

TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Reijo Olavi Komu 178240IASM

# **CLOUD AND SERVER APPLICATIONS**

Master`s thesis

Supervisor: Vladimir Viies

PhD

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Reijo Olavi Komu 178240IASM

# **PILVE JA SERVERI RAKENDUSED**

Magistritöö

Juhendaja: Vladimir Viies

PhD

Tallinn 2019

## **Author's declaration of originality**

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Reijo Olavi Komu

06.05.2019

## Annotatsioon

Antud töö eesmärgiks oli hinnata erinevate aplikatsioonide loomist pilve platvormidele ja serveritele. Tulenevalt kiirest arengust pilve tehnoloogia valdkonnas aina enam uusi rakendusi luuakse skalaarsete lahendustena pilvedes. Sellest sõltumata eksisteerib suur hulk lahendusi tava serveritel. Antud töö käigus analüüsitakse erinevaid serveri pakujate võimalusi ning samuti uuritakse pilve rakenduste loomise võimalusi. Töö käigus valmib kaks prototüüpi, mille puhul üks toetub ainult pilve lahendusele ja teine ainult serverile. Antud lahendusi võrreldakse omavahel ja tuuakse välja loodud lahenduste puudujäägid ja tugevused. Kuna tänapäeva ühiskonnas on pilvede kasutamine suuresti kasvanud on see tõstatanud küsimuse serveri kasutuse üle. Lai ulatuslik pilve levik on põhjustatud lihtsusest luua rakendusi, mis on võimelised toetama suuri arve kliente. Omakorda pakuvad pilved arendajatele uusi ja huvitavaid tehnoloogiad. Eesmärgiks on näidata et serveri lahendused on siiani kasulikud ja pakuvad lahendust probleemidele, mida pilved ei ole võimelised lahendama ning osades olukordades on serveri lahendused paremad, kui pilve.

Töö käigus hinnatakse erinevate pilve ja serverite maksumusi ning nende tugevusi ja nõrkusi. Mõlema prototüübi loomisega näidatakse, et serverite ja pilvede lahenduse vajalikust ja võimalust.

Töö on kirjutatud inglise keeles ja on 55 lk pikk , koosneb 3 peatükist, 24 joonisest ja 7 tabelist.

## **Abstract**

The aim for given thesis is to indicate the difference of building applications on the cloud and server. Even though the rapid expansion of cloud usages has increased, and nowadays newer application are built to scale. Server solution are still quite common. Different types of cloud and server option or analyzed. Cloud prototype is created using existing tools and cloud computing service providers to show the simplicity of constructing applications. Already created server solution is analyzed and compared against cloud. The goal of this is to show that even though cloud solutions are becoming more popular than server solutions, servers still are important and offer solution to problems that clouds cannot.

Given thesis compares the costs of building the application on server and cloud and assess their strengths and weaknesses. Both prototypes prove that some solution are better for cloud when others are for server.

The thesis is in English and is 55 pages long, includes 3 chapters, 24 figures and 7 tables.

## List of abbreviations and terms

SaaS	Software as a service
BaaS	Backend as a service
IaaS	Infrastructure as a service
CaaS	Communication as a service
SaaS	Security as a service
HTTP	Hypertext transfer protocol
API	Application programming interface
VM	Virtual Machine
ECMAScript	ECMA-262 standardized programming language
JavaScript	Object oriented programming language
PHP	Hypertext preprocessor
MySQL	Relational database
NoSQL	Not only Structured query language
JSON	JavaScript object notation
NPM	Node packet manager
AJAX	Asynchronous JavaScript and XML
XML	Extensible markup language
Vue.js	JavaScript framework
SQL	Structured query language

# Table of Contents

- Introduction ..... 11
- 1. Cloud Computing ..... 12
  - 1.1. Definition ..... 12
  - 1.2. Service Models ..... 13
  - 1.3. Deployment Models..... 15
  - 1.4. Cloud computing issues ..... 16
    - 1.4.1. Performance..... 16
    - 1.4.2. Reliability ..... 17
    - 1.4.3. Economics ..... 18
    - 1.4.4. Security..... 19
    - 1.4.5. Legal ..... 21
- 2. Cloud and Server applications..... 22
  - 2.1. Cloud applications ..... 22
  - 2.2. Server applications..... 27
  - 2.3. Hybrid ..... 30
    - 2.3.1. Solution 1..... 30
    - 2.3.2. Solution 2..... 31
    - 2.3.3. Solution 3..... 32
    - 2.3.4. Solution 4..... 33
  - 2.4. Comparison between cloud and server..... 33
- 3. Applications architectures and technologies ..... 35
  - 3.1. Cloud camera app ..... 35
    - 3.1.1. Programming languages ..... 37
    - 3.1.2. Code version management ..... 38

3.1.3. Application design.....	38
3.1.4. Development of code.....	39
3.1.5. Cloud application summary.....	44
3.2. Server based application .....	44
3.2.1. Application design.....	44
3.2.2. Server application summary .....	49
Conclusion .....	50
Kokkuvõte .....	51
List of references .....	52
Appendix 1 – Camera app camera view.....	54
Appendix 2 – Server applications frontpage .....	55

## List of figures

Figure 1 Server and cloud models [5].....	14
Figure 2 Cloud market revenue prediction. [1] .....	18
Figure 3 Cloud application .....	22
Figure 4 Solution 1 design.....	30
Figure 5 Solution 2 .....	31
Figure 6 Solution 3 .....	32
Figure 7 Solution 4 .....	33
Figure 8 Firebase console .....	35
Figure 9 Deployed version handling .....	36
Figure 10 Mode-view-view model .....	37
Figure 11 Packet installing .....	39
Figure 12 Validation of user .....	39
Figure 13 Users list.....	40
Figure 14 Start media capture device .....	41
Figure 15 Capture and Camera Switch.....	41
Figure 16 Firebase security rules.....	42
Figure 17 User panel .....	43
Figure 18 Event listener .....	43
Figure 19 Php validation .....	45
Figure 20 Database request.....	46
Figure 21 Session handling .....	47
Figure 22 SQL Injectable query.....	47
Figure 23 Timing attack .....	48
Figure 24 Database model [25] .....	48

## List of tables

Table 1 Availability of service [10].....	17
Table 2 Cloud Pricings for VMs [11] [12] [13] [14].....	23
Table 3 Zone virtual machine pricing [20].....	27
Table 4 Zone server rent pricing [21].....	28
Table 5 Veebimajutus.ee pricing [22] .....	28
Table 6 Trumpit server rent [23] .....	29
Table 7 Comparison between server and cloud [12] [20] [23].....	34

## Introduction

In the modern society where, new technology emerges the cloud computing becomes more accessible and easier to use. This raises the difficult question should all application be built surely on the cloud and stop using private servers.

This sort of rapid development of technology has enabled the usage of cloud computing. Convenience of the cloud has made designing new systems with the possibility of scalability reachable to everyone. Cloud solutions also have limited the time that is required to handle the user Data, Storage and Security and software development.

The market share of cloud computing is expanding on a daily basis. Recent studies have predicted that the infrastructure as a service (IAAS) is expected to reach \$83.5 billion by 2021. The software as a service (SAAS) has been forecasted to grow up to \$117.1 billion by 2021. Platform as a service is also expected to grow up to \$27.3 Billion. Nowadays the biggest companies in the market are Amazon, Microsoft and Google holding almost 80% of total market share [1].

This rapid expansion of cloud cause has been caused by the fact that the cost of using cloud service providers has become cheaper. Also, the need to manage every aspect of the system isn't always needed and that why enterprises have started to move from private server to cloud. Cloud enables to handle complex problems like scaling with limited effort.

# 1. Cloud Computing

In the recent years, cloud computing popularity and usages have increased rapidly. New application like Netflix and Uber all have been built using cloud solutions. Implementing their designs on the cloud has enabled them to rapidly expand and reach large audience. This why knowing and understanding cloud has become more relevant than ever before.

## 1.1. Definition

The definition of the cloud computing according NIST is following: “Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” [2].

Cloud itself consists of five characteristics, three service models and four deployment models according to the National Institute of Standards and Technology (NIST) [2].

Five key characteristics according to NIST are following:

- **On-demand self-service:**

This enables the consumer to get access to required server time, network storage capabilities without ever needing any human interaction. The whole process is automatically resolved by the cloud itself [2].

- **Broad network access:**

Given services and resources of cloud are accessible to any device (e.g., tablets, personal computers, phones, laptops) that is capable of connecting to the internet [2].

- **Resource pooling:**

Pooling offers resources of multiple hardware devices by interlinking those devices together using hardware virtualization. Virtualization enables to pool computing resources together into one enormous virtual resource which enables to improve the efficiency of given cloud system. This offers to the customer the appearance of one single great resource without ever feeling any sharing of the physical infrastructure [3].

- **Rapid elasticity:**

In the case when the demand for resources rapidly increases or lowers. The system automatically is capable to react to given demand changes. For the customer the supply of the resources may seem infinite [3].

- **Measured Service:**

Providing elastic and measurable service requires the Cloud computing service provider to be able to optimize underlying infrastructure and give the consumer needed information of their cloud service usage. With key feature being the transparency [3].

