern state, and future considerations. Scientific and Technical Information Processing, 39(3), p.173.

Atlassian (n.d.). Microservices vs. monolithic architecture. Atlassian. Available at: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith [Accessed 19 Apr. 2022].

Atlassian. (2019). Agile best practices and tutorials | Atlassian. Atlassian. Available at: https://www.atlassian.com/agile [Accessed 24 Mar. 2022].

Banerjee, S. (2016). Role of a Project Manager in Managing Agile Projects. Journal of Business & Financial Affairs, 5(3). doi: 10.4172/2167-0234.1000204

Birrell, A.D. and Nelson, B.J. (1984). Implementing remote procedure calls. ACM Transactions on Computer Systems, 2(1), doi: 10.1145/2080.357392, pp.39–42.

Brown, W. J., Anderson, V., Tan, Q. (2012) "Multitenancy - Security Risks and Countermeasures," 15th International Conference on Network-Based Information Systems, 2012, pp. 7-13, doi: 10.1109/NBiS.2012.142.

Casteleyn, S., Daniel, F., Dolog, P. and Matera, M. (2009). Engineering Web Applications. Berlin, Heidelberg: Springer Berlin Heidelberg.

Check J., Schutt R. K. (2012). Survey research. In: J. Check, R. K. Schutt., editors. Research methods in education. Thousand Oaks, CA: Sage Publications, p. 160.

CNCF. (2018). CNCF WG-Serverless Whitepaper v1.0. Available at: https://github.com/cncf/wg-serverless/tree/master/whitepapers/serverless-overview [Accessed 20 Feb. 2022].

Crowder, J.A. and Friess, S. (2015). Agile project management: managing for success. Cham: Springer, pp.19-35.

Durao, F., Carvalho, J.F.S., Fonseka, A., and Garcia, V.C. (2014). A systematic review on cloud computing. The Journal of Supercomputing, 68(3), p.1322.

Fette, I. and Melnikov, A. (2011). The WebSocket Protocol. RFC. Available at: https://www.semanticscholar.org/paper/The-WebSocket-Protocol-Fette-Melnikov/6a7f34970ece556019431dde42daa4982bfcc13d [Accessed 7 Mar. 2022].Foote, K. (2017). A Brief History of Cloud Computing - DATAVERSITY. DATAVERSITY. Available at: https://www.dataversity.net/brief-history-cloud-computing/ [Accessed 04 Feb 2022].

Happonen, A., Manninen, L., Hirvimäki, M., Nolte, A. (2021b), Expectations for young job applicants' digital identity related to company's social media brand development strategies, Small Enterprise Research, Vol. 10, No. 2, pp. 170-180, doi: 10.1080/13215906.2021.2000482

Happonen, A., Manninen, L., Santti, U., Mariappan, M. (2021a), Online brand, opportunities, realities and challenges for SMEs. Fresh recruits, a solution or new kind of orienteering challenge? International journal of engineering & technology, Vol. 10, No. 2, Article: 31813, pp. 220-231, doi: 10.14419/ijet.v10i2.31813

Hartikainen, J. (2020). Defining suitable testing levels, methods and practices for an agile web application project. Available at: https://lutpub.lut.fi/bitstream/handle/10024/161159/mastersthesis_hartikainen_ville.pdf?sequence=1&isAllowed=y.

Hashida, O. and Sakata, H. (2001). 'Client/Server Technology', in Handbook of Industrial Engineering. Hoboken, NJ, USA: John Wiley & Sons, Inc. pp. 715–716.

Hussein, M.K., Mousa, M.H. and Alqarni, M.A. (2019). A placement architecture for a container as a service (CaaS) in a cloud environment. Journal of Cloud Computing, 8(1). doi: 10.1186/s13677-019-0131-1, pp.1-3.

IBM (2019a). What is software testing? Ibm.com. Available at: https://www.ibm.com/topics/software-testing [Accessed 25 Mar. 2022].

IBM (2022a). Remote Procedure Call. www.ibm.com. Available at: https://www.ibm.com/docs/en/aix/7.1?topic=concepts-remote-procedure-call [Accessed 7 Mar. 2022].

