ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

INTERNSHIP REPORT

# Comparison of ReactJs and Angular

*Author:*
Kleio Fragkedaki

*Supervisor:*
Prof. Panagiotis Louridas

*Internship report and Thesis submitted as part of Bachelor degree*

*in the*

Department of Management Science and Technology

June 14, 2019

# Contents

# List of Abbreviations

| | |
|---|---|
| **CEO** | **C**hief **E**xecutive **O**fficer |
| **SMT** | **S**enior **M**anagement **T**eam |
| **B2B** | **B**usiness To **B**usiness |
| **B2C** | **B**usiness To **C**ustomer |
| **CX** | **C**ustomer **E**xperience |
| **DOM** | **D**ocument **O**bject **M**odel |
| **HTML** | **H**yper **T**ext **M**arkup **L**anguage |
| **JSON** | **J**avaScript **O**bject **N**otation |
| **RPC** | **R**emote **P**rocedure **C**all |
| **SPA** | **S**ingle **P**age **A**pplication |
| **UI** | **U**ser **I**nterface |
| **URI** | **U**niform **Resource** **I**dentifier |
| **REST** | **R**epresentational **S**tate **T**ransfer |
| **MVC** | **M**odel **V**iew **C**ontroller |
| **A**jax | **A**synchronous **J**avaScript **A**nd **X**ML |
| **Framework** | Reusable software environment to build applications. |
| **JavaScript** | High-level programming language. |
| **TypeScript** | Superset of JavaScript which adds optional typing to JavaScript. |
| **Angular** | JavaScript framework maintained by Google. |
| **React** | JavaScript library maintained by Facebook. |
| **Node.js** | Run-time environment for server-side JavaScript. |
| **NPM** | Package manager for Node.js modules. |

# Part I

# Internship Report

# Chapter 1

# Introduction

As part of my Bachelor degree, I did an internship for three months in Beat, a company that started as a Greek startup about 5 years ago. From March 18th 2019 until June 18th 2019, I contributed as a Software Developer intern in several projects that was referred to a product named "BeatHotels".

## 1.1 Company Description

Beat is a company that is developing a mobile application for taxi cab and peer-to-peer-ridesharing. The app is based on the idea of establishing a direct connection between drivers and passengers by offering both sides a modern alternative to conventional booking processes. First known as a greek startup named Taxibeat, the company was founded in 2011 by Nikos Drandakis in collaboration with associates Kostis Sakkas, Nikos Damilakis and Michael Sfictos. Taxibeat was acquired in February 2017 by MyTaxi, a subsidiary of the automotive manufacturer Daimler AG, and renamed to Beat. Nowadays, Beat is part of the FreeNow group and its CEO is Nikos Drandakis. The FreeNow group is the ride hailing joint venture from Daimler and BMW, and consists of the services MyTaxi, Beat, Kapten, Clever and Hive, the e-Scooter service.

