KARELIA UNIVERSITY OF APPLIED SCIENCES
Degree Programme In Business Information Technology

Valtteri Laine

Review of Server Side Development Technologies for Social Application Dibs

Thesis
MAY 2016

	THESIS **May 2016** **Degree Programme In Business Information Technology** Karjalankatu 3 80220 JOENSUU FINLAND 013 260 600

Author (s)
Valtteri Laine

Title
Review of Server Side Development Technologies for social application Dibs
Commissioned by
Joensuu Games cooperative

Abstract

The goal of this thesis is to be an instruction manual into server-side technologies for beginner server-side developers and, in addition, a guide in choosing the best technologies for each situation. Server side technologies are reviewed and explained from several categories including web host, server-side programming language, server databases and an internet protocol for communication between a server and a distributed client.

A social application containing game like features called Dibs was in development while writing this thesis for a cooperative company named Joensuu Games. The server side technologies are reviewed in general and the best technology from each category is chosen for developing Dibs. Evaluation criteria include history, popularity, ease of use, community and best use cases.

Six databases, five programming languages, two internet communication protocols and two web host categories are reviewed. For protocols there are clear best use cases as well as for web hosts and databases. The choice between best programming language is subjective at best. The goal of reviewing server-side technologies is to help beginner developers better understand server-side development. A functional prototype of Dibs was made during the writing of this thesis.

Language English	Pages 56

Keywords
server, distributed-client, IaaS, PaaS, web host, Scala, Node.js, Java, PHP, Ruby, MongoDB, MySQL, SQLite, Database, Programming language

Karelia UNIVERSITY OF APPLIED SCIENCES	**OPINNÄYTETYÖ** **Huhtikuu 2016** **Degree Programme In Business Information Technology** Karjalankatu 3 80220 JOENSUU FINLAND 013 260 600

Tekijä(t)
Valtteri Laine

Nimike
Pavelinteknologioiden esittely Dibs-sovellukselle

Toimeksiantaja
Joensuu Games -osuuskunta

Tiivistelmä

Tässä opinnäytetyössä annetaan ohjeita aloitteleville palvelinpuolen ohjelmoijille palvelinteknologioiden käytöstä ja parhaan mahdollisen kehitysteknologian valinnasta ohjelmistoa varten. Opinnäytetyössä läpikäytävät palvelinpuolen teknologiat voidaan luokitella seuraaviin teknologia kategorioihin: ohjelmointikieli, tietokanta, webhotelli ja kommunikointi protokolla.

Työssä kehitettiin Joensuu Games -osuuskunnalle Dibs-sovellusta. Dibs-sovellus on pelillisiä ominaisuuksia sisältävä sosiaalisen median sovellus, jolla luodaan ja jaetaan itse tehtyjä pelikortteja. Dibs-sovellus on esimerkki tapauksena eri teknologioiden valitsemiselle. Teknologioiden arviointikriteerit ovat historia, helppokäyttöisyys, suosio, yhteisö ja parhaat käyttötapaukset.

Työssä tarkastellaan kuutta tietokantaa, viittä ohjelmointikieltä, kahta Internet-protokollaa ja kahta webhotellikategoriaa. Protokollista, webhotelleista ja tietokannoista löytyi selkeät parhaat käyttötapaukset, mutta palvelinpuolen ohjelmointikielissä ei ollut selkeitä parhaita käyttötapauksia. Palvelimen eri teknologioiden läpikäyminen auttaa aloittelevaa ohjelmoijaa ymmärtämään palvelinkehittämistä kokonaisuutena. Dibs-sovelluksesta tehtiin opinnäytetyön aikana toimiva prototyyppi.

Kieli English	sivuja 56

Asiasanat
palvelin, distributed-client, IaaS, PaaS, webhotelli, Scala, Node.js, Java, PHP, Ruby, MongioDB, MySQL, SQLite, tietokanta, ohjelmointikieli

Contents

1 Introduction

Joensuu Games OSK is a cooperative game company founded in fall of 2014. Joensuu Games (JG) is located in Joensuu Science Parks Game Lab, and is working on multiple game projects including Sotka and Dibs. In addition, Joensuu Games works on various commissions, for example, websites, graphics and marketing. JG has numerous students working for them and it is common for students to receive gaming related thesis assignments from JG or perform their internships there. JG has around 30 members.

The assignment developed for JG is Dibs and the server side technologies are reviewed with Dibs in mind. The best technology options for Dibs are chosen from the following categories internet protocol, web host, programming language and database. In addition, there is a general review and comparison between the options for each category, on what are their advantages and best use cases. General terminology is explained.

After reading this paper the reader should have a general understanding of various server side development technologies. A server has multiple components that need to communicate with each other efficiently, sometimes over the Internet. This makes server development inherently more complex than developing local applications. This is why it was necessary to write an introduction into the various technologies that most servers have. General guidelines are written instead of in-depth analysis.

In the second chapter Dibs is explained. The working method chapters subjects are project management, time tracking, version control and other software used in developing Dibs.

In the third chapter the definition of the term server is explained in addition to a brief history of server-side technology. History is important as it provides perspective on how fast the field has developed. Furthermore, the chapter recalls some of the original technologies and uses of servers. Some of the technologies are still in use today. In addition, modern web hosts are examined and compared.

The fourth chapter is about protocols, such as HTTP and websocket. The basics of protocols are explained as well as the best use cases for each protocol. A protocol for Dibs is chosen, and the reasoning behind the decision is explained.

The fifth chapter concentrates on different programming languages. In this chapter there is some basic knowledge of each language and the languages are evaluated by their ease of use, popularity, syntax and age. After this chapter the reader should have an idea about the current trends on the programming field and of each languages syntax. There is a code example of a basic echo server for each language.

The sixth chapter concludes the thesis on databases. Not all servers need a database, Dibs, However, does. The most popular open-source solutions are compared and the best use cases are reviewed, for SQL and NoSQL databases. The small budget of Dibs and the positive trend of nonproprietary databases is why only nonproprietary solutions were looked at.

2 Assignment

2.1 Application

The work assignment from Joensuu Games cooperative is to develop a social mobile application called Dibs. Dibs is a gaming application for mobile devices, where users can create cards of themselves, trade them and finally share them in social media.

Dibs was planned as having a server with a distributed client solution. Distributed client solution stands for a system where there is a server to which multiple user clients connect. When the clients connect to the same server, the server then connects the clients and allows communication between them through the server. Client in this case is the Dibs mobile application. Users will create cards using the mobile application and then send them to the server and the server will send them to other clients. Dibs needs to serve tens of thousands users and grow rapidly. Consequently, the tools and services chosen for the assignment should be highly scalable and efficient.

To realize the technical side of Dibs server the developers need to consider the following aspects of a server:

- Server / web host
- Protocol
- Programming language
- Database

Several technologies from each category can be used for the solution, for example two databases. For small companies this is typically not necessary, and one technology from each category is enough, although using different technologies for different features can result in more optimal performance. For making a prototype of Dibs one from each category is enough. Sometimes a server does not need certain technologies at all, for example databases.

2.2 Work method

The approach in developing this project is SCRUM-methodology. When using Scrum methodology, the development of the software is split into small several week long sprints. In addition, developers do not necessarily have specific roles. (Meteoriitti 2013.) In each sprint Dibs developers will be working on a small update for the software, preferably complete features or updates. The goal is to have a testable version of the service at all times. Dibs developers decided to use the Asana web service, use Google Drive for editing and sharing documents and GitHub for version control. The client is made with Unity 3D version 5.

The communication channels were chosen to be Facebook, Asana, Google Drives collaborator features and real life meetings. A meeting is held at the beginning of each sprint and the main purpose of these meetings is to define the tasks done during that sprint and reflect on the previous sprint. A simple minutes document is produced for each meeting to keep track of everything discussed and decided during the meeting.

As the project advances, Dibs developers need to be able to create prototypes quickly, test features and see how users react to these features. As a result, tools that allow for quick development are chosen. Time should not be wasted in creating a sophisticated solution that nobody uses.

3 Server / web host

3.1 What is a server?

In information technology, a server is a computer program that provides services to other computer programs, running on the same computer or on other computers. The computer hardware that the server is running on can in addition be called a server, though computer hardware could provide other services in addition and not just run server software. Consequently, the word server is a common source for confusion, as the word server could refer to server hardware, server software or a combination of the two. Servers can communicate over the Internet, LAN or between programs on the same computer. (Whatis.TechTarget 2014.)

Behind every website, there is server hardware and server software, handling all of the requests from people visiting the website. The server responds to these requests by sending the requested information, which in this case is a website.

Any computer can perform as a server. A server computer can be optimized for its purpose just like a computer’s hardware can be optimized for gaming, office work or graphical design or any other task. In most cases for a server computer, a high amount of system memory, disk space and a server-oriented processor is optimal. For example, if a web shop server handles 10 000 customers a day, that is a large amount of user data to hold in system memory and to store on a server’s hard drive. It is difficult to predict how powerful hardware is needed. Testing and measuring in practice is typically the best option. (Confluence Atlassian 2015.)