## **1.2. Service Models**

According to NIST there exists three main models which most of the cloud services can be classified:

- Cloud Software as a Service (SaaS). Given service model offer clients some sort of a service which uses cloud. Given service model customers have no interaction between cloud infrastructure or clouds platform. Given model focuses on the application layer which customers can use interact through interface. Google Drive is one many SaaS type of service. Given service users have no knowledge about used infrastructure or software. Consumers are able to configured given application in certain ways (Figure 1) [3].
- Cloud Platform as a Service (PaaS). This service model layer offers to developer prepared environments where necessary tools and operating system are installed.

Consumers are able to modify tools or even remove them from existing environment, but given infrastructure and hardware are operated, controlled and managed by the owner of the infrastructure. Consumers are able to develop their own software and use given infrastructure environments to execute or deploy their software (Figure 1) [3].

- Cloud Infrastructure as a Service (IaaS). Consumer has the possibility to use underlying infrastructure to run any sort of software from application to down to the operating system. Service provider still owns the hardware and infrastructure and solves any hardware relate problems and keeps cloud infrastructure running. Often consumer is even offered some sort of security options which can added or modified in IaaS type of service (Figure 1) [4].

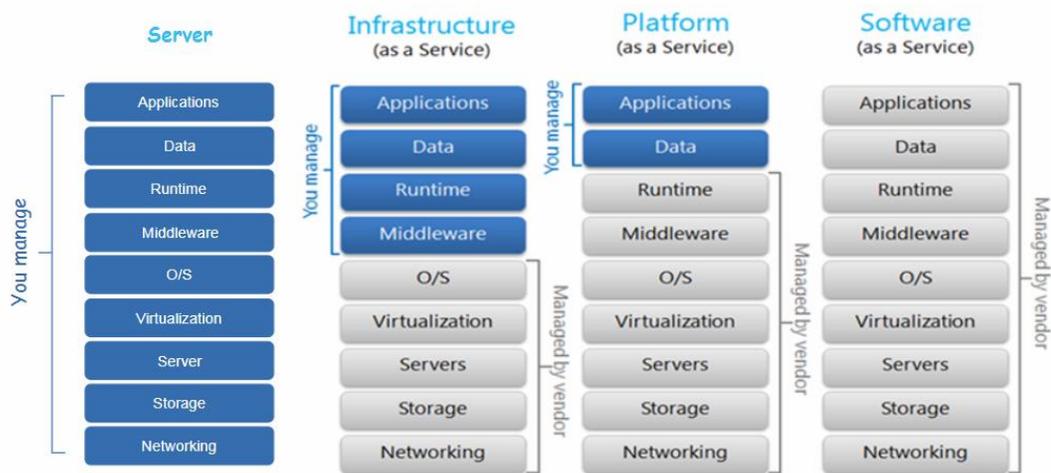


Figure 1 Server and cloud models [5]

Other type of cloud service models that exist which can be derivatives from other cloud models:

- Cloud Backend as a service (BaaS) often referred to also as “mobile backend as a service” (MBaaS). Is a new type of cloud model which offers application backend as a service. Often APIs (Application programming interface) are offered to be able to connect applications with cloud. Also, different types of features are offered to developers: user management, push notifications, customer analytics, social

integration, short message service, hosting. Some backend as a service providers make their backend accessible through software development kits (SDKs) others prefer application programming interfaces (APIs) [6].

- Communication as a Service (CaaS). Given models offers to customer different types of communication features: videoconference, instant messaging, Voice over IP. Main target of CaaS is small and medium business which otherwise would have difficulties to provide given solution which comes from running expensive hardware and software [7].
- Security as a Service (SaaS). Given model sources to customers the capability of handling and managing cyber security. This model is built on Software as a Service (SaaS) model. This model eliminates from the customer the need to ever set up and buying expensive antivirus software licenses [8].

### **1.3. Deployment Models**

Deployment models give insight the way cloud services are accessible and deployed from the point of provisioning location and companies structure. Four types of deployment models are mainly used: private, public, community and hybrid:

- Private cloud - Cloud infrastructure is available to single legal entity or organization. Often given clouds are managed and operated by the same organization that owns the infrastructure. Location of given cloud infrastructure can be in the same location as the company but is not mandatory [2].
- Community cloud - Given infrastructure is meant to be used by a specific community whose values, goals and concerns match. Management of given infrastructure can be done by the community or some by a third-party service or even some sort mixture of them.[2].
- Public cloud - Given cloud service infrastructure is meant to be used by general public. Infrastructure can be owned by different types of organization which include

academic, government or business. Given cloud infrastructure exist on the premise of the cloud owner [2].

- Hybrid cloud - The cloud infrastructure involves two or more types of cloud deployment models. Combined clouds remain as unique entities which work has been interlinked using technology to offer higher portability [2].

## **1.4. Cloud computing issues**

Often times cloud computing does not offer correct services for most IT firms. Not every type of application or solutions is capable of being deployed to cloud solution. From the complexity of cloud system and their underlying technology many types of problems may emerge. Its highly crucial when planning to implement any sort of application on cloud these issues are taken into consideration. Otherwise false premises and understanding of clouds might rise [2].

### **1.4.1. Performance**

Needs of application may vary which requires system performance to be highly adaptable. Performance is the key aspect of the cloud which requires an outstanding resource management. Otherwise clouds efficiency to scale could be lost [3].

### **Latency**

Highly problematic part of cloud is the latency. Often customers who build application on the cloud need the application to be available all around the world. In the case where given cloud service provider does not have any infrastructure in given region latency becomes huge problem. To lower any sort of latency issues web application optimization technologies are often used which enables to lower latency for users. Using software does not always resolve the latency issues when working with clouds. Latency issues are also caused by the pooling of the resources where system has to spread the workload between different client [9].

### **Scalability**

Having access to dynamically scaling computing power requires to developers to write scalable code. This requires the developers to adapt a new way of programming where

parallel processing is taken to consideration. This also causes the need for application to be redesigned to be able to get the full potential of dynamic computing power that is accessible when needed. Often legacy software requires rebuilding to be able to scale [9].

### 1.4.2. Reliability

Cloud reliability shows how failure resilient given provider is in probability. Measuring any sort of reliability when dealing with large system with many interlinked components is complicated. Clouds are highly dependent on the existents of internet to offer customers the availability of data. It is easier to calculate availability of data than the reliability of system. Availability shows in percentage how available service is, where measured part is the time data is not available (Table 1) .Network dependence is not only related to cloud provider but to any service that requires internet. Often natural causes (e.g., earthquakes, storms, etc.) can be the cause providers reliability is affected but its known to happen that automation errors also cause failures in the cloud systems [2].

*Table 1 Availability of service [10]*

Availability	Down time per year	Downtime per month	Downtime per week	Downtime per day
90%	36.53 days	73.05 h	16.80 h	2.40 h
99%	3.65 days	7.31 h	1.68 h	14.40 min
99.9 %	8.77h	43.83 min	10.08 min	1.44 min
99.99%	52.60 min	4.38 min	1.01 min	8. 64 sec
99.999%	5.26 mi	26.30 sec	6. 05 sec	0.864 sec

### 1.4.3. Economics

Global market of cloud providers is expanding on a yearly basis and is targeted to reach 200-billion-dollar revenue by year 2021 (Figure 2).

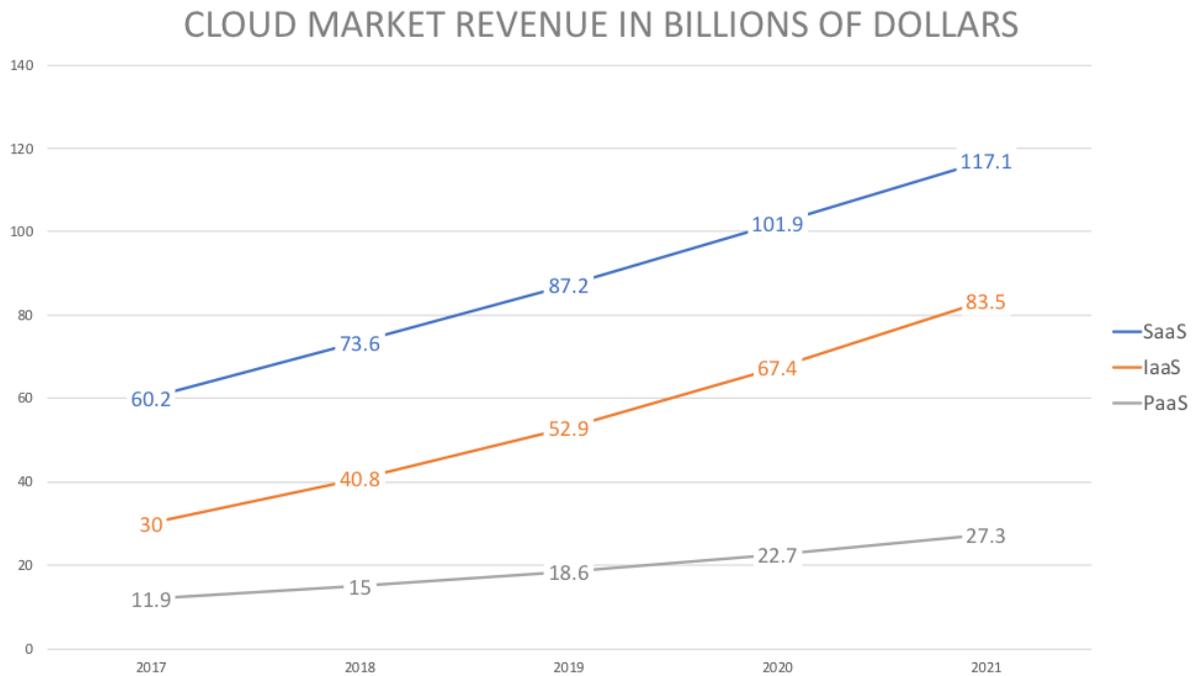


Figure 2 Cloud market revenue prediction. [1]

Even though cloud computing promotes elasticity and promises cost reduction. Economic risks have to be taken to consideration. Highest risks when dealing with cloud economics is business continuity. There always exists possibility, that the cloud provider could shut down. This requires from customers to always be up to date with the financial status of their cloud provider otherwise they themselves could be in danger of going out of business. Using hybrid solutions also helps mitigate the risk in the case the cloud provider should go bankrupt. Hybrid solution also cause extra cost to consumers from needing to run their own hardware and paying for cloud services [2].