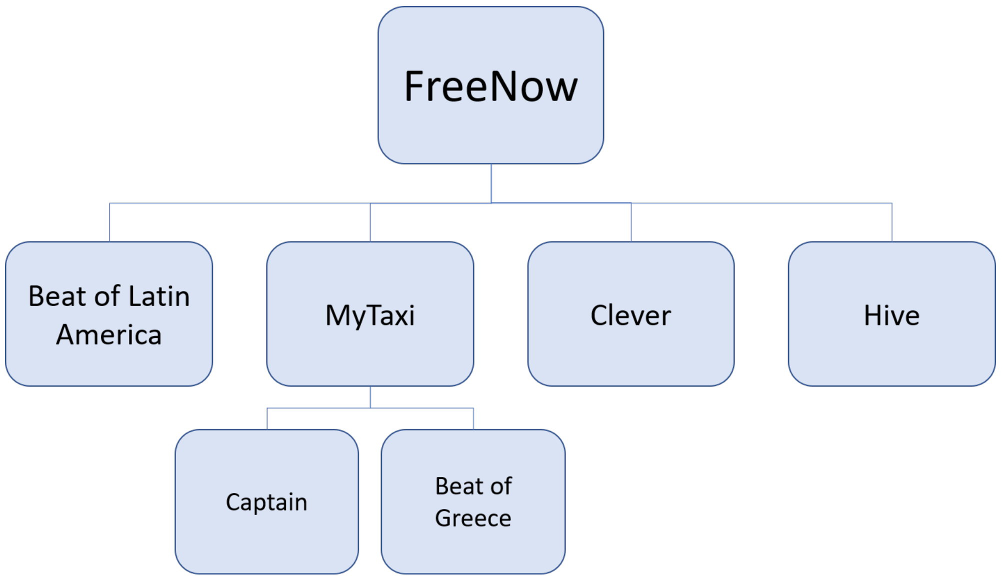


FIGURE 1.1: FreeNow's structure

Beat headquarters in Athens, Greece, while additional development and operation offices are located in Lima, Santiago, Cali, Medellín, Bogotá and Mexico City. It currently has more than 580 employees all over the world, with approximately 400 of them being in Greece. Company's teams work in small, autonomous groups of people following agile methodologies.

As regards Beat's structure, it is slightly different after its buyout. More specifically, the firm is separated in two parts based on its market targets, Beat of Latin America and Beat of Greece. This separation is because Greek market is basically part of MyTaxi group, while Latin America Market is only part of FreeNow, as shown at the driagram above. So, Beat has eight main departments, Business Operations, Finance, Engineering, Greek Market, People Operations, Marketing, Operating Office and Senior Management Team. The last department mentioned is conducted by Nikos Drandakis and his team as picture bellow reveals.

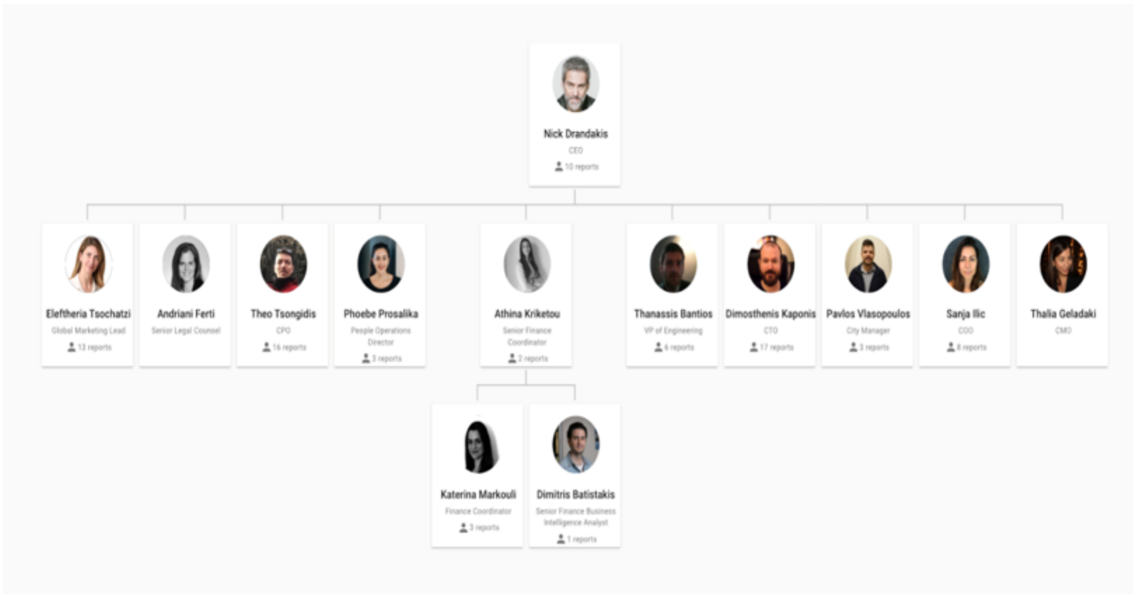


FIGURE 1.2: Beat's SMT Department

Products provided by Beat are mainly three in number. Beat App, with the extended service of peer-to-peer-ridesharing in Latin America Marketplace, Beat Hotels, B2B service only provided in Greece, and Hive, e-scooter services in Greek Market.