3.1 History and future of servers

The Internet has existed since 1950s (Wikipedia 2015a), although the modern World Wide Web was born in 1991. In 1990, a scientist at CERN by the name of Berners-Lee tested the Web Protocols that would be implemented for the very first World Wide Web server. In 1991, a physicist Paul Kunz installed the first web server in U.S., at Stanford University. This web server was used to retrieve documents from the university’s bibliographic database and even the researches were surprised of how well it worked. The computer used for running the web server was made by a company called Next. (ComputerWorld 2002.)

The first Google server was made in 1996, although the company was only founded in 1998 (Wikipedia 2015b). Today it is estimated that Google has up to 1.5 million servers running, though the exact amount and location of the servers is being kept secret (Plus.Google 2012). Google is just one of the large Internet corporations that exist today.

Considering web serves were first utilized in 1991 the growth in server count has been exponential. Most people globally do not even have access to the Internet; the growth will likely not slow down as the potential market expansion is enormous (ITU 2015).

3.1 What is a web host?

Web host is a cloud service to run your server software on, for example to run a website or the backend of an application like in the case of Dibs. Web hosts replace the need for developers to obtain server hardware. Consequently, Web hosts allow developers to effortlessly have their server software available on the Internet. With various web hosting services being available, developers can take shortcuts by employing various readymade packages to put their server software online. Web hosts provide a ready architecture specifically made for quick and easy deployment of a server software. Web hosting is a service and companies are doing their best to sell and develop their own web hosting services, with the target audience being the developers of various websites and applications.

Web hosts have differences for example in scalability, pricing, features and performance. Other differences include region, latency, operating system support, programming language support and database support. When choosing a web host developers need to make sure it can run the tools that have been chosen for the solution, and that the server is physically located near a region that most of the applications customer base is going to be in. This minimizes latency.

There are many different types of web hosting and literally hundreds of different providers of web hosts. That is why instead of going through the various providers, basic terminology is covered and then the best type of web host is decided for Dibs. (Websitesetup 2015.)

3.2 Scalability

Web hosts utilize two types of scaling. The two types are auto-scaling and dynamic-scaling. Scaling is used to match the current performance requirement of the server, with the hardware resources that the developers pay for from the web host. The difference between the two types of scaling is that automatic scaling scales without any human interaction. Dynamic scaling happens when a developer allocates more resources to the server, without any automation. Dynamic and automatic scaling can be combined. Scaling can reduce or increase server's resources. (Broadcast.oreilly 2008.)

The reason to have downwards scaling is to save money and resources. When the server experiences less traffic, the server can scale down and the developers save some money. (Broadcast.oreilly 2008.)

For this assignment a web host that offers seamless upwards and downwards scaling of the server is necessary, because of the fluctuating traffic. Users use their mobile phones at certain times, usually in the morning and in the evening after work, and obviously not during the night. Upwards scaling will prevent user spikes from crashing the server and making it unavailable to everyone. This can be exceptionally embarrassing as it happens when the service has a peak number of users.

The fact that a web host is scalable does not remove the requirement for optimizing or calculating the amount of computing power needed. The more the server requires computing power the more it will cost, and if there is an unexpected memory leak or some other performance sink the servers cost will increase rapidly. A DDOS attack or some other attack that taxes the server will have the same effect. Instead of being overwhelmed by the computing power sink the performance and the price will keep scaling upwards with the bill from web host company. (Broadcast.oreilly 2008.)

3.3 PaaS

Platform as a service (PaaS) is a web-hosting platform that provides software and hardware for developers. As a result, PaaS users do not have to build their own hardware infrastructure or their own development environment to run and develop their application. This is useful for teams with limited time, resources and knowledge. Deploying and connecting to an environment for server software to run on can only take a couple of minutes with a PaaS. Typically, a PaaS is accessed through a web browser or a console.

There are some potential issues with PaaS. Outsourcing server hardware can mean unexpected down times that the developers cannot affect. Typically, PaaS providers promise 95 – 99 percent up time, which is satisfactory for majority of servers. There have been situations where a PaaS stops supporting a language, or the PaaS's provider could go bankrupt or stop offering their services for some other reason. This would force developers to migrate their server to another web-host. Server that has been built around a certain PaaS could be very dependent on it and migrating to another web host can be time consuming. (SearchCloudComputing 2015.)

Dibs has a small team and a rapid schedule. Outsourcing server hardware and software to a web host saves time and money in this case. Depending on the PaaS service, setting up a server environment can be free, and after the application starts to receive traffic and requires more resources the developers can pay to scale the resources accordingly. PaaS offers more features and automation for developers, compared to IaaS for example; therefore, there are cheaper options.

Some PaaS providers are Appear IQ, Mendix, Amazon Web Services (AWS) Elastic Beanstalk, Google App Engine, RedHat OpenShift and Heroku (SearchCloudComputing 2015).

3.4 IaaS

Infrastructure as a service (IaaS) is a web hosting form similar to PaaS. It provides access to server hardware via the Internet, just like PaaS, although with IaaS, the features are more limited. The user does not have to worry about maintaining or upgrading hardware, and can instead concentrate on software. Sometimes the ready software solutions provided by PaaS are not desirable for a customer's particular use or they do not offer enough flexibility. This is when an IaaS is a better choice. Most of the PaaS downsides apply for IaaS such as reliability concerns. (InteRoute 2015.)

IaaS services are less expensive than PaaS as they concentrate on giving narrower spectrum of features. Some IaaS providers include Windows Azure, Google Compute Engine, Amazon AWS Rackspace Open Cloud, IBM SmartCloud Enterprise and HP Enterprise Converged Infrastructure (TomsITPro 2014).

3.5 Server / web host choice for Dibs

In this situation, a PaaS service is the best choice for Dibs. It saves developing time, and offers great scalability and performance. While it is more expensive, compared to an IaaS, the upkeep is minimal. It is more cost and time efficient to deploy the ready-made server features from a PaaS service compared to developing them our outsourcing them from elsewhere.

Some IaaS and PaaS services considered for Dibs were Jelastic, OpenShift, DigitalOcean and Heroku. The cheapest options are DigitalOcean or Jelastic as they are IaaS services. Heroku and OpenShift are PaaS services, and consequently more expensive. Although they are still free to adopt. Heroku offers a more comprehensive plugin library than OpenShift and additionally Heroku is more commonly in use. Heroku only provides dynamic scaling natively, although it can be upgraded to support dynamic scaling using a third party plugin (HireFire 2015). Heroku was chosen for its low starting cost, comprehensive plugin library and ease of use.

4 Protocol

4.1 What is a protocol?

Protocol is a set of rules both sides of the conversation have agreed to use for their successful communication. If there was no protocol and no rules for the conversation, communication would be much more complicated. The communication could be between a server and a program in computer terms, although protocol could stand for the rules of communication between any entities. For example, communication between people. If a person says “hi”, he expects the other person to greet him back. The same way when a computer sends a request to another, it is in most cases expecting a certain response. The type of response is determined by the protocol both parties have agreed to use. (TheNewBoston 2012.)

When browsing the Internet, you can tell what protocol a server is using by its Uniform Resource Locator (URL) you use to connect to it. The first part of an URL tells what protocol it is using. For example, *“http://www.thenowboston.org”* website is using the HTTP protocol; *“https://www.nordea.fi”* would be using a secure form of the HTTP protocol called HTTPS. In addition, there are other protocols, for example, the websocket protocol *“ws://www.WebSocketService.org”*, and the file transfer protocol *“ftp://www.FileSharingService.org”*. Reviewing what protocol to use for each scenario is important. HTTP and websocket protocols are the two most common protocols used for server to client communication.

4.2 HTTP

Hypertext Transfer Protocol (HTTP) is the most common protocol used in Internet communication and is one of the three foundation technologies of data communication for the World Wide Web (WWW) alongside HTML and Uniform Resource Identifier (URI). Tim Berners-Lee created all three in 1990 while he worked at CERN. (WebFoundation 2015.) Despite HTTP:s, age it is still in wide use and is one of the most commonly used protocols (AllAboutCookies 2015).

It is most suitable and commonly used for hypertext (Wikipedia 2015c). Hypertext represents a database system where the data is linked to each other through links. When selecting an object, the user is able, see all the other objects linked to it. For example, while reading a hypertext article about Mozart the user could jump to an article about Bach by clicking a link. (Webopedia 2015.) An example would be any web page utilizing HyperText Markup Language (HTML).

HTTP functions as a request-response protocol in application protocol layer (Wikipedia 2015c). HTTP is built on top of TCP/IP. HTTP establishes a connection using the TCP/IP and sends data using the connection in a HTTP protocol format (Tutorialspoint 2015c). The client, using the HTTP protocol, sends a request to the HTTP server whenever it wants a response or data from the server. The request provided to the server can include additional data in binary or string form and the response can have data in it (Wikipedia 2015d). With HTTP 1.0, the connection had to be re-established for each response-request set, however with HTTP 1.1, the support for keep-alive connections was added. This means that the same TCP/IP connection can be reused for posting requests and responses, which makes the HTTP 1.1 protocol a faster and lighter. This feature can be called “Persistent connections”. (www8 2015.) Both HTTP 1.0 and 1.1 are still in use as 1.0 is better suited for some applications, though 1.1 is more common. This is because 1.0 has less features and can be easier to implement. It is more useful for simple clients or in situations with limited resources. A web server can and should accept both 1.1 and 1.0 HTTP connections (JMarshall 2012).