#### **1.4.4. Security**

Security is the most complicated topic when dealing with clouds. There exist different types security issues which could be classified as following:

- Infrastructure and host - Given threats involve the whole clouds infrastructure. [9]
- Service provider – Threats that oriented to the customer who wishes to use cloud service [9]
- Generic – These types of threats can affect both infrastructure and provider. [9]

#### **Physical access to cloud**

In the case where unauthorized users can get access to facilities of cloud systems. Given unauthorized access threatens every part of the cloud: Hardware could be lost which is used to offer cloud service, Clients data theft which is highest risks. This why its highly important that cloud providers would limit any sort of human interaction between physical cloud server [9].

#### **Cryptography**

For securing data cloud providers should always use the latest and strongest cryptography that exist. Otherwise in the case of data breach data could be easily opened. Its highly important that any implementation of encryption and cryptography is validated to work correctly. Mishandling encryption could ruin the raw data or put client's at high risk [9].

#### **Weak isolation**

Infrastructure provider in cloud must encapsulate clients in Virtual Machines. Badly configured Virtual Machine could enable client's software and data to be accessible to other's running in them same virtual machine. Great deal of consideration and care has to be made when dealing with encapsulation. This problem becomes more relevant when dealing with the public cloud [9].

## **Data processing**

Handling of data is the most critical security issue in cloud. Customers never know if and how well their data is protected. Customers only can rely on the fact that it's in the best interests of cloud provider to take every step necessary to protect their data. Cloud service providers also have issues when dealing with user data. Legal problems from provenance of the data could arise in the case where data has been stolen or contains illegal content. Cloud providers should offer methods to customers to get insight to the way their data is being protected and handled [2].

## **Economic denial of Service**

Economic denial of service (EDoS) is the newest kind of threat that has risen from cloud environments. Different scenarios of this type of attack exist. Identity theft – Criminal gets access to customers account uses their cloud resources for their personal gain. Same time customer is being billed for things that they didn't use. Attacker also take advantage of stolen account credibility to pressure their customers for money. Problem could be even greater if customer has set no limits to spending [9].

## **Browsers**

Often the most problems causing factor in the cloud industry is the browser security level. Customer mainly use their personal computers to interact with cloud to access through different sorts of browsers. Browser haven't been generally developed to be safe to interact directly with clouds. Security breaches in the browsers could be high risk to the whole infrastructure of cloud [9].

#### **1.4.5. Legal**

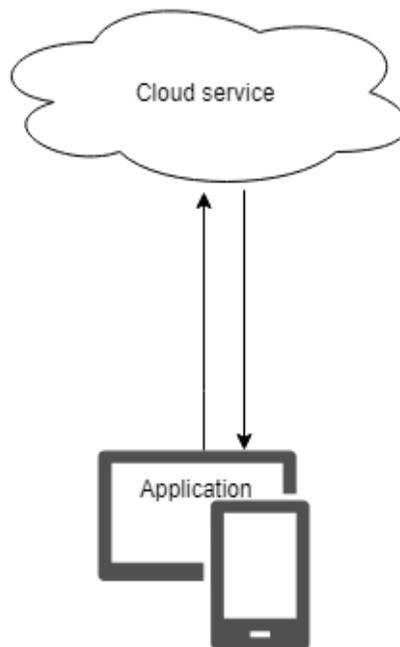
Consumer of cloud services are legally obligate to secure customers information. Deployed applications on cloud infrastructures cannot be verified to have all the security guaranteed by the provider. Lack of visibility for consumers from the cloud provider side may cause difficulties and cause legal issues. Third party audits can be used to verify that cloud providers systems are compliant with set promises, but this does not always solve the problem. Without having enough guarantee from cloud provider certain application and data cannot be stored in cloud [2].

## 2. Cloud and Server applications

This chapter analyzes the different types of cloud service providers and server providers when constructing applications. Also Compare cloud and server pros and cons which they offer to customers.

### 2.1. Cloud applications

Creating application is possible using different types of cloud services. The main types that offer the possibility to develop applications are PaaS and IaaS. IaaS gives the consumer the possibility to manage the Cloud infrastructure. Managing the cloud infrastructure might be needed in the case if the user wants to create their own type of features and automation. IaaS enables the developer to manage, create, install and monitor the infrastructure the way they see fit. In the case of IaaS, the way the consumer is billed differs. In some cases, the developer is billed by the time of CPU (Central processing unit), Network, number of IP connections to cloud. Some IaaS providers also bill the developer by the number of servers in the cloud they use. In general, the pure cloud applications are directly designed to communicate with cloud service provider (Figure 3).



*Figure 3 Cloud application*

Downside using the IaaS structure for developing cloud applications is it requires more knowledge. The developer is required to set up the virtual machine's software, manage networking and storage and handle all the security. Implementing the system takes more time from the need to manage more difficult aspects. The cost of the IaaS setup varies depending the way the provider charges. Given table shows most popular cloud provider cheapest options (Table 2).

*Table 2 Cloud Pricings for VMs [11] [12] [13] [14]*

Provider	Memory (GB)	VCPUs	Storage	Pricing
Digital Ocean	1	1	25GB	\$0.007/hr
Digital Ocean	2	1	50GB	\$0.015/hr
Amazon EZ2	0.5	2	EBS only	\$0.0105/hr
Amazon EZ2	1	2	EBS only	\$0.021/hr
Azure Virtual	1	1	\$0.0006/per transaction	\$0.008/hr
Azure Virtual	4	2	\$0.0006/per transaction	\$0.085/hr
Google Compute	3.75	1	30GB	\$0.0612/hr
Google Compute	7.5	2	30GB	\$0.1124/hr

In all the cloud services there exists many different options to match any customer need as best as possible. Most often pricing system in cloud providers is not transparent and can cause unexpected extra costs. Developers can be billed by the amount of times they query the

database which can become quite expensive or when given hardware limit is reached new cloud server is used which cost is higher.

Some cloud providers also offer free tier for developers which enables them test and build application without worrying about costs at beginning. Free tiers usually don't include all the features that cloud offers.

Cloud application can be developed also by using different PaaS providers. Using PaaS eliminates the need to manage the virtual servers which lowers time needed to set up the system. The cost of running the system depends on from the service provider. Implementing designs on PaaS can become expensive when pricing is calculated the time used on CPU. This type of pricing requires from developers highly efficient code. The PaaS advantage over IaaS is that the infrastructure of the cloud is not required to be monitored. The infrastructure is monitored by service provider. PaaS architecture enables companies to use microservice architectures and build and deploy code from different types frameworks and programming languages. Largest and most popular PaaS providers are:

- Google APP engine

Google app engine is offers platform to manage web frameworks and cloud computing and host application in data centers. To avoid problems with client's code Google Data centers, use sandboxing where application is stored in a shell inside of the servers. Google App engine removes the need to manage and configurate servers and enables developers to focus on building and deploying applications. Google app engine also offers a free amount of resources that developers can use for building and testing their applications. Everything that exceeds given free limit will be billed by only what have you used [15].

- Azure Web App engine

Azure Web app engine offers the capability to build and deploy mission critical web applications that scale with the needs. Azure web app engine also offers vast number of databases and features for developing. Azure Web app engine is capable of running multiple programming languages: .Net, .NET core, Java, Ruby, Node.js, Php or Python. Also, different types of environments are supported Linux and Windows [16].

- Heroku

Heroku is one of the first cloud platforms and has been developed since 2007. Heroku now supports different types of programming languages: JAVA, Clojure, Scala, Node.js, PHP, Python, and GO. Heroku runs developer's application inside dynos. Dynos are fully managed runtime environments which enables developers to deploy their code written in any supported language. Developer are not required to in interfere thanks to complicated automation provided by Heroku [17].

- Amazon Web Services

Amazon Web services (AWS) is the biggest cloud resource supplier in the world. Owning up to 34% of total cloud market. AWS offers wide range of services: computing, networking, database, application services, deployment and other tools. AWS billing differs greatly depending what type of plan is used. Most commonly application owner is billed based on the amount CPU, GPU and Storage usage. AWS services are accessible all around the world [18].

- Oracle Java Cloud Service

Oracle cloud is cloud computing service that is offered by Oracle Corporation. Oracle Cloud offers many different services Data managements, application development, Integration, Business analytics, Security. For application development different functionality is offered some which lower the need of code development and make releasing an application easier. Oracle cloud bills mainly by looking at the amount hours of processing was used. Different type of database is supported MySQL, NoSQL [19].