### 1.1.1 Beat App

Beat App is a B2C service provided in both Androids and IOS operating systems. This service is a connection between drivers and candidate passengers.

In other words, someone looking for a cup can find the closest one without any extra costs by using passenger's Beat app. Anyone that has downloaded the app can call a cup from any place, be able to see driver's rating from other Beat passengers, their personal data such as name, plates and more car details or services provided. In addition, before ride starts, an estimated price is given,

FIGURE 1.3: Beat App

list of the closest drivers is shown, and the candidate passenger has the ability to choose one of them based on the details and services provided, and to rate when ride is completed.

On the other hand, driver use another app through which the connection between two-sides is succeed. Available drivers can be located and the closest ones are shown to the candidate passenger. Driver can also accept or reject a ride, see the recommended route and see where passenger is before departure.

### 1.1.2 Beat Hotels

Beat Hotels is a B2B service that has the same aim with Beat App, the connection between candidate passengers, in this case people staying in a Hotel and request for a taxi, and drivers. The difference between these two services is, except from the aimed passengers, the virtual queues of drivers created in each Hotel.

There is a driver app, different from Beat driver's app, created in React Native and Node.js. This app is used from the driver in order to check the available Hotel queues and how much complete they are, get in a queue and start a ride.

On the other side, Hotel's have a customized dashboard, written in ReactJs and Node.js. Through this dashboard, they can call a taxi for a customer, see where the taxi is any time and get statistics of rides and revenue they have from rides completed.

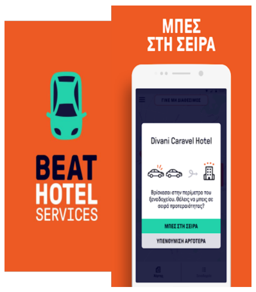


FIGURE 1.4: Beat Hotels

Finally, a similar dashboard is developed for the agents of Beat with all the informations of each Hotel that Beat's corporate. The agents have the permissions also to change the amount of a ride if it is needed, check all completed rides, or block a driver with inappropriate behavior.

### 1.1.3 Hive

Hive is an e-scooter sharing system service. Scooters are made available to use for short-term rentals and can be dropped off or picked up from arbitrary locations in the service area. Hive has an app through which a candidate user, can find where e-scooters are in the map, how much battery they have and their cost, scan the barcode of the e-scooter, unlock and rent it.

However, as regards Beat's part in Hive service is limited. Beat only provide customer experience services by receiving e-mails from users and resolving their issues occurred either from e-scooters or the app. Beat also is responsible of placing the e-scooters in the right places and charge them.

FIGURE 1.5: Hive

## 1.2 Internship Goal

As regards internship's goal, is to gain experience as a Software Developer, learning tools like React, Redux and Node.js, understanding how an application works in production. Learning how to code in Javascript and improving a web site for BeatHotel's agents by completing tasks given, and creating npm packages, were my main responsibilities as an intern.

## 1.3 Report's Structure

The internship's report is an overview of what I have been interacted with during my internship, analyzing the projects and results, skills that I have gained or used, and my role as an intern in general.

# Chapter 2

# Basic Characteristics

In the following section I will describe the basic characteristics of my Department's structure and my role as an intern.

## 2.1 Department

The Department that I am part of is the Greek Market. Greek Market is managing the marketplace of Greece and is also referred as the Beat of Greece as mentioned in the previous chapter.

### 2.1.1 Role in the Company

The Greek Market is the only department focused on Greece. Its role inside the company is to manage any demands referred to this marketplace. Demands on marketing, finance, business analysis, customer experience and the development of product BeatHotels that is provided only in Greece, are all responsibility of this department.

The Beat App is developed by other departments and any changes required for the greek marketplace are forwarded from the Greek Market to the Engineering Department of Beat.

### 2.1.2 Department's structure

Greek Market is constituted by five teams, Operations, Costumer Experience, the Engineer's of Beat Hotels named as GR Squad team, Marketing and Finance team. The General Manager of this Department is Vasilis Danias and the total number of people working in it is 37.