HTTP/2 was released in 2015. It reduces the amount of data sent with each request – response, removes the need for multiple connections, allows the server to push unrequested resources to the client and several other improvements. (Akamai 2015.) Current browsers that support HTTP/2 are Chrome, Chrome for iOS, Firefox, Internet Explorer on Windows 10 only, Microsoft Edge, Opera and Safari 9 (Caniuse 2015).

HTTP could be used for internet communication that does not require constant back and forth communication with a very low latency. Websockets are more suitable for back and forth low latency data-transfer. HTTP can do intermittent updates, although it is not as efficient compared to websockets. There are some libraries to go around HTTP’s limitations like Ajax and Comet, although even those solutions are still utilizing HTTP.

Here is an example of the data sent with the request in HTTP 1.1 (TutorialsPoint 2015a). This provides an idea of the amount of data being sent with each request. Knowing what the request and responses contain can help developers customize a connection to their own specific needs.

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://clearforest.com/">string</string>
```

This request is 356 bytes or 356 characters. Here is an example of HTTP 1.1 response (Tutorialspoint 2015b).

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
<html>
<body>
<h1>Hello, World!</h1>
</body>
```

```
</html>
```

In HTTP/2 requests and responses will be sent in binary. Binary form makes them less prone to errors and smaller in size. Binary form means that a decoder is needed, which makes debugging a bit more difficult and slower. (GitHub 2015a.)

If HTTP has to be used instead of WebSockets, there are several approaches to emulating similar functionality to Websocket with HTTP protocol. Functionality similar to Websocket would be full-duplex communication and server pushing for example. In certain situations, these technologies could be beneficial over regular HTTP or Websockets. HTTP was not originally intended for similar functionality to websocket. Consequently, the result can be resource intensive or overly complicated. These technologies include:

- Ajax Polling
- Ajax Long-Polling
- HTML5 Server Sent Events (SSE) / EventSource
- Comet. (Stack Overflow 2015a.)

4.3 WebSocket

Websocket Internet protocol is full duplex and real time. Full duplex means that it allows for two-way communication in near real time. After a websocket connection has been established both sides, the server and the client, can send data to each other at any time.

This makes the websocket protocol excellent for performance demanding assignments, where updates are constantly being sent back and forth between the server and the client. Applications like real-time graphs, real-time videogames, chats, collaborative tools and other applications that require constant updates are all great uses of the websocket protocol (InfoWorld 2013a). This is because with websockets, the client does not have to send a request to the websocket server to ask if there is anything new. Instead the websocket server sends an update to the client when there is something new. This is called server pushing.

Sometimes using HTTP over websockets can save resources as keeping the connection open for a websocket requires communication. If the user, for example, downloads a webpage and then reads that webpage for the following 15 minutes before requesting any other data, it can be less resource intensive to download the page using HTTP instead of websockets. (Pubnub 2015.)

4.4 Protocol choice for Dibs

If the user spends a long time in the editor, nothing is requested from the server and maintaining a websocket connection would be wasteful. Examples of communication between the client and the server would be saving, receiving or updating a card. These events do not require server pushing, and are initiated by the client instead.

Another aspect of Dibs is browsing cards stored on server using the user client. The idea is to have a Stack of cards that the user can browse by swiping the top card aside, like in a mobile dating application called Tinder. This could be done with HTTP or websockets. Downloading one card at a time from the server, every time the user swipes would create a delay in the displaying of the next card, no matter the protocol. To fix this the user would have, for example, ten cards downloaded and ready to be displayed to the user as he swipes through the card stack. The data would be requested by the client instead of initiated by the server.

For this assignment websockets were chosen. Websockets are more efficient, even if developers don not use websockets full scale of features. The request response messages are smaller and it allows efficient real time communication. In games and game like applications like Dibs, this is essential.

5 Programming language

5.1 Trends

There are numerous options for a programming language for implementing the server software for Dibs. In order to determine most relevant modern programming languages, the market share and the trends of different server-side programming languages are inspected. More popular programming languages are well validated, and there is typically plenty of learning resources and large communities available. In addition, there is more demand for talent with these programming languages on the job market. A programming language with a positive trend is more likely to have more available vacancies, while a programming language experiencing a negative trend is likely to have a surplus of talent on the job market. Existing solutions utilizing various programming languages give us perspective to what the programming languages are capable of.

Different sources for programming language trends have different ways of ranking the programming languages, therefore several are evaluated. The sources evaluated are GitHub, RedMonk, Jobs Tractor and TIOBE Index. GitHub has information on which programming languages have the most activity and repositories on GitHub. RedMonk evaluates data on GitHub and Stack Overflow. Jobs Tractor evaluates the number of job postings on twitter. TIOBE index looks at the number of skilled engineers, courses and search engine rankings. (SitePoint 2015a.)

Constantly the highest ranking server-side programming languages are Java, JavaScript, PHP, Python, C# and Ruby. Scala barely falls behind the top 10 programming languages on RedMonk comparison. (SitePoint 2015a.) All of these have positive trends, even when compared to all programming languages and not just the ones used for server-side programming. The dominant server-side language for websites is PHP (W3techs 2015).

5.2 Scala

Scala is a programming language that, as the name suggests, made to be as scalable as possible. Large websites like Twitter, LinkedIn, Arcusys and Intel utilize Scala. Scala is an entirely object oriented programming language (Scala-Lang 2012). Everything from values to functions are objects and acts like an object. Scala works seamlessly with Java libraries (Developers.RedHat 2015). For developers who have a background with object orientated programming and Java, Scala should feel familiar to them.

Scala would be an excellent choice for this assignment, although at the time of writing this thesis Scala's documentation and community size is inadequate. Finding learning material and active communities for Scala proved difficult. Since popular websites like Twitter, Reaktor, FourSquare, The Guardian and LinkedIn have adopted Scala, it is expected for the programming language to become more popular and as a result the documentation to improve and the community to grow larger. (Scala-lang 2009.)

Simplistic server example in Scala (Stack Overflow 2015b.):

```
import java.net._
import java.io._
import scala.io._

val server = new ServerSocket(9999)
while (true) {
    val s = server.accept()
    val in = new BufferedSource(s.getInputStream()).getLines()
    val out = new PrintStream(s.getOutputStream())

    out.println(in.next())
    out.flush()
    s.close()
}
```

Some Scala IDEs:

- NetBeans (NetBeans 2015)
    - Cross platform
    - Free
    - Open Source
- Eclipse (Eclipse 2015)
    - Cross platform
    - Free
    - Open source
- IntelliJ IDEA Community (JetBrains 2015a)
    - Free version
    - Cross platform

5.3 Node.js

Node.js is an asynchronous event and JavaScript based programming language. It is strongly typed and utilizes callbacks. A background with JavaScript or a similar language can make the transition to Node.js easier, compared to strongly typed programming languages, for example C++. Node.js was released in 2009 (Nodejs 2011b.)

JavaScript is a strongly typed dynamic language. Dynamic means that declaring variable types is not needed. Strongly typed means that on runtime the compiler is aware of the variables type. A common misconception is that JavaScript is a weakly typed language. JavaScript’s compiler engine has to be aware of the variables type on runtime in order to perform implicit casting. Implicit casting means that a variable will be casted to another variable type automatically. For example in JavaScript the following piece of code “`console.log("1" - 1);`” will yield zero, instead of throwing an error for incompatible variable types. This does not mean that the language is weakly typed, it means that on runtime it detects that the program is subtracting a variable of type string from a variable of type int and as a result casts the string into an integer. This is only made possible because the compiler is aware on runtime of the variables types. The fact that the compiler is aware of runtime variables means that the language is strongly typed. (Wikipedia 2015c; Webopedia 2015.)

Node.js has plenty of libraries and as a result doing certain things like setting up a simple HTTP server requires half of the code of doing the same in Java. Here is an example of a simple HTTP server in Node.js:

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 1337;

http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello World\n');
}).listen(port, hostname, () => {
  console.log(`Server running at http://127.0.0.1:1337/`);
});
```

This HTTP server responds with the message “Hello World” to all incoming connections and its IP is ”127.0.0.1” and port “1337”. The required HTTP module is part of the Node.js standard library and does not need to be installed separately. (Nodejs 2015.)

Unique aspect of Node.js is its asynchronicity. Usually programming languages are read line by line and the program waits for each line to complete before moving on to the next one. In Node.js the program moves line by line, without waiting for the line of code to finish its function. For example, if different messages are printed to the console, the messages could come up in any order.

For example, the following piece of code;

```
console.log("Log number 1");
console.log("Log number 2");
console.log("Log number 3");
```

could result the following console output:

```
Log number 3
Log number 1
Log number 2
```

or even

```
Log number 2
Log number 3
Log number 1
```

This makes programming Node.js more complex and different. Node.js is fast, scalable, based on JavaScript and easy to use, which makes Node.js a good candidate for Dibs server-side programming language.

IDEs that support Node.js:

- WebStorm (JetBrains 2015b)
  - Proprietary
  - Cross platform
- Komodo IDE (KomodoIDE 2015)
  - Proprietary
  - Cross platform
- Cloud9 - cloud-based IDE (c9 2015)
  - Cloud-based, works in browser
  - Free
  - Cross platform
  - Open source
- Aptana Studio (Aptana 2015)

    - Open source
    - Free
    - Built on Eclipse IDE
    - Standalone version natively only for Windows

5.4 Java

Java is an object-oriented programming language originally developed by Sun Microsystems in the early 1990s. James Gosling was responsible for starting the project, and he and his colleagues released the 1.0 java version to the public in 1995, which is the same year as PHP. Java was created to be similar to C-languages, but with greater uniformity and simplicity than C or C++. Familiarity with C#, C or C++ will help with Java. (FreeJavaGuide 2015.)

As an object-oriented programming language, almost everything in Java is an object, apart from numbers, Boolean values and characters. Everything in Java is written in a class. Still the fact that not everything is an object in Java, means it is not a pure object-oriented programming language. The purpose of object-oriented programming languages is to improve the reusability of the code. (FreeJavaGuide 2015.)

Some features missing from Java are multiple inheritance, operator overloading, class properties and tuples. These were dismissed due to a possible impact on performance. (FreeJavaGuide 2015.)

There are several Java IDEs to choose from. Here is a small list of some of the most relevant and free to use IDEs for Java. There is some overlap as IDEs support various programming languages, especially with Scala and Java.

- NetBeans (NetBeans 2015)
    - Cross platform
    - Free
    - Open Source

- Eclipse (Eclipse 2015)
    - Cross platform
    - Free
    - Open source
- IntelliJ IDEA Community (JetBrains 2015a)
    - Free version
    - Cross platform
- JDeveloper (JDeveloper 2015)
    - Free
    - Cross platform
- DrJava (DrJava 2015)
    - Cross Platform
    - Light weight
    - Free
    - Open source

Here is a simple example server implemented with Java version six. (GitHub 2015b).

```
package com.stackoverflow.q3732109;

import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;

import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;
import com.sun.net.httpserver.HttpServer;

public class Test {
      public static void main(String[] args) throws Exception {
            HttpServer server = HttpServer.create(new
            InetSocketAddress(8000), 0);
            server.createContext("/test", new MyHandler());
            server.setExecutor(null); // creates a default executor
            server.start();
      }
      static class MyHandler implements HttpHandler {
            @Override
            public void handle(HttpExchange t) throws IOException {
                  String response = "This is the response";
                  t.sendResponseHeaders(200, response.length());
                  OutputStream os = t.getResponseBody();
                  os.write(response.getBytes());
                  os.close();
            }
      }
}
```

5.5 PHP

Hypertext Preprocessor (PHP) is an open-source solution especially for dynamic websites and server-side programming. In addition, it is used for general-purpose programming. Generally, PHP is embedded into HTML, or used with various web template systems and frameworks. The first version of PHP was released in 1995, same year as Java. In the beginning PHP evolved without a de facto standard, until in 2014 the canonical PHP interpreter was accepted as a de facto standard. Programming language that PHP is most similar to is Perl. PHP is a dynamic and a weakly typed language. (Wikipedia 2015e.) The largest difference for example to JavaScript, is that PHP is executed on the server, which generates HTML that is then sent to the client. JavaScript is instead executed on the client. (PHP 2015.)

Of the website server-side programming languages, PHP is dominant with over 80% of websites utilizing PHP. This is counted by evaluating the top ten million websites. (W3techs 2015.)

Example of a PHP server done with websockets. This simple example prints out “Client is here”, if the client manages to connect to the server.

```
<?php
```

```
    $address="127.0.0.1";
    $port=9875;
    echo "I am here";
    set_time_limit (0);
    if(false==($socket=  socket_create(AF_INET,SOCK_STREAM, SOL_TCP)))
    {
        echo "could not create socket";
    }
    socket_bind($socket,  $address,  $port)  or  die  ("could  not  bind
socket");
    socket_listen($socket);
    if(($client=socket_accept($socket)))
        echo "client is here";

    socket_close($socket);
    ?>
```

Due to the fact that PHP has been very popular and it has been available for so long, it has gathered a large community and the internet is full of guides and tutorials on how to utilize PHP. This is what makes PHP a great entry-level language into server-side programming.

IDEs that support PHP from most to least popular (SitePoint 2015b):

- PhpStorm (JetBrains 2015c)
    - Proprietary
    - Cross platform
    - Most popular
- Sublime Text 2 (Sublime 2015)
    - Free (Developer has to pay 70$ to sell their product)
    - Minimalistic
    - Customizable
    - Requires plugin
    - Cross platform
- NetBeans (NetBeans 2015)
    - Cross platform
    - Free

    - Open Source
- Zend Studio (Zend 2015)
    - Only native Windows support
    - Proprietary
    - Mobile development
- Aptana Studio (Aptana 2015)
    - Open source
    - Free
    - Built on Eclipse IDE
    - Standalone version natively only for Windows
- PhpDesigner (MpSoftware 2015)
    - Only for Windows
    - Proprietary

5.6 Ruby

Ruby was released in 1995. It was designed and developed by Yukihiro Matsumoto in Japan. Perl, Smalltalk, Eiffel, Ada and Lisp inspired Ruby. Ruby is dynamically typed and supports multiple programming paradigms, for example, functional, object-oriented, and imperative. Ruby is a pure object-oriented programming language. (Wikipedia 2015f.)

Ruby's cross platform IDE’s (Wikipedia 2015g)

- Aptana Studio 3
    - Cross platform
    - Propietary
- RubyMine 6
    - Cross platform
    - Proprietary
- NetBeans with Ruby Plugin
    - Cross platform
    - Free

- Komodo
    - Cross platform
    - Proprietary
- Arcadia. (CodeCondo 2014.)
    - Cross platform
    - free

Example Ruby code for a server (Sandbox.mc 2015):

```
equire 'socket'
server = TCPServer.new(12321)