Even though PaaS offers higher level development than IaaS and moves complicated infrastructures handling and monitoring to service providers. Even higher type of service exists nowadays. Backend as a service (BaaS) offers functionality for application without ever needing to write any backend code. Making rapid development possible.

- Cloudboost

Cloudboost is a backend as a service company from India. Cloudboost offers different types of products for software developers building mobile or web applications. The main features of Cloudboost is the database which offers an API which developers can use to store, search, query and access real-time data. Cloudboost offers NoSQL as backend database [20].

- Firebase

Firebase is BAAS platform that is targeted to web and mobile applications. Firebase offers different types of features:

- Firebase Auth, which enables developers to use social media logins,
- Real-time database this service enables to synchronize all data across all clients and store in firebase cloud.
- Firebase hosting enables to store dynamical web application in firebase cloud and also use Firebase functions to execute custom functions.
- ML kit offers to develop machine learning system. It also supports TensorFlow models to be uploaded.

Firebase platform is owned by Google which gives the platform more unique features about system application stability and analytics. Firebase also offers a free developer tier that enables to build application [21].

- Parse

Parse was a mobile backend as a service provider which was acquired by Facebook. In 2017 given service was shutdown. The platform was made open source to enable developers to migrate to different cloud providers. Parse platform is community operate platform with minimal functionality.

In general, the cloud biggest downside is the security of data. In every given cloud service type: IaaS, PaaS, BaaS the physical servers are in unknown location with unknown people with access to server. Given the risk of data security the developer or the product owner has to keep in the mind that data is not in their control. Some cloud service providers can be

combined with outer networks where data is stored in private client servers. In the case of the Firebase the developer is not able to download or export the user data that has been stored. This offers both security and risk. In the case when owner loses control over the account user data cannot be stolen.

## 2.2. Server applications

Client-server model enables the spreading of the workload between service providers servers. On most cases this type of communication achieved using computer network. This type of model is also most frequently implemented. Using this model has downside. It requires complicated infrastructure which is viable on small scale. When the load on the server increases more infrastructure is required. In most cases adding more infrastructure is time consuming and expensive. In large scale this type model becomes even more expensive and complicated. Large scale server often is required to be housed in special data centers to solve power consumption and heating problems. Handling all the server-side issues are not always wanted. Nowadays there exists services that offer servers in data center to be rented for monthly subscription or part of server can be rented in the case of virtual server rent:

- Zone.ee

Zone offers two types of server renting possibilities. They offer customers a virtual machine that can be rented in the server to be used for web application and customers are billed by monthly basis (Table 3) Also, they offer the possibility to rent a whole server in their server park which also has a constant cost (Table 4).

*Table 3 Zone virtual machine pricing [22]*

Type	Storage (GB)	Mails (GB)	Mongo DB	Node.js	SSH	Own IP	Pricing
Packet 1	128	8	No	Yes	Yes	No	6,66€
Packet 2	256	12	Yes	Yes	Yes	No	12,66€
Packet 3	512	16	Yes	Yes	Yes	Yes	21,60€

Table 4 Zone server rent pricing [23]

Type	Storage (GB)	RAM (GB)	CPU	Pricing
Packet 1	2x SSD 480GB	32	Intel Xeon E-2134 4C/8T 3.50 GHz	219,48€
Packet 2	4x SSD 480GB	96	Intel Xeon Silver 4114 10C/20T 2.20 GHz	495,55€

- Veebimajutus.ee

Veebimajutus.ee offers the possibility to rent a virtual machine from their server park to run your own small website or application. Comparison between important value which Veebimajutus.ee offers (Table 5).

Table 5 Veebimajutus.ee pricing [24]

Type	Storage (GB)	Mails (GB)	SSH	Mongo DB	Own IP	Pricing
Standard	150	10	Yes	No	No	5,50€
Pluss	250	15	Yes	No	Yes	10.75€

In the case where user wants to have physical access to their server and the location of server determined by them given service exists:

- Trumpit

They offer to clients the possibility to rent a physical server that they will set up at the customers office or home. This type of service enables to be in control of the infrastructure and data by choosing the location where given data is stored.

Owning and managing infrastructure is only viable if the customer base is certain and rapid load on the system is not expected. Owning servers makes predicting the cost of running the application easy because only certain rent has to be paid for owning the servers (Table 6).

*Table 6 Trumpit server rent [25]*

Type	Storage	Ram (GB)	CPU	RAID	Pricing
Small	2TB HDD	8	Intel XEON	YES	45€ / month
Medium	4TB HDD	16	Intel XEON	YES	65€ / month
High	4TB HDD 512GB SSD	16	Intel XEON	YES	95€ / month

In the case of renting a virtual machine given security risks still exist where customer must trust the service provider. To be able to fully secure the data physical server must be rented which can be quite expensive. Ranging from 45€ up to 500€ per month. Given hardware that those server offer can handle large number of users.

### 2.3. Hybrid

Hybrid solutions with cloud and server are required when merging into one solution is not viable or would need unnecessary refactoring or reworking of the applications code. The type of hybrid solutions that can be used depends from the needs of given system. Hybrid solutions could also have different types of clouds involved like community and private clouds. Most common reason for building hybrid solutions is the need for scalability. Any type of hybrid solution generally is more expensive non hybrid solution. Higher expense comes from the need to pay for both cloud usage and already existing infrastructure.

#### 2.3.1. Solution 1

Older application that already have been built using servers often require developers to implement hybrid solution for new features. Using cloud and server same time serves two main purposes (Figure 4).

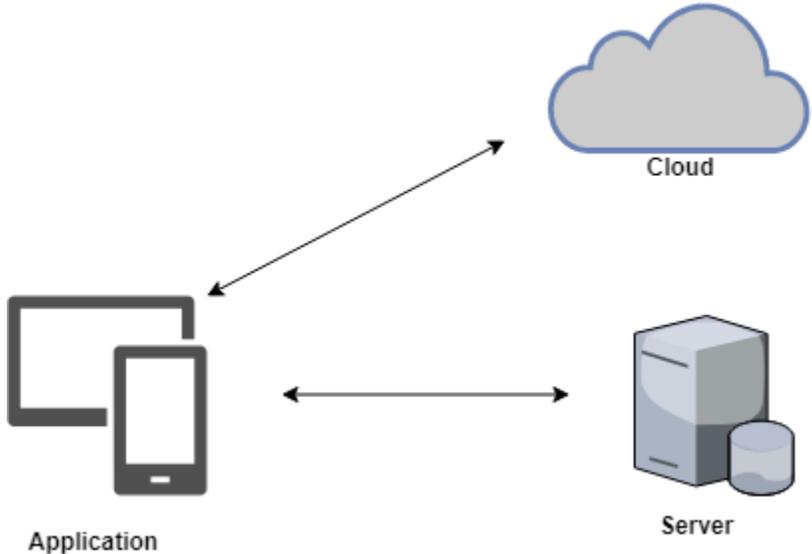
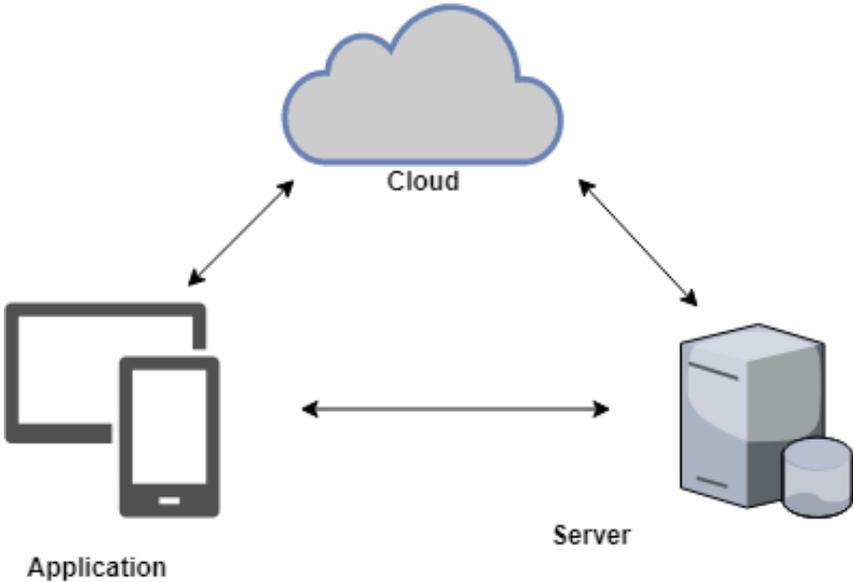


Figure 4 Solution 1 design

Allowing to integrate new features to system or enable slow integration to the cloud. One of the features that could be built on cloud is chat. The main problems when moving to cloud are the database movement and the code efficiency. In the integration process the database model must be convertible to given database format that cloud supports if that is not possible and hybrid solution has to be built. Implementing this type of model eliminates the need to move the already existing database to cloud provider (Figure 4). Only new types of features and upgrades could be moved to the cloud providers servers. This would enable to increase the availability of the application and offer new types of features for customer. Also keeping all the sensitive data stored in private servers.

**2.3.2. Solution 2**

Given solution involves all way communication where all the counterparts have access to send and receive information from each other (Figure 5).