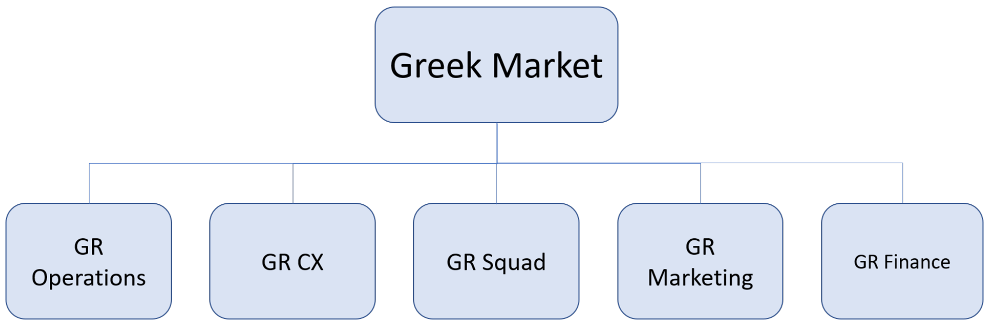


FIGURE 2.1: Greek Market's structure

### 2.1.3 Basic Procedures

Department's basic procedures are based on the management of three products in the borders of greek market, Hive, Beat App and Beat Hotels.

In more details, each team has different responsibilities. CX team is responsible for training drivers, resolving tickets and detecting any problems regarding these three products. The term tickets is any calls or visits made, or emails sent by either a passenger or driver.

Operations team is responsible for designing and controlling the process of production and redesigning business operations in terms of using as few resources as needed and meeting customer requirements. Marketing is creating, communicating, delivering, and exchanging offerings that have value for both customers and society in total. Competitions, sponsorships, banners, products or videos created for advertisement and social media management are made by this team.

Finance is responsible for beat driver's payments, while GR Squad is the team responsible for developing BeatHotels service.

### 2.1.4 GR Squad

GR Squad ia a newly conducted team which is responsible for the development of BeatHotel. Team is consisted by eight people, three front-end developers, including myself, three back-end, one Product Owner and one Scrum Master following the agile culture as the other Beat teams.

BeatHotel is a service provided only in Greece and started about seven months ago. The technologies that are used for the development of BeatHotel's driver app, dashboards for each Hotel and Agents' dashboard, are React Native, ReactJS, Node.js and for databases Firebase and Redis.

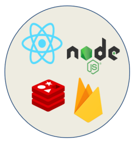


FIGURE 2.2: Technologies Used for BeatHotels

## 2.2 My Role

As a Software Developer Intern in GR Squad team, my role is releasing code that have immediate impact on BeatHotel service's users.

At the first one and a half month of my internship, I was a full stack developer and had the opportunity to work on both front and back-end elements of BeatHotel system. During this period, I was responsible to deliver npm packages, code in Node.js and ReactJS in order to complete requested projects for Agent's and Hotels' Dashboards, improve and extend tests and code coverage.

After this period of coding in Javascript for both back and front-end, getting familiar with BeatHotel service, my team and the way things flow, I had to choose between front-end and back-end developer. So, as a front-end I started to deliver tasks only referred to client-side development in order to maintain and extend existing web-site dashboards in ReactJS.

### 2.2.1 Skills Required

The skills required for this internship are enumerated below.

- Ability to produce high quality, maintainable and reusable Javascript code in React and Node.js
- Ability to build a three-layer web application
- Good knowledge of Unix based Systems
- Basic understanding of both Sql and N0-Sql databases
- Ability of problem solving and understanding algorithms complexity
- Familiar with GitHub Usage
- Ability to work in a team, communicate ideas, be an active member and deliver on time