while (connection = server.accept)
  Thread.new(connection) do |conn|
    port, host = conn.peeraddr[1,2]
    client = "#{host}:#{port}"
    puts "#{client} is connected"
    begin
      loop do
        line = conn.readline
        puts "#{client} says: #{line}"
        conn.puts(line)
      end
    rescue EOFError
      conn.close
      puts "#{client} has disconnected"
    end
  end
end
```

Ruby’s syntax is very minimalistic and as a result unique. Ruby relies on indentions instead of brackets to determine blocks of code. This forces developers to always format their code in a certain way, making it more uniform.

Large websites built with Ruby are, for example, Indiegogo, GitHub, Hulu, Funny or die, Ask.fm, Twitch (SkillCrush 2015). These are websites with millions of users.

5.7 Programming language choice for Dibs

The options presented here are formidable and any of them could produce a satisfactory solution for the assignment. Each language is capable of providing the necessary performance, features and scalability required.

PHP is by far the most popular, which greatly helps in finding tutorials, support and existing solutions. With other programming languages there are not as many ready answers online, which means the developers have to do more research and coming up with new solutions. Still there is an active community for each of these languages. Old established languages like PHP, Java and Ruby have more material online, while languages like Scala and Node.js have less.

Performance vise it is hard to say which programming language is the most efficient. Performance of a programming language is highly dependent on the implementation of the benchmark software, which makes benchmarking unreliable.

In order to play it safe and have plenty of help from community, PHP is a good choice. In the case of Dibs, the developers are junior developers, which makes available learning material even more important. Still for Dibs, Node.js was chosen as the event-driven asynchronous nature is new and interesting, and it has been successfully deployed for similar solutions. Node.js has comprehensive documentation and developers are familiar with it whether they are front- or backend developers. This is important as other people will be programming for the assignment and later on hiring new Node.js programmers will be easier and as a result cheaper than less known languages like Ruby or Scala.

6 Databases

Database is a separate entity from the application. A database is used to store persistent information. For example, saving customer information, passwords, emails, purchases or user behavior data. In video games, databases could be used to save player progress, items, and achievements. For multi-user server applications, it is critical to have an efficient database system, as the overall server performance is dependent on it. It is difficult to migrate databases data to another database solution, once the application is in use or update the database. This is why it is important to make a solid foundation for the database system in the first place. It is essential to research different options to be able to choose the most suitable database system for the application.

Databases can have complex data structures. For example, a certain piece of data is tied to another piece of data, which is tied to something else and together they form a complex structure that spans across the database. Sometimes this is necessary and has performance advantages, however overly complex structures can lead to difficulties.

6.1 Database trends

According to the database ranking on DB-Engines the top 5 most relevant databases in November 2015 are Oracle, MySQL, Microsoft SQL Server, MongoDB and PostgreSQL, as listed in Table 1. Databases that have the most positive trends from best to worst are MongoDB, Oracle, PostgreSQL and MySQL. Microsoft SQL server has been losing traction, while MongoDB has clearly been the most positively trending. Still MongoDB only has around fourth of MySQL’s traction.

Table 1 – List of top ranking databases and their DB-Engines scores (DB-Engines 2015a; DB-Engines 2015c).

Rank Nov 2015	**DBMS**	**Database Model**	**Score Nov 2015**	**Score Δ Nov 2014**	**Score Δ Nov 2012**
1	Oracle	Relational DBMS	1480.95	+28.82	-35.61
2	MySQL	Relational DBMS	1286.84	+7.77	+13.37
3	Microsoft SQL Server	Relational DBMS	1122.33	-97.87	-127.51
4	MongoDB	Document store	304.61	+59.87	+202.84
5	PostgreSQL	Relational DBMS	285.69	+28.33	+90.62

The only proprietary database solutions on the list are Oracle and Microsoft SQL Server. MongoDB, MySQL and PostgreSQL are open source. Anyone can integrate them into their software. In general, the trend has been in favor of open source solutions. In January 2013, the market share for open source database solutions was 35.53% and for commercial 64.47%. To this day, the open source solutions have been slow increasing their market share while commercial licenses have been losing traction. In November 2015, the market share for open source database solutions was 44.13% and for commercial 55.87%. (DB-Engines 2015b.) The chart above reflects this as in the past three years all of the commercial databases in the top five have lost traction while all of the non-commercial ones have gained traction. Some of them large amounts, for example MongoDB and PostgreSQL. (DB-Engines 2015c.)

This is how to the scores on DB-Engines are calculated:

- Number of mentions of the system on websites is gathered using Google and Bing.
- Data on general interest in the system is gathered from Google Trends.
- Frequency of technical discussion about the system on Stack Overflow and DBA Stack Exchange.

- Number of mentions of the system in job offers on websites Indeed and Simply Hired.
- Number of profiles in professional networks, in which the system is mentioned. Information is gathered from LinkedIn.
- Relevance in social networks is calculated from how many times the database system is mentioned in tweets.

The exact way the final score is formulated is not told. (DB-Engines 2015d.)

It is important to note that this trend comparison does not tell everything. For example, school assignments, personal projects and other non-commercial ideas could be inflating MySQL. Often times schools cannot afford to teach proprietary software, which could exaggerate open-source solutions popularity. Other factors could be altering the statistics in addition.

6.2 Relations

There can be multiple tables in a relational database. There is no set maximum for tables in MySQL (dev.mysql 2015). As the database is relational, these tables are connected with different kind of relations. For example, by a one to one, one to many, or many to many relation (code.tutsplus 2010). There are other relations however these are most common.

One to one relationship represents a relation where two tables are connected and there is only one instance of both tables. For example, there is a table called Earth and a table called Milky Way. These two tables contain information and are connected with a one to one relationship. Meaning there can be only one Milky Way and Earth table. (Code.Tutsplus 2010).

One to many relationships represents a connection between two tables where there is one instance of the other connected table and many of the other table connected. For example, there are two tables. One is called Sun and the other is called planet. There can only be one instance of the Sun although Sun can have multiple instances of planets. (Code.Tutsplus 2010).

Many to many relationship represents a connection between two tables where there are many instances of both tables. For example, a connection between two tables where the other one is “orders” and the other one “products.” One product can have multiple orders and one order can have multiple products. This typically requires an extra table to connect the two tables. In the case of “products” and “orders” tables many to many connection it is possible to have a table called “orderDetails” where the orders and products are linked to each other by their respective ID’s. (Code.Tutsplus 2010).

Relational databases are used for reducing redundant data and increasing performance and flexibility. Relational data reinforces itself by applying constraints to the data. The constraints can be relational, for example, if this record exists in this table, in addition it must exist in the same form in the other connected table. SQL Should be chosen for situations where it is critical that the data is saved correctly. (DigitalOcean 2014a.)

6.3 Normalized and denormalized

Normalized means that instead of having large tables of data, data is divided into multiple smaller tables. The divided data in tables can then be connected with relations to make it complete again. Since SQL databases are relational they are well suited for normalization.

Normalizing reduces the amount of duplicate data, which makes update and write functions faster as the data only has to be written or updated in fewer places to the database. In a denormalized database, the data could exist in multiple different parts of the database requiring the data to be written as many times as there are locations for it. Since the tables are smaller, this means that accessing them is less memory intensive and as a result faster. The disadvantage of normalized databases is that when reading a set of data it has to be gathered from multiple tables. Joining the information contained in multiple tables makes the whole process slower. (Madhu-dwh.blogspot 2013.) MySQL seems to be especially slow at joining, as it only supports nested loop joins (use-the-index-luke 2013).

Denormalized databases stand for databases that have larger tables of data. The data is not split into multiple related tables. When the data is stored in large tables instead of splitting them into smaller ones this results in data duplication. This makes updating and writing slower as the data has to be updated in multiple places in the database, instead of just one like in normalized databases. The advantage of denomalizing is that it is not necessary to go through many tables of data to gather a set of information. This means less heavy join actions and as a result increased read performance. (Madhu-dwh.blogspot 2013.)

In conclusion, normalized databases should be used in situations where plenty of write and update actions are used and denormalized in cases where mainly read actions are used.

6.4 Scaling out or up

When adding nodes or servers to scale the service, it is called horizontal scaling, or scaling out. For example, with MongoDB the recommended method for scaling is scaling out (MongoDB 2015b). MongoDB allows for seamless scaling of the database by adding more nodes. In addition, this can be done with other databases whether they are SQL or NoSQL databases. Adding more nodes allows storing more data, increased performance and splitting the data across multiple servers. (MongoDB 2015a.) This can make the server more reliable, since if one of the servers break, the others will fill in. Scaling with this approach can be cheaper, compared to other solutions (Vtagion 2013).