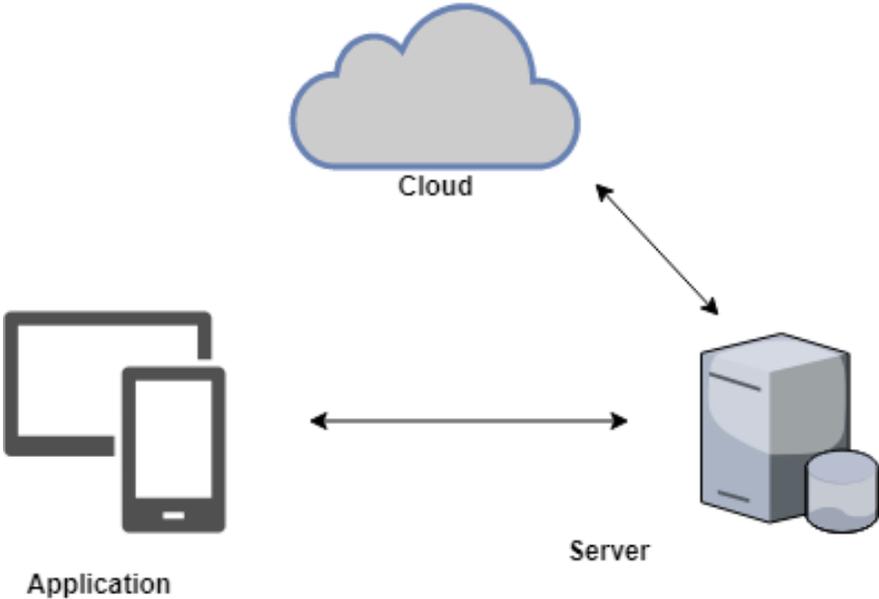
*Figure 5 Solution 2*

Multi way communication could have many implications. Implementing the storage part on the cloud and processing on the server enables the enterprise to cut down the cost of using cloud and still have a use for existing servers. Only in the case when the users amount suddenly increases the cloud computing is used. Non system critical information could be

stored in the cloud and critical information of operation in private servers. Less sensitive information exists in public clouds the safer the end users are and less vulnerable the client is. Using public cloud may end with organization losing the access to the physical data where the information has been stored. Building the application to have all ways communication can be challenging and time consuming. Given architecture is targeted to application that require high availability.

**2.3.3. Solution 3**

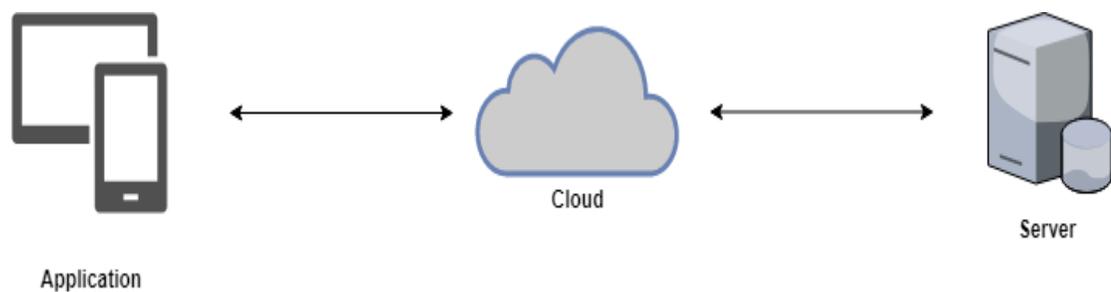
In some cases, the application does not communicate with the cloud directly (Figure 6). Applying this architecture comes from the server being critical component of system. This type architecture is required when dealing with classified documents or expansion of the system is required without removing existing infrastructure. Keeping the existing physical hardware and using the cloud to give system unlimited scalability. In many cases cloud offers unique features that are complicated to solve or need excess time to solve and involving cloud infrastructure is easier and faster.



*Figure 6 Solution 3*

#### 2.3.4. Solution 4

Application to cloud and cloud to server model enables to develop the system parts separately. (Figure 7). Basic functionality of the system could be built on the cloud that enables the application to be accessible around the globe. Servers could be used only to store databases to eliminate the data security risks that come from building application on the cloud. Applying this model to older system is complicated it requires the capability to remove code from private servers and merge the functionality to cloud.



*Figure 7 Solution 4*

#### 2.4. Comparison between cloud and server

Cloud and server both offer unique features when building applications onto them. When comparing cloud and server offers from hardware and pricing point the storage size differs a lot (Table 7). In the case where user application usage a lot of storage room cloud service becomes a problem. In cloud solution when maximum limit is reached extra cost are charged. Exact rate that is charged average to few cents per GB. The storage difference is even greater when dealing with physical server.

Table 7 Comparison between server and cloud [12] [22] [25]

Provider	type	Storage (GB)	RAM (GB)	CPUs	Pricing (month)
Digital Ocean	Cloud	25	1	1	4.48€
Digital Ocean	Cloud	50	2	1	9.6€
Zone	Virtual Machine	128	0.512	-	6,66€
Zone	Virtual Machine	256	0.768	-	12,66€
Trumpit	Server	2000	8	-	45€
Trumpit	Server	4000	16	-	65€

Cloud servers are usually slower compared to physical server. The speed difference comes from the high number of layers that can exist between the software and hardware. This efficiency difference becomes even greater when application user base increases and more computing power is required. To meet the requirements even more small nodes are required to be running in cloud when one physical server could handle the workload. In the long run given difference can cause cloud application to become more expensive than server-based solution.

### 3. Applications architectures and technologies

Given chapter analyzes constructing applications using different technologies and architectures to use in different projects. Given applications that are analyzed are base on cloud or server. Implication differences are shown and analyzed between different types of solutions.

#### 3.1. Cloud camera app

Given prototype has been built on the Firebase which uses Google Cloud as its cloud service provider. The choice was made because Firebase eliminates the need for backend code and given platform also offer free developer tier for developing the application. The app uses built in real-time database in firebase which is a NoSQL database. Firebase classifies as backend as a service (BaaS) cloud. Its main goal is to offer simple way to build backend for applications. Firebase offers cross platform support which includes following platforms: iOS, Android and Web applications. Firebase also offers cloud hosting which enables to deploy the whole project in the cloud to make it accessible globally. To manage application and have an overview of what is happening, Firebase offers an application console with different types of tools (Figure 8).

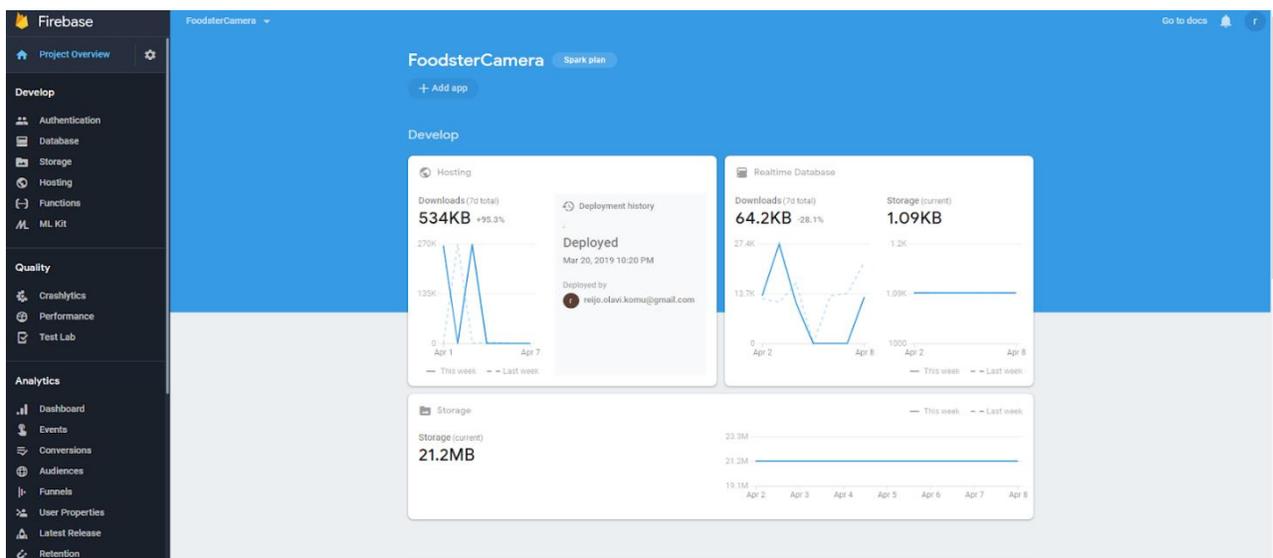


Figure 8 Firebase console

Given tools that are offered are ML kit, Hosting, Functions, Database, Storage, Authentication etc. Projects built on firebase are easily scalable. Developer isn't required to do anything extra to the application to have access for more resources. Inside Firebase platform two types of databases exist: Realtime database is NoSQL database which has json format which can be managed from the Firebase console. Firebase also offers Cloud Firestore storing which is NoSQL database with characteristics of MongoDB. The two main differences between real-time database and cloud Firestore are the way service is billed for the developer. Realtime database bills by the number of users connected to the database. Cloud Firestore bills based on the Read and Writes. The second difference comes from the scalability Cloud Firestore was designed to handle very high loads of read and writes. Realtime database main purpose was to be able to sync all the data across the users.

Firebase console also offers version handling which enables the developer to roll back to previous version in case software bugs or instability (Figure 9).