# Chapter 3

# Projects/Activities

## 3.1 Introduction

## 3.2 Project 1

## 3.3 Project 2

## 3.4 Project 3

## 3.5 Project 4

# Chapter 4

# Results

## 4.1 Project 1

### 4.1.1 Description

### 4.1.2 Best Practices

### 4.1.3 Schedule

### 4.1.4 Problems occurred

## 4.2 Project 2

### 4.2.1 Description

### 4.2.2 Best Practices

### 4.2.3 Schedule

### 4.2.4 Problems occurred

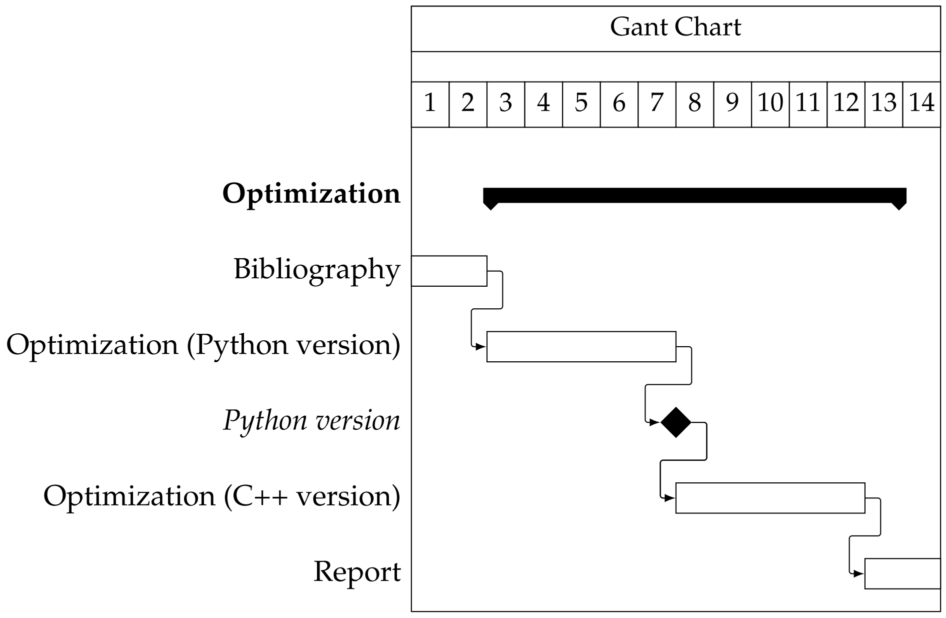
# Chapter 5

# Updated Time Management

## 5.1 Time Schedule

| Activity | Duration |
|---|---|

## 5.2 Gantt Chart



# Chapter 6

# Skills

| Skills | Related Methodologies | Examples | Summary |
|---|---|---|---|
| Monday | 11C | 22C | A clear day with lots of sunshine. However, the strong breeze will bring down the temperatures. |
| Tuesday | 9C | 19C | Cloudy with rain, across many northern regions. Clear spells across most of Scotland and Northern Ireland, but rain reaching the far northwest. |
| Wednesday | 10C | 21C | Rain will still linger for the morning. Conditions will improve by early afternoon and continue throughout the evening. |

# Part II

# Thesis

# Chapter 1

# Introduction

## 1.1 Definition

## 1.2 Research Goal

## 1.3 Thesis's Structure

# Chapter 2

# Background

Web applications have made a rapid progress over the last years in the field of both technologies used and architectural approach. A web application is consisted of a client and a web server. Client is considered to be, for example, a web browser in a mobile phone or in a desktop computer. Client sends a request to the server, where processes are performed depending on the request, and a response is sent back to the client. (Voutilainen, 2017) As picture bellow reveals, client is making a request and server attempts to fulfill it by referring to a data base, performing calculations, controlling or sending another request to other servers.

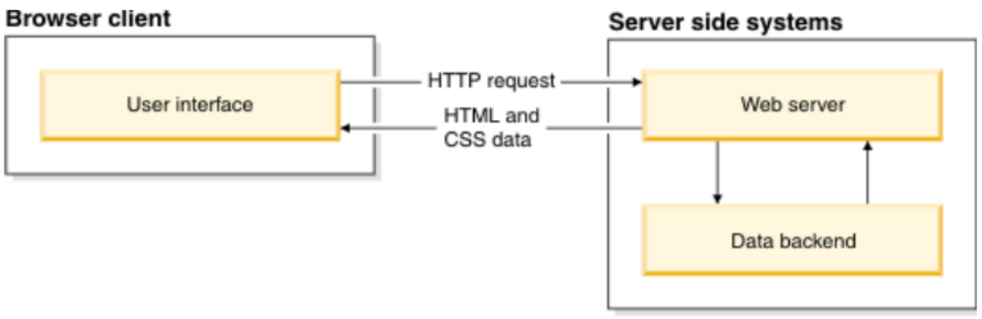


FIGURE 2.1: Web application data transfer model

Functionalities can be spotted in the client side as well. Client can customize HTML DOM, Document Object Model structure used for accessing different elements, display interactive visualizations or alerts through the utilization of Javascript programming language. (Voutilainen, 2017)