The other option is vertical scaling or scaling up, which refers to buying more powerful server hardware, or upgrading the existing system. Scaling up can be limiting as after a certain point more powerful hardware does not exist, or the existing system does not allow for more upgrades. Adding or replacing hardware requires server downtime, while the upgrades are done. Scaling up requires investing into high-end hardware, to replace old hardware. This results in having old hardware, which may not be able to be utilized, and having to invest into expensive hardware that is adequate for a long time. This upgrade process can cost millions for large companies. (Vtagion 2013.) Previously MongoDB has been more reliant on scaling up, although with the release of 3.0, MongoDB is able to better utilize scaling out (ReadWrite 2015).

6.5 SQL

Standard Query Language (SQL) is used with relational database management systems (RDBMS) also known as SQL databases. RDBMS database solutions are for example Microsoft Access, MySQL, PostgreSQL, and SQLite (Wikipedia 2015h). These support different features, although because they all utilize SQL the code is similar. MariaDB and MySQL are especially similar and very compatible (MariaDB 2015). MariaDB was branched from MySQL (Wikipedia 2015i).

Relational databases that use SQL for data retrieval like the SQLite, MySQL and Microsoft Access store data in tables. A relational database table consists of rows and columns. Together rows and columns form tables. Table 2 is an example of a generic relational database table. Columns are sometimes called “attributes” and rows “tuples” or “records”. Among developer’s rows and columns are the more commonly used terms. (www-01.ibm 2015.)

Table 2 - Example of a relational table with four (4) columns and three (3) rows.

ID	FIRSTNAME	LASTNAME	ADDRESS
1	Bingo	Bango	Mannerheimintie 5
2	Erkki	Esimerkki	Nottinghamstreet 12 A
3	John	Smith	Boulevardstreet 22 B a

Each row can be edited, deleted or more can be added at any time. Rows must be unique in that they have at least one column where data is different from the other rows. Otherwise, it will be a duplicate and cause problems. To solve this an ID is usually added for each row for ease of access and making the rows unique. The ID can be any combination of characters, although most commonly and auto incremental number is used. ID Columns names have to be unique. (www-01.ibm 2015.)

6.5.1 MySQL

MySQL is the most popular open-source RDBMS (DB-Engines 2015a). MySQL is feature rich and it is considered a one size fits all solution, and while it is possible to accomplish almost anything with MySQL, the performance might suffer with certain solutions. As always it is largely about the implementation of the database and not the database itself. MySQL especially allows for immense configuration and customization.

MySQL supports all modern operating systems including Windows, various Linux operating systems and Apple OS X from 10.9 forwards (MySQL 2015a). MySQL can be used with most programming languages, for example PHP, Perl, Python, Ruby, Java, C#, C++, JavaScript, Node.js and others (MySQL 2015b).

6.5.2 SQLite

SQLite is an open-source, relational database. SQLite does not have user management and as a result is not suited for multi-user clients or servers. Consequently, SQLite is usually embedded within a client to handle the client’s local data. (DigitalOcean 2014b.) A better-suited database solution for a server is MySQL or PostgreSQL that has user management (Postgresql 2015). At the time of writing this thesis, the latest version is 3.9.2 (SQLite 2015a).

A limitation of SQLite is that it only supports one writer at a time per database file. Write transactions usually only take milliseconds, therefore writers can take turns. This is an acceptable solution and only becomes a problem with very high amount of users. A SQLite database is limited to 140 terabytes in size and SQLite stores the entire database in a single disk file. In large dataset situations, it is usually better to distribute the database into multiple across multiple disk files and across multiple disks. (SQLite 2015b.)

Best use cases for SQLite are device-local storage with low writer concurrency and databases with less than a terabyte of data. SQLite is saved into a single file, which makes it very portable, easy to inspect and to manage. It is intentionally kept minimal and simple, which makes it more stable and faster to develop with. (SQLite 2015b.) For example, use it on a client to store local data in a mobile application, or on a personal computer.

6.5.3 PostgreSQL

PostgreSQL is an open-source RDBMS database system. PostgreSQL ranks a little under MongoDB, in DB-Engines websites ranking. At the time of writing this thesis, it is far less popular, although has certain advantages and features over MySQL. (DB-Engines 2015c.) At the time of writing this thesis, the latest version of PostgreSQL for Ubuntu is 9.4 (PostgreSQL 2015.)

PostgreSQL is similar to MySQL, the main differences being that PostgreSQL was made to be standards-compliant and extensible, whereas MySQL is not fully SQL compliant. SQL compliance stand for, how strictly the system adheres to the SQL standards. SQL is part of the American National Standards Institute (ANSI) and part of the International Organization for Standardization (ISO). (Wikipedia 2015j.)

As PostgreSQL is similar and as feature rich as MySQL, it can be used in similar situations and has similar advantages. It features data integrity features, complex, custom procedures, integration, complex designs, while it is lacking in in fast read operations, simple set ups and Replication.

Data integrity represents the overall completeness, accuracy and consistency of data. A database system can reinforce the databases data integrity, for example, by only accepting certain kind of data to be saved into a row or column. (techopedia 2016). Typically, SQL databases have more integrity reinforcing features natively than NoSQL databases.

Integration support means being able to migrate into other database solutions. PostgreSQL is fine for smaller solutions, although if the application reaches millions of users, or stores large amounts of data, integration should be done. PostgreSQL has excellent integration support. (DigitalOcean 2014b.)

6.6 NoSQL

NoSQL represents databases that do not utilize Standard Query Language (SQL). NoSQL is equal performance and scalability vice compared to relational database models. NoSQL does not require developers to design schemas to the same extent as with RDBMS and SQL databases. This reduces development time. In addition, NoSQL allows turning object-oriented programming objects into documents and back more efficiently than RDBMS or SQL. NoSQL databases store data as documents instead of tables.

There are several database types that can be categorized as NoSQL. These database types are the following:

- Document databases pair every key with a data structures which are called documents. Documents, depending on implementation, are able to contain key-value pairs, key-array pairs or nested documents.
- Graph stores use graph structures to store data, about networks, social connections and customers for example. Graph stores include Neo4J and HyperGraphDB

- Wide-column stores as the name suggests, support a high amount of columns for large datasets. For example, HBase and Cassandra are wide-column stores.
- Key-value stores simply pair each value with a key. For example, Riak, Voldemort and Redis databases are key-value stores. This is a similar structure to maps in programming languages like Java. (MongoDB 2015d.)

6.6.1 MongoDB

MongoDB is the most relevant NoSQL database at the time of writing this thesis. Only being behind Oracle, MySQL, Microsoft SQL Server and being above PostgreSQL, Microsoft Access and SQLite. (DB-Engines 2015a.) MongoDB was released in the year 2009 and is the fourth popular option for databases (Wikipedia 2015k). MongoDB is growing fast and continuing to do so. MongoDB is open source (docs.MongoDB 2015).

Due to MongoDB’s large user base, it has a large online community and plenty of teaching material. Something that less popular database systems do not necessarily have. MongoDB has been praised for its ease of use, excellent development tools and complete documentation. (Infoworld 2013b.) These help in learning MongoDB and learning MongoDB is likely the easiest approach to learning NoSQL databases.

However, there is plenty of debate online about the performance of MongoDB, especially with high amount of users. It is hard to find reliable up to date data from neutral sources about the performance, and even then, it is always questionable, whether the method of testing actually reflects realistic usage cases. The testing of database system performance, especially for high user amount cases, requires resources. This means that the performance testing is dependent on large corporations, which aren’t always reliable. Large corporations like Facebook, google, EA, eBay, Squarespace and The Washington Post utilize MongoDB.

Because of this the performance is likely good enough in situations with large user bases (MongoDB 2015c).

6.6.2 HBase

Apache HBase is rank 15 in DB-Engines websites overall ranking. HBase is an open-source NoSQL database solution, or more accurately utilizes a wide column store database model. (DB-Engines 2015a.) Companies that have had HBase in production for over three years include Trend Micro, EBay, Yahoo!, Facebook, RocketFuel, and Flurry (Radar.oreilly 2014).

Wide column store database model means that it stores data in records, with the ability to have large amount of dynamic columns. In addition, Wide column stores are schema-free, just like document stores. The first wide column store is considered Google’s BigTable. (DB-Engines 2015e.)

HBase does not replace relational database model solutions (RDBMS). HBase is the most useful when the data being saved to the database varies to a degree that it is difficult to make a schema. HBase provides a key-based access to data when storing and retrieving. HBase is able to handle large amounts of data efficiently. (blog.cloudera 2011.)

6.6.3 Couchbase

The Couchbase database solution was published in January 2012 by Couchbase incorporation. Couchbase is an open source project. (Wikipedia 2015l.) In DB-Engines database popularity, ranking Couchbase is ranked at 24 (DB-Engines 2015a). Companies that use Couchbase include LinkedIn, Concur, Millenial Media, Mirror Image, Amadeus, Ryanair and Willis (Couchbase 2015a).

Couchbase has support for user management and schema free content management. Couchbase is available for mobile and personal computers

(Couchbase 2015b). MongoDB would likely be an easier entrance to the NoSQL world, thanks to the more user-friendly development tools and large community (Infoworld 2013b).

6.7 Database choice for Dibs

The best database model depends on the application. SQL databases such as MySQL and PostgreSQL offer better data-validation, more planned data and complex data structures. This can make the database solution more stable, as a slightly wrong insertion of data throws an error instead of the database accepting it and providing problems later on. The disadvantage of this is that that development times will be longer as it requires more planning and designing databases can be time consuming, especially since it is required to pre-define each datatype that is saved into the database. DRBMS or SQL databases therefore will likely be better for final implementation and not for quick prototyping of various database reliant applications. With relational data, it is easier to avoid duplicate data, as it is possible to have a single entity of the data and link it to all the tables where it is needed.