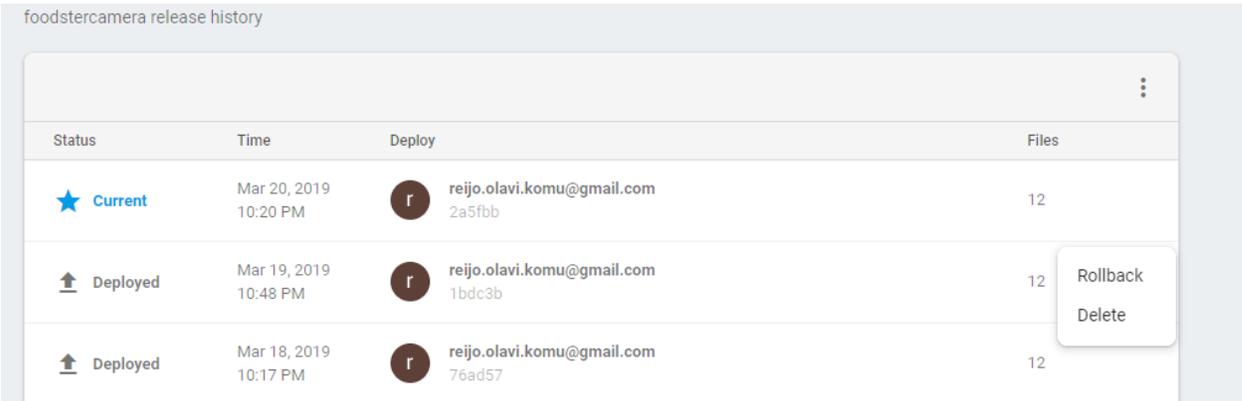


Figure 9 Deployed version handling

### 3.1.1. Programming languages

To make the application easily developable and manageable a JavaScript framework Vue.js was used. Vue enables to build single page applications with minimal effort. For data processing in the backend Firebase cloud service was used.

Vue.js offers to developers the possibility of building dynamically rendered webpage applications. Vue.js supports vast number of different libraries and package which enable simple integration of features.

The main goal of using Vue.js was to eliminate the need for complicated JavaScript and enable clean code and simple solution development. Vue in its core is influenced by the Model-view-viewmodel (MVVM) pattern (Figure 10). MVVM model separates the back-end logic from front end logic.

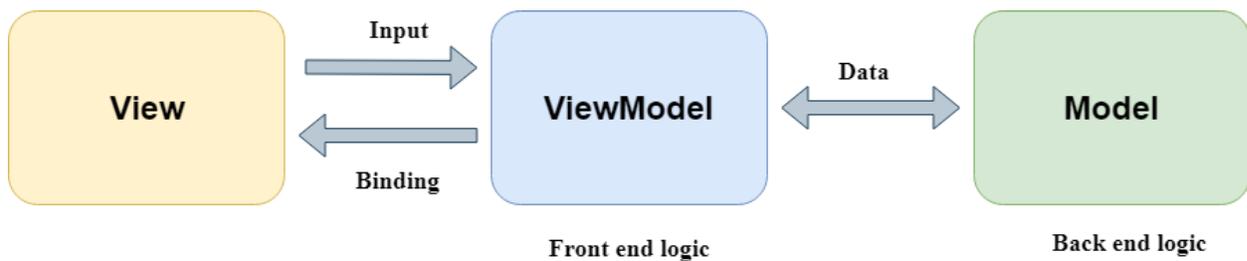


Figure 10 Mode-view-view model

Every new application in VUE.js first starts with creating a new instance. Every instance of Vue has options object which can be shared between different instances by passing it along. Generally, Vue.js application is made of root Vue instance where every other Vue component is organized into a nested tree [26].

### **3.1.2. Code version management**

Code organization and management is important. Without managing code and having any sort of version control is complicated and bad practice. To simplify the development process and streamline code deployment, GIT was used. Git itself is open source platform for version management, different types of providers of git exist. For given project Bitbucket was used. Bitbucket offers private repositories for free which is not available in most Gits platforms.

### **3.1.3. Application design**

In Vue.js the software is split into components, every component is only render for user when the given component is called. Every component has a template which contains basic HTML (Hypertext markup language) to design how the page looks. Also Vue.js offers conditional rendering where parts of content are only render for user when needed inside the component.

The whole application exists only as frontend solution with simple user interface (Appendix 1) Communication with the cloud is achieved in the root of the application where, connection to firebase is achieved using firebase library

Building the application to be single page application enables the content to be rendered faster and does not require to reload the full page every time the small part of the content changes. Vue.js in its core does not have routing handled. Routing enables to block unauthorized user to move to pages where they don't have access and routing enables cleaner way to move between components inside the application.

It is recommended to use ESCMAScript 6 which requires to install package called babel. In 2019 there isn't a web browser that can directly execute ESCMAScript 6, and it must be recompiled to older version. Using ECMAScript enables to make the code more readable and easily understandable. To compile modules and put them together into compressed build Webpack library is used. This comes from the need to limit the size of the project when deployed. Getting all the packages that are required to develop the application node package manager ( npm ) was used.(Figure 11).

```
LapseLaps@LapseLaps-PC MINGW64 ~/Desktop/NewApp/nodeServer/foodstercameraserver
(master)
$ npm install --save firebase
```

Figure 11 Packet installing

By calling npm install and telling it to save given library is installed to project which then can be used by the developer.

### 3.1.4. Development of code

Design of the code used Agile methodologies ideas to build features and when finished then to deploy them. The way project has been designed there exist no backend code. Firstly, we analyze how account creation has been created using firebase and Vue.js. We need to create a form that can accept “username”, “password”, “password Repeat” and “email”. Then we can write simple methods that can validate all the fields of data and give instant feedback to user without needing to send any requests to server. If the code passes all the validations data is sent to cloud to be validated. (Figure 12).

```
register:function(e){

    if(this.accountInfo.email == null)
        return this.errors = 'Email Required'
    if(this.accountInfo.password == null)
        return this.errors = 'Password Required'
    if(this.accountInfo.passwordRepeat == null)
        return this.errors = 'Repeat Your Password Required'
    if(this.accountInfo.email.length < 10)
        return this.errors = 'Email too short'
    if(this.accountInfo.password.length < 10)
        return this.errors = 'Password too short'
    if(this.accountInfo.password != this.accountInfo.passwordRepeat)
        return this.errors = 'Password mismatch'
    firebase.auth().createUserWithEmailAndPassword(this.accountInfo.email,
    this.accountInfo.password).then(user =>{
        this.$router.go({path: this.$router.path })
    },err =>{
        this.errors = err.message;
    });
}
```

Figure 12 Validation of user

To deliver the data we use JavaScript promise object which on completion of its asynchronous operation returns value. In the case something is wrong error will be given which will be shown to user. Once user creation has succeeded account will be logged in. For user comfort purposes email confirmation was not implemented but using firebase it is as easy as creating an authentication. Simplicity come from firebase where it automatically verifies all the data that was sent when user tried to register. Biggest downside of using firebase authentication is that registered user emails cannot be exported. This limits often companies' opportunities to send advertisement. In this case some parts might be needed to be embedded to server to offer more functionality (Figure 13).

Identifier	Providers	Created	Signed In	User UID ↑
reijo12@hotmail.ee	✉	Mar 18, 2019	Mar 18, 2019	S5qOCrsiOtNOUSSLdG08IC9skSD3
reijo.olavi.komu@gmail.com	✉	Mar 18, 2019	Apr 9, 2019	XslyEFIpcgRzsElwKNdnn8TLVGh2

*Figure 13 Users list*

Inside the application users can take picture or view pictures that have been taken. First application searches if any sort capturing video capturing device exists, if given device exists media stream will be started (Figure 14). Also, to use camera modern browser require applications to ask permission from users.

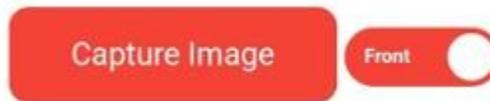
```

mediaDeviceStart(){
  if('mediaDevices' in navigator && 'getUserMedia' in navigator.mediaDevices){
    navigator.mediaDevices.getUserMedia({ video: this.cameraMode})
    .then(mediaStream => {
      this.$refs.video.srcObject = mediaStream
      this.$refs.video.play()
      this.dataStream = mediaStream
    }).catch(error => console.error('getUserMedia() error:', error))
  }
}