Architecture, which is a set of defined terms and rules used as instructions to build a web service, influence both software design and engineering. The choice of a web application's architectural design impacts on development time and maintenance costs, every-day's transactional performance, response times, continual application's flexibility and scalability. The architecture is selected based on the app's complexity, integration level, number of users and their geographical distribution, network's nature and long-term transactional needs. (Cemus et al., 2015)

### 2.0.1 Architectural Designs

In the late 1950s, mainframe architecture was introduced. This architecture was designed for mainframe computers that are used to large-scale computing applications, such as data storage or customer statistics. Every kind of program and data was stored in a single machine and users could only reach this centralized computer only through the terminals' usage.

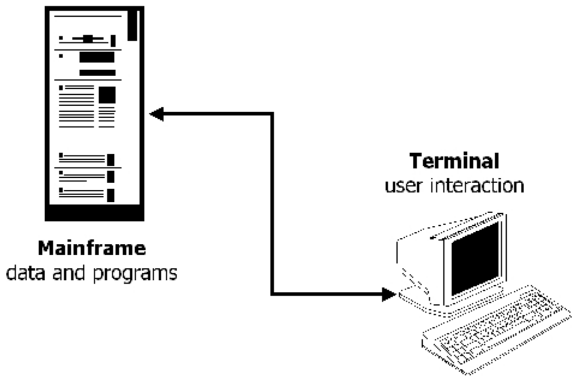


FIGURE 2.2: Mainframe Architecture

In the 1980s, two-tier architecture was introduced due to the entry of network connected computers. In more details, this architectural approach consists of an application running in the client and interacting with the server as a database management system. In that perspective, client contains the presentation logic, navigation of the application, business rules and the database access. By changing the business rules in two-tier architecture, the client had to be modified and tested all over again, even in case the user interface is the same. For minimizing the costs of conversions, presentation logic and business rules had to be separated, fact that gained the principle of three-layer architecture.

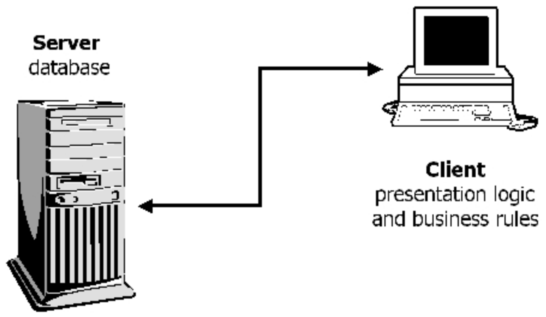


FIGURE 2.3: Two-Tier Architecture

In a three-tier architecture, also referred as multi-tier, there are up to three interacting layers. System functionality is thereby distributed with each own of these tiers having a subset of responsibilities.

As the Figure 2.4 shows, first tier refers to the presentation logic, known as client, and includes user interface and input validation. Second tier or, in other words, middle tier or application server, provides data access and the business logic. Finally, third tier is the data server and provides business data and resources. The pros

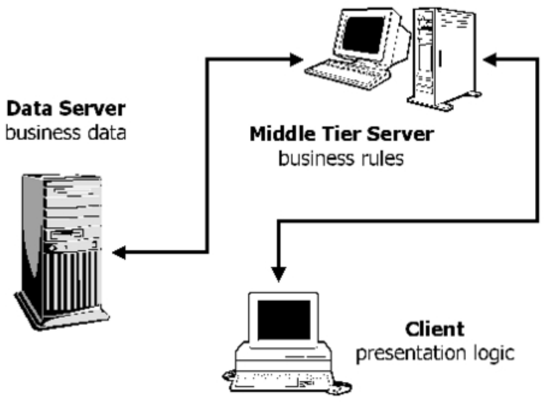


FIGURE 2.4: Three-Tier Architecture

of three-layer architectural design is the ease replace or modification of any layer without influencing or changing the others. Another advantage is the load balancing that this separation of layers and the database functionality provides. (Ramirez, 2000)