The easiest SQL solution to learn is MySQL, as it is the most popular and as a result has a large community with many tutorials. In addition, the commands used with MySQL are more intuitive than in some other SQL languages, for example in PostgreSQL. This is why MySQL is likely the easiest SQL database for learning and practicing SQL databases. On the other hand, PostgreSQL offers easier transitions to other database solutions once the application has overgrown its last database model and additionally arguably better performance.

NoSQL has the opposite advantages to SQL. Since NoSQL is not reliant on elaborate schemas, or is not that precise with the data types that are saved into it, it allows for more rapid development. This also means that the data is not as validated and consequently there is a higher chance of incorrect data compared to RDBMS. It is possible to implement relations in NoSQL, although it is not native and a RDBMS should likely be used instead.

7 Conclusion

7.1 Summary

At the beginning of the thesis it was established that one technology from each technology category is chosen from the following list to fully realize Dibs and all of its features:

- server / web host
- protocol
- programming language
- database

Programming language was chosen to be Node.js. JavaScript was personally familiar to me, which is what Node.js is built upon. This made the transition to Node.js easier, while Node.js still offered new and interesting things to learn. Node.js has been used for large websites and services and it is capable of performance and scalability. The programming language chosen comes heavily down to the personal preference and history of the developers, as all of the server-side programming languages are capable of realizing the server-side software of Dibs. At the time of writing this thesis, Node.js is experiencing growth.

Node.js came off as having very strong scalability support, and the capability to run large web services efficiently, as mentioned in the Node.js chapter. In addition, Node.js supports MongoDB very well, and they are often recommended to be used together. On the client side of Dibs, a programming language called UnityScript is used, which is similar to JavaScript. Consequently, the technologies used for Dibs are more uniform, reducing stress of developers caused by handling and learning multiple different languages.

The choice for databases is MongoDB. MongoDB is a NoSQL based database system that allows for rapid application development. It is the most relevant NoSQL database system and it is maintaining a positive growth trend. This makes MongoDB talent useful in the job market. Since dibs developers are in the early stage of development of the Dibs assignment and have little users, performance is not the number one priority. Instead Dibs developers want to focus on faster development times and the ability to make rapid changes. MongoDB offers this with very easy to use tools and the less strict schema planning.

The best Internet protocol for communication between the server and the client is the websocket protocol. Websocket is a full-duplex near real time communication protocol. Full duplex means that it allows for both ways communication. Once the connection is established it does not have to be re-established every time a data is sent. HTTP and other HTTP based protocols usually require this. Instead Websockets open the connection once and maintain it. Websockets are necessary for real time games and game like software like Dibs.

A web host or server hardware is something for developers to run their server software on. Basically it could be any computer with the appropriate access to the Internet, although more modern options are available, for example Platform as a Service (PaaS). PaaS is a virtual web hosting platform, where developers can host their application software and websites. PaaS offers fast deployment, a plethora of tools and plugins to ease development and scaling of the available computing resources. All of these make development easier and faster compared to other services like IaaS or other web hosting options.

Web hosts are the last thing chosen for the Dibs assignment and as a result it is necessary to make sure it supports all of the chosen technologies. In this case they are Node.js, MongoDB and websocket. A PaaS called Heroku offers support for all of these, and in addition an excellent user interface and access to tools. It is a popular web hosting PaaS, and as a result has plenty of support, plugins and other materials available online. Heroku is not the cheapest option, although the upkeep costs for a small server are still minimal, and with the powerful tools of Heroku, the developers save time and money.

The combination of Node.js, MongoDB, websocket and Heroku offer the most modern solution for Dibs assignment. These tools make it possible to have rapid development, changes and to make prototypes quickly, while still offering good performance and scalability. They have active, innovative communities with the enthusiasm of people driving new technologies and new solutions. They are not the most tried and proven, however they still have plenty of success stories and large services behind them.

7.2 Dibs

During the writing of this thesis two prototypes of Dibs were made. Neither of the prototypes were released.

The first prototypes development was started in early August and its development was ended in November. During this time a simple client was made with Unity and the server side was deployed to a PaaS called OpenShift. The database utilized at the time was MySQL, and the server utilized the Node.js programming language. Communication between the client and the server was realized with websockets. At the time the reason for choosing MySQL and OpenShift was mainly the personal familiarity with the technologies, and I had not done the research yet on the various technologies. I had no experience with server-side programming languages, and at the time Node.js was personally most interesting.

The most time consuming part of developing the first prototypes server side was the MySQL database and accessing it with Node.js. MySQL and Node.js are a rare combination which made finding learning material for the combination difficult. There was next to none examples online on utilizing MySQL with Node.js, and I had to frequently rely on Stack Overflow, a very useful website for getting help from other software developers. There would have been plenty of learning material available for MongoDB with Node.js. Consequently, utilizing MongoDB instead of MySQL would have been a better choice, because of the more available learning material and because NoSQL databases like MongoDB allow for faster development.

The reasoning behind using MySQL in developing the first prototypes server side was because it would have provided better performance in social applications, with high number of users. In social applications everything is inherently connected, through friends, messaging, interests and so on. Because MySQL is a relational database where all of the data is inherently connected, it is well suited for social applications. Because the software was in early development with little to none users, MongoDB would have allowed for faster concept validation and testing. Which is far more important in early development than worrying about performance with high amount of users. As a result, plenty of time was wasted in features that nobody ended up using.

The second prototypes development was started in January. A stricter schedule was utilized for the second prototype and a greatly reduced feature list. The idea was to get the first prototype into a testable state in a month, and this required omitting the server side entirely, instead relying on other services to share the cards created with the Dibs client.

Consequently, this resulted in an impressive looking and a very usable software in a very short time, which still provided the core idea of Dibs, and a good foundation to continue building Dibs on. Eventually sprint by sprint features would have been added, including server side functionality. In the first prototype we ended up stepping ahead of ourselves, being feature greedy. Instead we should have concentrated on the core essence of what the user would get out of the idea, creating cards.

As a result of boiling the idea of Dibs down to its very essence, me and the people developing Dibs, realized what it actually was and realized it did not align with their core values. As a result, developing Dibs was abandoned, in favor of other pursuits. The journey of developing Dibs taught plenty, both in software development and personal growth.

7.3 Discussion

The reason why the paper does not go into further detail is mainly because of the comprehensive subject area of the thesis. The span of this thesis includes six different databases, five programming languages, two protocols and two types of server platforms. This is by any standards a very considerable amount of technologies to study in a single paper.

Consequently, this resulted in the analysis boiling down to the absolute basics and easy to understand evaluation criteria for each technology. In addition, analysis that would have required considerable time was not emphasized. Interestingly the evaluation criteria that was not easily available for various technologies was performance. Developers might consider this to be the most important comparison criteria, although consistently, in various sources, it did not come up at all. I would assume this is either because it is extremely hard to evaluate, or the performance is similar between the technologies in each category. Delving into more detail on performance evaluation between various technologies would have made this paper more interesting for data-processing academics, but a waste of time for result-oriented developers.

It is disappointing that a clear distinction was not able to be made between certain technologies. Some of the technologies presented are distinctly similar, and as a result, would require more in-depth analysis to recommend one over the other. For example, MySQL compared to PostgreSQL, MongoDB compared to Couchbase and a distinction between most of the programming languages. Because a distinction was not able to made though, it is indicative of the fact that the differences are minimal and do not necessarily concern junior developers.

The research the developer should expand on depends on the developer's position in the development team, and the nature of the company the developer is in. If the developer is part of a larger team, specialization into as few as possible technology categories will likely yield better results than a developer who tries to do everything. In small companies specialization is not always possible. There are plenty of startup company situations where a single developer is handling solely all of the server side technologies, including databases, protocols and the core of the server done with the developer’s choice of programming language. Consequently, for these developers, who do not have the time to research or learn completely new technologies, the results of this study are likely calming, as it was concluded that any of the popular technologies are capable of realizing successful solutions.