```

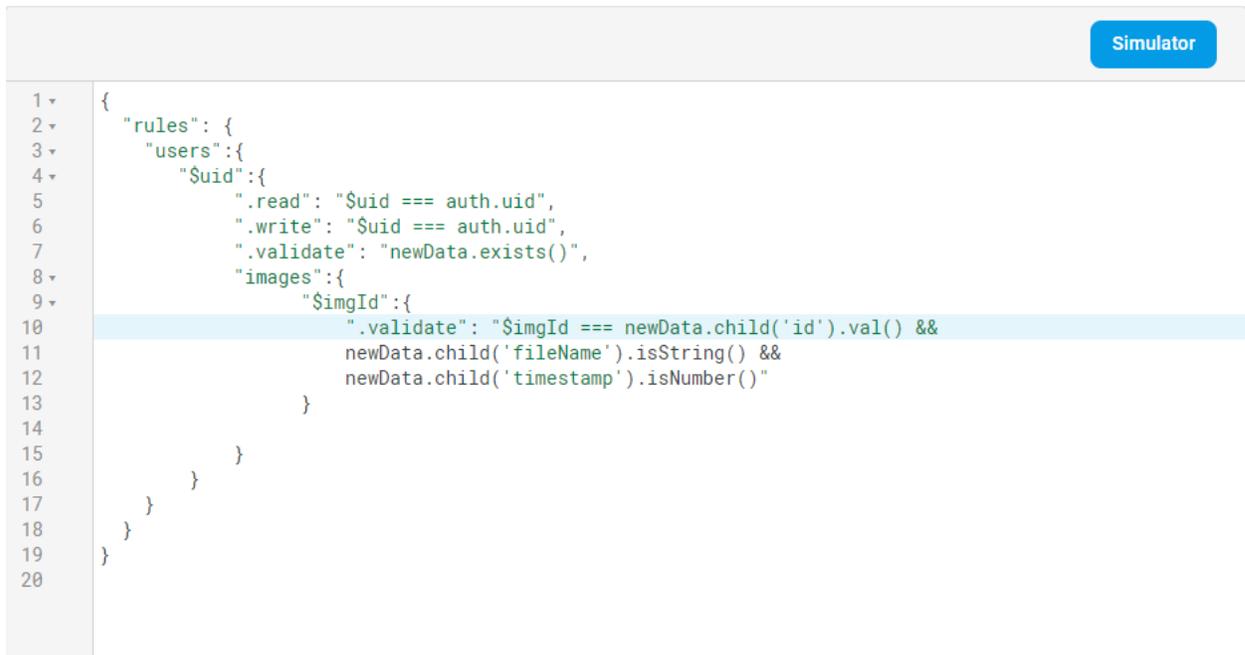
*Figure 14 Start media capture device*

In case the device has multiple video, capture devices button will be rendered which allows to switch between them (Figure 15).



*Figure 15 Capture and Camera Switch*

Once the user has taken picture given data will be sent to database. To send files to database firebase libraries promise can be used where given path determines where it is stored. Also, application marks the location and timestamps and saves data under user token key into real-time database. Realtime database has the possibility to create rules which validate, allow read and write into the database. (Figure 16).



```
1 {
2   "rules": {
3     "users": {
4       "$uid": {
5         ".read": "$uid === auth.uid",
6         ".write": "$uid === auth.uid",
7         ".validate": "newData.exists()",
8       }
9     }
10    "images": {
11      "$imgId": {
12        ".validate": "$imgId === newData.child('id').val() &&
13          newData.child('fileName').isString() &&
14          newData.child('timestamp').isNumber()"
15      }
16    }
17  }
18 }
19 }
20 }
```

*Figure 16 Firebase security rules*

Given rules are necessary to validate all the inputs. Simplicity of creating rules eliminates the large amount of code required to validate and store data in the database and streamlines process.

In home directory inside the application the user is able to see the picture they have taken (Figure 17).

Name :picture-1553112900029	Date Added: 20.3.2019 22:15:00	Response : No Data	Status : Not Analyzed	Search	Show
Name :picture-1553113071606	Date Added: 20.3.2019 22:17:51	Response : No Data	Status : Not Analyzed	Search	Show
Name :picture-1553113079868	Date Added: 20.3.2019 22:17:59	Response : No Data	Status : Not Analyzed	Search	Show
Name :picture-1553113083085	Date Added: 20.3.2019 22:18:03	Response : No Data	Status : Not Analyzed	Search	Show

*Figure 17 User panel*

To get all the data about user taken images. Database event listener has to be created (Figure 18).

```

newImageListener(){
  this.db.on('child_added',snap =>{
    this.images.push(snap.val());
  });
}

```

*Figure 18 Event listener*

Event listener is listening to any changes to data and for the existing of data in the database. When data changes its automatically pushed to user to view. Using the firebase API has eliminated a lot of complicated code that is required to execute those functions and shortened the time it takes to develop application. Downside of building the app on BaaS is it ties down the project to given cloud service provider.

### **3.1.5. Cloud application summary**

Using modern technologies and existing technologies enables to remove a list of complicated problems from the development. Building full application on cloud enables to develop scalable application with ease. While designing application for cloud, data security has to be kept in mind. There is always a possibility of losing control over the data. In given solution free tier of cloud service was never exceeded. Cloud applications can become quite expensive in the case of bad design.

## **3.2. Server based application**

Secondary prototype was developed to indicate the possibility to build application without using any sort of cloud. Given application was built using php, jQuery, HTML and CSS and relational database MYSQL. Given prototype consist of frontend and backend. Communication between frontend and backend happens using jQuery AJAX (asynchronous JavaScript and XML). Using ajax enables to send data without affecting the current page which enables dynamical websites.

### **3.2.1. Application design**

Given application consist of two parts frontend and backend which requires to develop both parts code. In the front-end JavaScript library jQuery was used to implement the asynchronous functionality (Appendix 2). If the user is trying to register account first, we have to validate all the fields are filled. This type of validation can be done on the frontend part also but is not required. If the user removes JavaScript from the frontend, they can remove given validation solution and it has no effect. In the backend this type of validation is mandatory (Figure 19).

```

$username = trim($_POST['user']);
$password = trim($_POST['pass']);

if(!$username & !$password){
    echo "All fields required";
    return false;
}
else if(!$username){
    echo "Username required";
    return false;
}
else if(!$password){
    echo "Password required";
    return false;
}
else{
    include_once '/Dbconfig.php';

    if($user->is_loggedin()!=""){
        echo "Logged in already";
    }
    else{
        if($user->login($username,$password)){
            return false;
        }
        else{
            echo "Wrong Details !";
        }
    }
}
}

```

*Figure 19 Php validation*

Because we are building the application on server, we also have to build the logic that communicates with database (Figure 20). This also might be required, depending what type of cloud model is used when developing cloud application.

```

public function Select_Ingredient ($Array_ingredient){

    $ID_Array = [];
    foreach($Array_ingredient as $f_name){
        $f_name="%$f_name%";
        $query="
            SELECT
                Ingredient_ID,Ingredient
            FROM ingredient_table
            WHERE
                Ingredient
            LIKE (?)
        ";
        $string=mysqli_prepare($this->connection,$query
        mysqli_stmt_bind_param($string,"s",$f_name);
        mysqli_stmt_execute($string);
        mysqli_stmt_bind_result($string,$id,$Ingredient);

        while(mysqli_stmt_fetch($string)==true)
        {
            array_push($ID_Array,$id,$Ingredient);
        }
        mysqli_stmt_close($string);
    }
    return $ID_Array;
}

```

*Figure 20 Database request*

When user tries to login again data must be validated and sent to custom back end code to handle verification. The difficult part of creating login and registration system is the way how sessions of users are handled. In this application cookies were used to keep users logged in (Figure 21). Given cookies enable the user to log in and use other functionality.

Handling security when developing applications is complicated this requires great understanding how system works. In the case when developing application with backend code different types security risks have to be assessed:

```

public function is_loggedin(){
    if(isset($_SESSION['user_session'])){
        return true;
    }
}

public function logout(){
    session_destroy();
    unset($_SESSION['user_session']);
    return true;
}

```

*Figure 21 Session handling*

- SQL injection – This type of attack can happen when SQL queries are not sanitized properly (Figure 22). SQL injections are very dangerous because it leaves user databases open to be accessed by malicious people. This why every user input should be properly escaped.

```

$username = "test123";
$password = "secretPassword";

$sql = 'SELECT user_id FROM accounts WHERE username = '.$username.' AND
password = '.$password.'';

```

*Figure 22 SQL Injectable query*

- Timing attacks – Defending of against timing attacks requires great knowledge. Timing attacks are possible when simple compare is used (Figure 23). The issue comes from how the compare is built in languages. The code finishes comparing when mismatch is reached. This would enable the person to brute force the codes secret key by sending symbols and measuring the time it takes to get a response. To avoid this, compare function should be used which take same time whether the strings that are being compared are true or false.

```

$userKey = "secretKey";

if($_GET['secret'] !== $secret){
    return false;
}

```

Figure 23 Timing attack

When working with own hardware and servers, being able to secure all of these parts can be quite complicated. Given solution that have been shown have not been optimal.

In the case of building application on server it is not always important if the minimal amount SQL queries are made to database. Its important database model is well thought trough otherwise even on small scale database might have issues serving all the clients. Given application database model was quite small (Figure 24).