## 2.1 Three-layer Architecture

Three-layer Architecture is a software architectural pattern in which the application is separated into three logical layers known as presentation, business logic and data storage layer. This architecture is adopted by client-server applications that have frontend, backend and database.(Chen and Zhang, 2013) Data access and calculations required by the presentation layer, are part of the business logic layer and, for this purpose, calls to middle-tier servers are made. Middle-layer servers can be approached by various clients, which are from different applications as well. Each one of these tiers has specific responsibilities and can be handled independently. (Gallaugher and Rarnanathan, 1996)

Interaction between client and server is succeeded mostly through the Remote Procedure Call, which is a way to describe calling mechanism among procedures and to exchange data via messages. In RPC, clients request data by passing parameters needed and specify data structure to received values in order the request to be fulfilled.

HTTP is the protocol used for messages' passing. HTTP stands for Hypertext Transfer Protocol through which resources are exposed across distributed systems. An HTTP request includes a body with the representation of resources required by the client. An architectural style that is based on RPC implementation is called Representational State Transfer. REST is implemented in client/server models where the client is gaining data or interacting with resources managed by server.

There are a lot of advantages adapting three-architecture layering design. First and foremost, modularity, having separate software entities allows each layer to be managed independently of the others.(Chen and Zhang, 2013) This has as a result different groups of people to focus on different tiers, so as parallel development to be succeeded and people to become specialists. It has to be mentioned that skills needed for application development varies, and having some experts responsible for each tier can improve the general quality of the application.(Gallaugher and Rarnanathan, 1996)

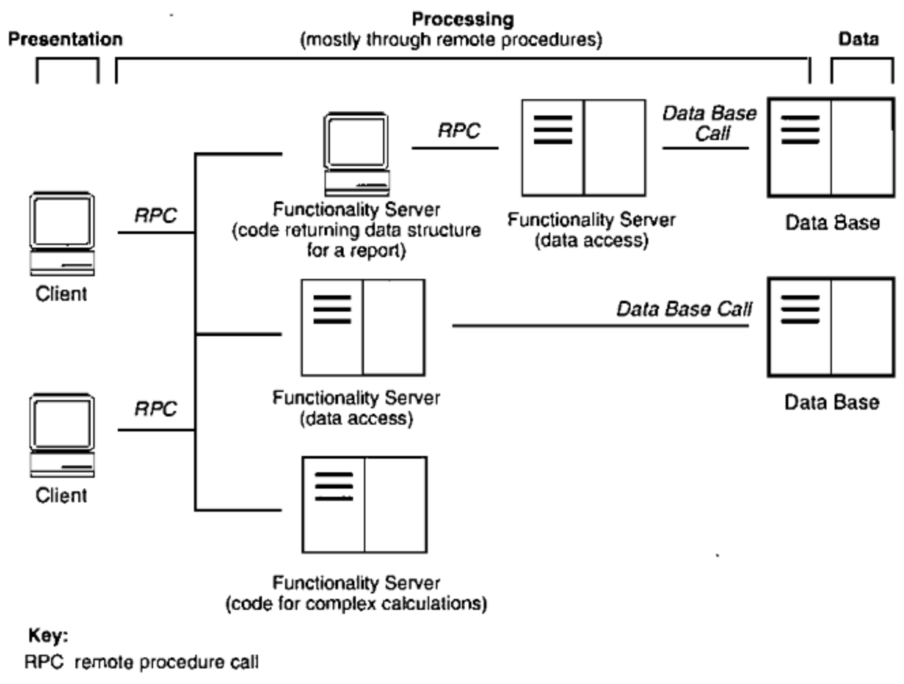

Processing
(mostly through remote procedures)
Presentation
Data
RPC
Data Base Call
RPC
Functionality Server (code returning data structure for a report)
Functionality Server (data access)
Data Base
Client
Data Base Call
RPC
Functionality Server (data access)
Data Base
Client
Functionality Server (code for complex calculations)
Key:
RPC remote procedure call