REFERENCES

Akamai. 2015. HTTP/2 is the future of the Web, and it is already here!
https://http2.akamai.com/. 1.12.2015

AllAboutCookies.org. 2015. What is a Protocol?
http://www.allaboutcookies.org/faqs/protocol.html. 1.12.2015

Aptana. 2015. Aptana Studio 3
http://www.aptana.com/index.html. 17.12.2015

blog.cloudera. 2011. Apache HBase Do's and Don'ts
https://blog.cloudera.com/blog/2011/04/hbase-dos-and-donts/.
25.11.2015

Broadcast.oreilly. 2008. On Why I Don't Like Auto-Scaling in the Cloud
http://broadcast.oreilly.com/2008/12/why-i-dont-like-cloud-auto-scaling.html. 17.12.2015

C9. 2015. Powerful workspaces
https://c9.io/. 17.12.2015

Caniuse. 2015. HTTP/2 protocol
http://caniuse.com/#feat=http2.1.12.2015

CodeCondo. 2014. Top 5 Ruby IDE Solutions for Web developers
http://codecondo.com/top-5-ruby-ide-solutions-development-pleasure/.
6.12.2015

code.tutsplus. 2010. SQL for Beginners: Part 3 – Database Relationships
http://code.tutsplus.com/articles/sql-for-beginners-part-3-database-relationships--net-8561. 13.10.2015

Computerworld. 2002. The Story So Far
http://www.computerworld.com/article/2576978/enterprise-applications/the-story-so-far.html. 16.11.2015

Confluence Atlassian. 2015. Server Hardware Requirements Guide
https://confluence.atlassian.com/doc/server-hardware-requirements-guide-30736403.html. 14.10.2015

Couchbase. 2015a. Customers
http://www.couchbase.com/case-studies. 27.10.2015

Couchbase. 2015b. Use Cases
http://www.couchbase.com/use-cases. 27.10.2015

DB-Engines. 2015a. DB-Engines Ranking
http://db-engines.com/en/ranking. 20.10.2015

DB-Engines. 2015b. Popularity of open source DBMS versus DBMS
http://db-engines.com/en/ranking_osvsc. 20.10.2015

DB-Engines. 2015c. DB-Engines Ranking – Trend Popularity
http://db-engines.com/en/ranking_trend. 20.10.2015

DB-Engines. 2015d. Method of calculating the scores of the DB-Engines Ranking
http://db-engines.com/en/ranking_definition. 20.10.2015

DB-Engines. 2015e. Wide Column Stores
http://db-engines.com/en/article/Wide+Column+Stores. 25.11.2015

Developers.RedHat. 2015. OpenShift Online Plans
https://openshift.redhat.com/app/account/plan. 24.9.2015

dev.mysql. 2015. C.10.2 Limits on Number of Databases and Tables
https://dev.mysql.com/doc/refman/5.5/en/database-count-limit.html. 11.10.2015

DigitalOcean. 2014a. Understanding SQL And NoSQL Databases And Different Database Models
https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models. 20.10.2015

DigitalOcean. 2014b. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems
https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. 23.11.2015

docs.MongoDB. 2015. The MongoDB 3.0 Manual
https://docs.mongodb.org/manual/?_ga=1.43320678.270217770.1448172801. 22.11.2015

DrJava. 2015. About DrJava
http://www.drjava.org/. 17.12.2015

Eclipse. 2015. Mars Release
https://eclipse.org/. 17.12.2015

FreeJavaGuide. 2015. History
http://www.freejavaguide.com/history.html. 5.12.2015

Plus.Google. 2012. How many servers does Google have?

https://plus.google.com/+JamesPearn/posts/VaQu9sNxJuY. 17.10.2015

GitHub, 2015a. HTTP/2 Frequently Asked Questions
https://http2.github.io/faq/#why-is-http2-binary. 18.10.2015

GitHub, 2015b. shahriar / Java simple server
https://gist.github.com/shahriar/b529cf8d64ddd45d0c92. 19.4.2016

HireFire, 2015. Autoscaling for your Heroku dynos
https://www.hirefire.io/. 24.2.2016

Infoworld. 2013a. 9 killer uses for WebSockets
http://www.infoworld.com/article/2609720/application-development/9-killer-uses-for-websockets.htm. 1.12.2015

Infoworld. 2013b. NoSQL showdown: MongoDB vs. Couchbase
http://www.infoworld.com/article/2613970/nosql/nosql-showdown--mongodb-vs--couchbase.html. 25.10.2015

InteRoute. 2015. What is IaaS?
http://www.interoute.fi/what-iaas. 18.12.2015

JDeveloper. 2015. Oracle JDeveloper
http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html. 17.12.2015

JetBrains. 2015a. IntelliJIDEA The Most Intelligent Java IDE
https://www.jetbrains.com/idea/#chooseYourEdition. 17.12.2015

JetBrains. 2015b. WebStorm The smartest JavaScript IDE
https://www.jetbrains.com/webstorm/. 17.12.2015

JetBrains. 2015c. PhpStorm Lightning-smart PHP IDE
https://www.jetbrains.com/phpstorm/. 17.12.2015

ITU. 2015. MDGs 2000-2015: ICT revolution and remaining gaps
http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx.12.12.2015

JMarshall. 2012. HTTP Made Really Easy
https://www.jmarshall.com/easy/http/. 17.12.2015

KomodoIDE. 2015. One IDE, All Your Languages
http://komodoide.com/. 17.12.2015

Madhu-dwh.blogspot. 2013. Data Warehouse
http://madhu-dwh.blogspot.fi/2013/01/data-warehouse.html. 20.10.2015

MariaDB. 2015. MariaDB is a binary drop in replacement for MySQL https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-compatibility/. 23.11.2015

Meteoriitti. 2013. Ketteryys haltuun: Scrum pähkinänkuoressa https://www.meteoriitti.com/Artikkelisarjat/Ketteryys-haltuun/Ketteryys-haltuun-Scrum-pahkinankuoressa/. 22.11.2015

MySQL. 2015a. Supported Platforms: MySQL Database https://www.mysql.com/support/supportedplatforms/database.html. 22.11.2015

MySQL. 2015b. Frequently Asked Questions http://www.mysql.com/about/faq/. 22.11.2015

MongoDB. 2015a. MongoDB at scale https://www.mongodb.com/mongodb-scale. 22.11.2015

MongoDBb. 2015b. Performance Best Practices for MongoDB https://s3.amazonaws.com/info-mongodb-com/MongoDB-Performance-Best-Practices.pdf. 23.11.2015

MongoDB. 2015c. Who uses MongoDB https://www.mongodb.com/who-uses-mongodb. 26.11.2015

MongoDB. 2015d. NoSQL Databases Explained https://www.mongodb.com/nosql-explained. 26.11.2015

MpSoftware. 2015. Welcome to phpDesigner 8 http://www.mpsoftware.dk/phpdesigner.php. 17.12.2015

NetBeans. 2015. NetBeans IDE https://netbeans.org/. 17.12.2015

Nodejs. 2015a. About https://nodejs.org/en/about/. 6.12.2015

Nodejs. 2011b. Node v0.5 Roadmap https://nodejs.org/static/documents/nodeconf.pdf. 30.4.2016

PHP. 2015. What is PHP? http://php.net/manual/en/intro-whatis.php. 5.12.2015

Postgresql. 2015. Chapter 19. Database Roles and Privileges http://www.postgresql.org/docs/8.3/static/user-manag.html. 23.11.2015

Pubnub. 2015. REST vs. WEBSOCKETS https://www.pubnub.com/blog/2015-01-05-websockets-vs-rest-api-understanding-the-difference/. 2.12.2015

Radar.oreilly. 2014. 5 Fun Facts about HBase that you didn’t know
http://radar.oreilly.com/2014/04/5-fun-facts-about-hbase-that-you-didnt-know.html. 25.11.2015

ReadWrite. 2015. How The MongoDB Database Learned To Scale
http://readwrite.com/2015/02/09/mongodb-scale-document-database-nosql. 23.11.2015

Sandbox.mc. 2015. While Loop
http://sandbox.mc.edu/~bennet/ruby/code/wh1_rb.html. 7.12.2015

Scala-Lang. 2012. Scala in the Enterprise
http://www.scala-lang.org/old/node/1658. 1.12.2015

SearchCloudComputing. 2015. Platform as a Service (PaaS) definition
http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS. 18.12.2015

SitePoint. 2015a. What’s the Best Programming Language to Learn in 2015
http://www.sitepoint.com/whats-best-programming-language-learn-2015/.
4.12.2015

SitePoint. 2015b. Best PHP IDE in 2014 – Survey Results
http://www.sitepoint.com/best-php-ide-2014-survey-results/. 5.12.2015

SkillCrush. 2015. 37 Sites You LOVE Built With Ruby On Rails
http://skillcrush.com/2015/02/02/37-rails-sites/. 7.12.2015

SQLite. 2015a. SQLite Download Page
https://www.sqlite.org/download.html. 24.11.2015

SQLite. 2015b. Appropriate Uses For SQLite
https://www.sqlite.org/whentouse.html. 23.11.2015

Sublime. 2015. Sublime Text
http://www.sublimetext.com/. 17.12.2015

Stack Overflow. 2015a. What are long-Polling, Websockets, Server-sent Events (SSE) and Comet?
http://stackoverflow.com/questions/11077857/what-are-long-polling-websockets-server-sent-events-sse-and-comet. 4.12.2015

Stack Overflow. 2015b. Scala equivalent of python echo server client example
http://stackoverflow.com/questions/6414942/scala-equivalent-of-python-echo-server-client-example#comment7599726_6416755. 6.10.2015

Techopedia. 2016. Data Integrity

https://www.techopedia.com/definition/811/data-integrity-databases. 6.6.2016

TheNewBoston. 2012. Computer Networking Tutorial – 10 – What is a protocol? https://www.youtube.com/watch?v=VlKks__ZhI0&index=10&list=PL6gx4Cwl9DGBpuvPW0aHa7mKdn_k9SPKO. 14.10.2015.

TomsITPro. 2014. IaaS Providers List: Comparison And Guide http://www.tomsitpro.com/articles/iaas-providers,1-1560.html. 20.12.2015

Tutorialspoint. 2015a. HTTP – Requests http://www.tutorialspoint.com/http/http_requests.htm. 1.12.2015

Tutorialspoint. 2015b. HTTP – Responses http://www.tutorialspoint.com/http/http_responses.htm. 1.12.2015

Tutorialspoint. 2015c. HTTP – Overview http://www.tutorialspoint.com/http/http_overview.htm. 1.12.2015

use-the-index-luke. 2013. MongoDB is to NoSQL like MySQL to SQL – in the most harmful way http://use-the-index-luke.com/blog/2013-10-01/mysql-is-to-sql-like-mongodb-to-nosql. 22.11.2015

Vtagion. 2013. Scalablity: Scale-up or Scale-out, What it is and Why You Scould Care http://www.vtagion.com/scalability-scale-up-scale-out-care/. 23.11.2015

W3techs. 2015. Usage of server-side programming languages for websites http://w3techs.com/technologies/overview/programming_language/all

WebFoundation. 2015. History of the Web http://webfoundation.org/about/vision/history-of-the-web/. 1.12.2015

Webopedia. 2015. Hypertext http://www.webopedia.com/TERM/H/hypertext.html. 1.12.2015

Websitesetup. 2015. WHAT IS WEB HOSTING? SHARED, VPS, DEDICATED & CLOUD COMPARISON http://websitesetup.org/what-is-web-hosting/. 17.12.2015

Whatis.TechTarget. 2014. Server http://whatis.techtarget.com/definition/server. 1.12.2015

Wikipedia. 2015a. History of the Internet https://en.wikipedia.org/wiki/History_of_the_Internet. 1.12.2015

Wikipedia. 2015b. Google

https://fi.wikipedia.org/wiki/Google. 1.12.2015

Wikipedia. 2015c. Hypertext Transfer Protocol
https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. 5.10.2015

Wikipedia. 2015d. Request-response
https://en.wikipedia.org/wiki/Request%E2%80%93response. 1.12.2015

Wikipedia. 2015e. PHP
https://en.wikipedia.org/wiki/PHP. 5.12.2015

Wikipedia. 2015f. Ruby (programming language)
https://en.wikipedia.org/wiki/Ruby_(programming_language). 6.12.2015

Wikipedia. 2015g. Comparison of integrated development environments
https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments#Ruby. 17.12.2015

Wikipedia. 2015h. List of relational database management systems
https://en.wikipedia.org/wiki/List_of_relational_database_management_systems. 11.10.2015

Wikipedia. 2015i. MariaDB
https://fi.wikipedia.org/wiki/MariaDB. 25.11.2015

Wikipedia. 2015j. SQL
https://en.wikipedia.org/wiki/SQL. 6.10.2015

Wikipedia. 2015k. MongoDB
https://en.wikipedia.org/wiki/MongoDB. 23.9.2015

Wikipedia. 2015l. Couchbase, Inc
https://en.wikipedia.org/wiki/Couchbase,_Inc. 25.11.2015

www-01.ibm. 2015. Tables, rows, and columns
http://www-01.ibm.com/support/knowledgecenter/SSPK3V_6.3.0/com.ibm.swg.im.soliddb.sql.doc/doc/tables.rows.and.columns.html. 11.10.2015

www8. 2015. Key Differences between HTTP/1.0 and HTTP/1.1
http://www8.org/w8-papers/5c-protocols/key/key.html. 1.12.2015

Zend. 2015. The PHP IDE for Smarter Development
http://www.zend.com/en/products/studio. 17.12.2015