IBM (2022b). WebSocket. Available at: https://www.ibm.com/docs/en/was-liberty/base?topic=liberty-websocket [Accessed 7 Mar. 2022].

IBM Cloud Education (2020). What is Three-Tier Architecture. www.ibm.com. Available at: https://www.ibm.com/cloud/learn/three-tier-architecture [Accessed 2 Mar. 2022].

IBM. (2011). Microservices. Available at: https://www.ibm.com/cloud/learn/microservices [Accessed 19 Apr. 2022].

IBM. (2019b). What is software development? Available at: https://www.ibm.com/topics/software-development [Accessed 24 Mar. 2022].

IDG. (2022). IDG Cloud Computing Survey 2020. Available at: https://foundryco.com/tools-for-marketers/research-cloud-computing/ [Accessed 03 Feb 2022].

Interaction Design Foundation (2019). What is User Experience (UX) Design? The Interaction Design Foundation. Available at: https://www.interaction-design.org/literature/topics/ux-design [Accessed 25 Mar. 2022].

ISO/IEC. (2000). ISO/IEC FDIS 9126-1 Information technology — Software product quality — Part 1: Quality model. Available at: https://www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-1%20Standard.pdf [Accessed 11 Mar. 2022].

JFrog (2019). Hosting Your Application on Amazon Cloud Container Service. Available at: https://jfrog.com/blog/5-steps-to-hosting-your-application-on-amazon-cloud-container-service/ [Accessed 22 Apr. 2022].

Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C.-C., Khandelwal, A., Pu, Q., Shankar, V., Carreira, J., Krauth, K., Yadwadkar, N., Gonzalez, J., Popa, R. and Patterson, D. (2019). Cloud Programming Simplified: A Berkeley View on Serverless Computing. Available at: https://arxiv.org/pdf/1902.03383.pdf pp.5-6. [Accessed 19 Feb 2021].

Kitchenham, B. & Pfleeger, S. L. (2002a) Principles of survey research: part 5: populations and samples. ACM SIGSOFT Software Engineering Notes 27(2), doi: 10.1145/571681.571686, pp.17–20.

Kitchenham, B. A. & Pfleeger, S. L. (2002b) Principles of survey research: part 3: constructing a survey instrument. ACM SIGSOFT Software Engineering Notes 27(2), doi: 10.1145/511152.511155, pp.20–24.

Kitchenham, B. and Pfleeger, S.L. (2002c). Principles of survey research part 4. ACM SIGSOFT Software Engineering Notes, 27(3), doi: 10.1145/638574.638580, pp.20–23.

Kitchenham, B. and Pfleeger, S.L. (2003). Principles of survey research part 6. ACM SIGSOFT Software Engineering Notes, 28(2), doi: 10.1145/638750.638758, p.24-27.

Kitchenham, B.A. and Pfleeger, S.L. (2002d). Principles of survey research. ACM SIGSOFT Software Engineering Notes, 27(2), doi: 10.1145/511152.511155, p.20.

Kitchenham, B.A. and Pfleeger, S.L. (2002e). Principles of survey research part 2. ACM SIGSOFT Software Engineering Notes, 27(1), doi: 10.1145/566493.566495, pp.18–20.

Kubernetes (2022). What is Kubernetes. Kubernetes.io. Available at: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ [Accessed 22 Apr. 2022].

Kuuskeri, J. (2014). Engineering Web Applications; Architectural Principles for Web Software. Doctoral Dissertation. doi: 10.13140/RG.2.1.2457.4889. pp. 5-18

Lawi, A., Panggabean, B.L.E. and Yoshida, T. (2021). Evaluating GraphQL and REST API Services Performance in a Massive and Intensive Accessible Information System. Computers, 10(11), p.138.

Luo, X., Zhang, S. and Litvinov, E. (2019). Practical Design and Implementation of Cloud Computing for Power System Planning Studies. IEEE Transactions on Smart Grid, 10(2), p.2303.