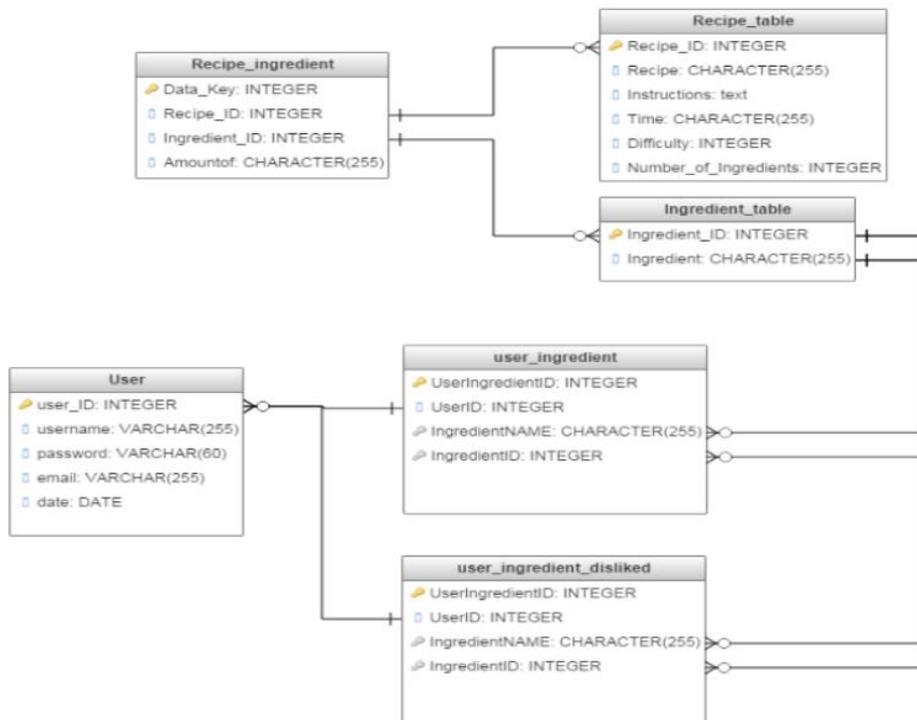


Figure 24 Database model [25]

Given database model was have some limitation which can cause some tables to become too long which could require to rework the design.

In the case of implementing application on server it is possible to monitor every part of the code and measure the time it takes to execute. This could be used to optimize code and eliminate unneeded usage of resources. Also building applications on server enables to easily maintain all the backups of databases without any extra cost. Hardware failure is considered one of the greatest risks of owning your own server rack. If servers fail full system outage could happen and all the data would be lost.

### **3.2.2. Server application summary**

Building application on the server can be quite complicated by sear number of things developer has to know. Server solution should be considered only if the number of users is certain or scaling is not necessary. Server solutions are also created for prototyping in given case security doesn't have to be considered. Running small applications on the server is cheaper than on the cloud. Application that requires a lot of SQL queries and compute time will make any application expensive when its running on the cloud. The best practice would be to build a hybrid solution that can use local and cloud service.

## **Conclusion**

Given thesis main goal was to analyze if cloud solution is viable in every case and if all application should be moved to cloud. In given thesis different type cloud models were analyzed. Also, the cost of running application on cloud was assessed against server solution.

Two types of prototypes were created to analyze difference between building application on the cloud against building applications on the server. Different types of technology were used, and different types of problems were solved. Understanding the pros and cons of either type of solutions is important. Developing applications for servers or for the cloud are both viable. In both cases developers rarely own the hardware which the code is executed on. Which leaves the security issue of data to both. Choice of server or cloud must be made on the premise does the software has to be scalable or use new technology. In the case where the software doesn't have to scale exponentially server solution should be used. Server solution enables the developer to solve problem in less efficient manners and have a certain amount flexibility, but servers leave developers open to more security threats. In the case where rapid development and modern technologies are required, application should be built on cloud.

Given thesis offers the possibility analyze clouds in more detail, analyze speed and availability in more detail between different cloud service providers. Also constructing your own private cloud could be possible topic to do research. Creating complicated application with unique features which highly depend on backend as service could offer unique future topics.

## Kokkuvõte

Antud töö eesmärgiks oli analüüsida ja hinnata pilve lahenduste sobivust igasse projekti ning kas kõik aplikatsioonid ja rakendused tuleks üle viia pilve teenuse pakkujatele. Samuti hinnati töö käigus serveri ja pilve lahenduste maksumust.

Töö käigus loodi kaks prototüüpi, mille analüüsi käigus eristus serveri ja pilve rakenduste loomise. Rakenduste loomiseks kasutati erinevaid veebitehnoloogiaid, millega lahendati erinevat sorti probleeme. Oluline on mõista mõlema rakenduse tugevusi ja nõrkuseid. Antud töö järeldus, et rakenduse loomine nii serveritele ja pilvedel mõlemale on sobilik. Enamikel juhtudel arendajad tänapäeval ei oma füüsilist riistvara. Milles tulenevalt eksisteerivad andme turvariskid mõlematel lahendustel. Valides serveri ja pilve vahel tuleb arvestada, et kas antud lahendus vajab skalaarset arvuti võimsust ning kui palju uusi tehnoloogiaid soovitakse kasutada. Samuti serveritele loomine pakub arendajatel lahendada probleeme vähem efektiivselt ning muudab arendamist paindlikumaks. Samas seavad serveritele loodud lahendused tihti serveri omanikud ja arendajad haavatavamaks rohkematele turvariskidele. Juhul kui kiiresti minimaalse produkti loomist ja moodsaid tehnoloogiaid soovitakse kasutada tuleks luua rakendused pilve teenustele.

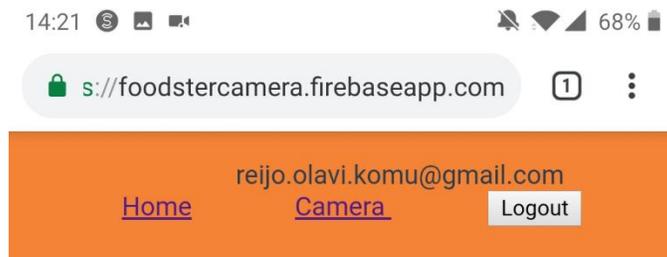
Antud töö võimaldab tulevikus analüüsida erinevaid pilve teenuse pakkujaid detailsemalt. Analüüsides pilve teenuse pakkujate kiiruseid ja kättesaadavust. Samuti võimalik tulevikku eesmärk oleks luua enda privaat pilv. Kuna pilve teenused pakuvad palju uusi tööriistu siis oleks võimalik ka luua keerulisemaid ja huvitavamaid rakendusi tulevikus ning neid analüüsida.

## List of references

- [1] [Online]. Available: <https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/>. [Accessed 12 April 2019].
- [2] "<http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-146.pdf>," [Online]. [Accessed 01 April 2019].
- [3] R. Hill, L. Hirsch, P. Lake and S. Moshiri, *Guide to Cloud Computing: Principles and Practice*.
- [4] K. Chandrasekaran, *Essentials of Cloud Computing*.
- [5] [Online]. Available: <https://rajivramachandran.wordpress.com/2012/06/19/cloud-service-models-iaas-vs-paas-vs-saas/>. [Accessed 11 04 2019].
- [6] "[https://s3.amazonaws.com/academia.edu/documents/](https://s3.amazonaws.com/academia.edu.documents/)," [Online]. [Accessed 2 04 2019].
- [7] "<https://whatis.techtarget.com/definition/Communications-as-a-Service-CaaS/>," [Online]. [Accessed 12 April 2019].
- [8] "<https://digitalguardian.com/blog/what-security-service-definition-secaas-benefits-examples-and-more/>," [Online]. [Accessed 12 April 2019].
- [9] J. R. Vacca, *Cloud computing security foundations and challenges*.
- [10] IBM, *Introduction to the new mainframe: Large scale commercial computing Chapter 5 Availability*.
- [11] "<https://aws.amazon.com/ec2/pricing/on-demand/>," [Online]. [Accessed 20 April 2019].
- [12] "<https://www.digitalocean.com/pricing/>," [Online]. [Accessed 26 March 2019].
- [13] "<https://azure.microsoft.com/en-us/pricing/details/virtual-machines/windows/>," [Online]. [Accessed 12 April 2019].
- [14] "<https://cloud.google.com/compute/pricing/>," [Online]. [Accessed 15 April 2019].
- [15] "<https://cloud.google.com/appengine/>," [Online]. [Accessed 09 April 2019].

- [16] "<https://docs.microsoft.com/en-us/azure/app-service/overview>," Microsoft, [Online]. Available: <https://docs.microsoft.com/en-us/azure/app-service/overview>. [Accessed 07 April 2019].
- [17] "<https://www.heroku.com/platform>," [Online]. [Accessed 23 April 2019].
- [18] "<https://www.srgresearch.com/articles/leading-cloud-providers-continue-run-away-market>," [Online]. [Accessed 05 April 2019].
- [19] "[https://cloud.oracle.com/en\\_US/essbase](https://cloud.oracle.com/en_US/essbase)," [Online]. [Accessed 03 April 2019].
- [20] [Online]. Available: <https://cloudboost.io>. [Accessed 20 April 2019].
- [21] "Firebase," Google, [Online]. Available: <https://firebase.google.com/>. [Accessed 26 March 2019].
- [22] "[www.zone.ee](http://www.zone.ee)," [Online]. [Accessed 03 April 2019].
- [23] "<https://www.zone.ee/et/privaatserver/puhas-raud/>," [Online]. [Accessed 21 April 2019].
- [24] "<https://www.veebimajutus.ee/veebimajutuse-vordlus/>," [Online]. [Accessed 21 April 2019].
- [25] "[https://www.trumpit.ee/et/rent\\_seade/2/Serveri-rent/](https://www.trumpit.ee/et/rent_seade/2/Serveri-rent/)," [Online]. [Accessed 12 April 2019].
- [26] "<https://vuejs.org/v2/guide/>," [Online]. [Accessed 26 March 2019].
- [27] R. O. Komu, "Multifunktsionaalne otsing toiduretseptide andmebaasi näitel," 2017.

## Appendix 1 – Camera app camera view



### Simple Camera App



## Appendix 2 – Server applications frontpage