Mahmood, Z. Saeed, S. (2013). Software Engineering Frameworks for the Cloud Computing Paradigm. London: Springer London. pp. 35-80.

Mell, P. and Grance, T. (2011). The NIST Definition of Cloud Computing. NIST Special Publication 800-145, pp.1-7.

Metso, L., Happonen, A., Ojanen, V., Rissanen, M., Kärri, T. (2019), Business model design elements for electric car service based on digital data enabled sharing platform, Cambridge International Manufacturing Symposium, 26-27.09.2019: Cambridge, UK, doi: 10.17863/CAM.45886, p. 6.

Metso, L., Happonen, A., Rissanen, M., Efvengren, K., Ojanen, V., Kärri, T. (2020), Data Openness Based Data Sharing Concept for Future Electric Car Maintenance Services, In Advances in Asset Management and Condition Monitoring. Smart Innovation, Systems and Technologies, Vol 166, Chapter 36, doi: 10.1007/978-3-030-57745-2_36, pp. 429-436.

Microsoft 2021. WCF and ASP.NET Web API - WCF. docs.microsoft.com. Available at: https://docs.microsoft.com/en-us/dotnet/framework/wcf/wcf-and-aspnet-web-api [Accessed 28 Mar. 2022].

Microsoft Azure 2022. API design guidance - Best practices for cloud applications. Microsoft.com. Available at: https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design [Accessed 8 Mar. 2022].

Nabil, D., Mosad, A. and Hefny, H.A. (2011). Web-Based Applications quality factors: A survey and a proposed conceptual model. Egyptian Informatics Journal, 12(3), doi: 10.1016/j.eij.2011.09.003, pp.211–217.

Nusairat, J. F. (2020). Rust for the IoT : building Internet of Things apps with Rust and Raspberry Pi. California: Apress, pp-133-135.

OASIS Open. 2012. Web Services Quality Factors Version 1.0. Available at: http://docs.oasis-open.org/wsqm/WS-Quality-Factors/v1.0/cos01/WS-Quality-Factors-v1.0-cos01.html#_Toc342315478 [Accessed 15 Mar. 2022].

Onrego. (2020). IaaS, CaaS, PaaS, FaaS, SaaS – mitä mikäkin tarkoittaa? Available at: https://onrego.fi/julkisen-pilven-palvelumallit-avattuna/ [Accessed 17 Feb. 2022].

Oulevay, S. (2021). COMEM+ Architecture & Deployment course at HEIG-VD: Cloud Computing. Available at: https://github.com/MediaComem/comem-archidep [Accessed 17 Feb 2022].

Pomberger, G., Bischofberger, W., Kolb, D., Pree, W-. Schlemm, H. (1998). Prototyping-Oriented Software Development - Concepts and Tools. Structured Programming. 12. pp. 43-46

Ponto, J. (2015). Understanding and Evaluating Survey Research. Journal of the advanced practitioner in oncology. 6. 168-171.

Red Hat (2019a). What is an API? Redhat.com. Available at: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces [Accessed 28 Mar. 2022].

Red Hat. (2019b). What is container orchestration? Available at: https://www.redhat.com/en/topics/containers/what-is-container-orchestration [Accessed 17 Feb 2022].

Red Hat. (2020). What is CaaS? Available at: https://www.redhat.com/en/topics/cloud-computing/what-is-caas [Accessed 17 Feb 2022].

Roberts, M. (2017). What is serverless: understand the latest advances in cloud and service-based architecture. Available at: https://www.symphonia.io/what-is-serverless.pdf [Accessed 17 Feb 2022] pp.7-26.

Sarkar, A. & Shah, A. (2018). Learning AWS - Second Edition. 2nd edition. Packt Publishing, p.9.

Sehgal, N.K. & Bhatt, C.P. (2018). Cloud computing: concepts and practices. Cham, Switzerland: Springer. pp.3,43.

Soldani, J., Tamburri, D.A. and Van Den Heuvel, W.-J. (2018). The pains and gains of microservices: A Systematic grey literature review. Journal of Systems and Software, 146, doi: 10.1016/j.jss.2018.09.082. pp.215–232.

Stack Overflow. (2021). Stack Overflow Developer Survey 2021. Stack Overflow. Available at: https://insights.stackoverflow.com/survey/2021 [Accessed 04 Feb 2022]

van Eyk, E., Toader, L., Talluri, S., Versluis, L., Uta, A. and Iosup, A. (2018). Serverless is More: From PaaS to Present Cloud Computing. IEEE Internet Computing, 22(5), doi: 10.1109/MIC.2018.053681358, pp.8–17.

Visma Consulting Oy. (2022). Etusivu – Visma Consulting. www.vismaconsulting.fi. Available at: https://www.vismaconsulting.fi/ [Accessed 06 Feb 2022].

W3C. (2007). SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). Available at: https://www.w3.org/TR/soap12/. [Accessed 26 Feb 2022].

Appendix 1. Translated and simplified questionnaire format

Survey on cloud computing service choices for small and medium-sized mobile and web projects

In the fall of 2021, the standardization of technologies for small and medium-sized projects in the PCS unit was defined. One of the areas mentioned is cloud architecture, on which this study focuses. The aim of the study is to identify the requirements for server solutions implemented in Visma Consulting's PCS unit's mobile and web projects, and to create a “basic toolkit” based on the requirements that can be used in PCS unit projects to select cloud services.

This survey is conducted as part of a thesis. Responses are treated confidentially, and the data cannot be used to identify individual respondents. It will take about 5-10 minutes to complete this survey.

The questionnaire has three parts. The first has questions about basic information, the second about server solutions, and the third about cloud services. If you can't find the right answer, choose the one that works best for you. You can leave open feedback and describe the situation.

The survey is available until May 20, 2022.

If you have any questions, please contact juho.kontiainen@visma.com

1) First part:

Role in the organization:

- Developer
- Architect / Consultant

- Project manager
- UI / UX Consultant
- Other

Work experience:

- 0 years
- under 2 years
- 2-5 years
- 5-10 years
- +10 years

Work experience in web and mobile projects:

- 0 years
- under 2 years
- 2-5 years
- 5-10 years
- +10 years

How many web and mobile projects have you been involved in?

- 0
- 0-5

- 5-10
- 10-25
- +25

Main role in web and mobile projects (multi-choice):

- Front-end
- Back-end
- Architect / Consultant
- Project management
- UI / UX
- Other

2) Back-end part

This section charts the server needs and requirements of the PCS unit for small and medium web and mobile projects.

The aim of this section is to find out which cloud services are best suited to the needs of the server solutions used in the PCS unit's projects.

Container technology has been used in projects I have been involved in (e.g., Docker)

- From 1 to 5, where 1 is never and 5 is almost always

The microservices architecture has been used in the projects I have been involved in.

- From 1 to 5, where 1 is never and 5 is almost always

How important have the following factors been in the projects you have been involved in?

- From 1 to 5, where 1 is not important and 5 is very important

- Factors: Low response time, Support for a large number of concurrent users, Scalability, Configurability of the application environment

Have there been any other important factors in your projects in terms of server solutions? You can write them here. (open-question)

3) Cloud service part

This section charts the background of cloud services used in PCS unit projects.

The aim of the section is to find out which cloud services are currently used in the PCS unit's projects and what are the most important factors for choosing a cloud service.

Computing services with examples:

Virtual Machines: Amazon EC2 and Azure Virtual Machine

Container services: Amazon ECS & EKS, AWS App Runner, AWS Fargate

Function services: AWS Lambda, Azure Functions

PaaS services: AWS Elastic Beanstalk, Azure App Service, Google App Engine

The following types of computing services have been used in the projects in which I have been involved (explanations of the categories above):

- From 1 to 5, where 1 is never and 5 is almost always

- Types of computing services: Virtual Machines (IaaS), Container services (CaaS), Function services (FaaS), PaaS

What factors do you consider most important in choosing cloud services? (Select 3)

- Security, Manageability of the service and its resources, Configurability, Compatibility with other services, Performance

Are there any other important factors that come to mind when choosing a cloud service? You can write them here. (open-question)