FIGURE 2.5: Three-Tier Architecture

Scalability of three-tier architectural pattern is another benefit. This architecture provides flexibility in resource distribution. In more details, servers are portable, and dynamically distributed and changed based on the application's requirements. (Gallaugher and Rarnanathan, 1996) Each of the tiers scales horizontally in order trafficking and request demand to be supported. For example, scalability can be done by adding more Elastic Compute Cloud instances, which is a cloud service providing security and compute capacity, and load balancing to each layer. (Chen and Zhang, 2013)

Another core profit of this distribution is that code units can be reusable. This logic minimize the maintenance costs, development efforts and the ability to easily switch technologies used. Moreover, there are various additional features that support modularized applications, which means that integrated security, server control and dynamic fault-tolerance are supported. (Gallaugher and Rarnanathan, 1996)

Security is easier to be performed in three-layer architecture. All of the interactions within the application are made through private Internet protocol addresses. Users access client/server systems via the presentation tier. The other two layers, server and database, are not exposed to network which offers protection, security and barriers against malicious users. Fault tolerant is also provided for adaption in any unexpected change. (Chen and Zhang, 2013)

In conclusion, this layering model is the most frequent architectural design and is defined as components' separation into different tiers. Each layer's components are abstract with limited dependences between them, reusable and easy maintained. (Chen and Zhang, 2013)

### 2.1.1 Presentation Layer

Presentation layer or client is the user interface part of the application. In other words, this layer controls the interaction between user and system, and is also responsible for input validation. Presentation tier's infrastructure is exclusively related to interface elements. (Chen and Zhang, 2013)

### 2.1.2 Business Logic Layer

Business logic layer or server is the body of an application. This is related to how the service works and it is responsible for computations and connection between client and database. (Chen and Zhang, 2013) Several server requests and data access is made through the middle tier instead of client, thus traffic, memory and disk storage requirements are minimized. (Gallaugher and Rarnanathan, 1996)

### 2.1.3 Data Layer

Data access layer or database is responsible for providing business data and resources. Data are stored in this layer and are accessed from other layers when it is needed. (Chen and Zhang, 2013)

## 2.2 Rest

Representative State Transfer is an architectural style for building networked hypermedia applications that are easy to implement, lightweight, maintainable and scalable. A service based on Rest architectural approach is called Restful. The main purpose of a Restful service is providing data access to the client through a window. By the term of data or resources, we mean everything that can be captured in a computer-based informational system, such as pictures, videos, Web pages or any other business information. (Vaqqas, 2014)

Rest architectural style is based on four main principles. The first one is that resources are identified by URIs, which is a universal addressing space for resources and service exploration. Furthermore, the principle of uniform interface is the way resources are by using a set of four operations. More specifically, resources can be controlled by PUT, POST, GET, DELETE, which are for update, add, retrieve and remove actions. The next principle stands for self-descriptive messages. In other words, resources can be represented in a variety of formats depending on the requirements, such as JSON, XML, HTML or even plain text, and meta-data are also available in order caching to be controlled, transmitting errors to be detected and authentication to be performed. Last but not least, every interaction with a resource is stateless. This means that each request message is self-contained and there is no need of state transfer. (Pautasso, Zimmermann, and Leymann, 2008)

HTTP is the protocol that is used in almost every Restful Web service and it is implementing the service with features of a uniform interface and caching. This protocol is giving the ability of communication between client and server through messages. Client sends a request to the server, and server returns a response to that request. (Vaqqas, 2014) Rest APIs are the bridge between this communication, so as the client to get the resources needed from the server without knowing how the structure of Restful API works and vice versa. (Padmanaban et al., 2018)

In conclusion, Rest architectural approach is a lightweight infrastructure, where services can be built with minimum tooling and effort that leads to inexpensive and

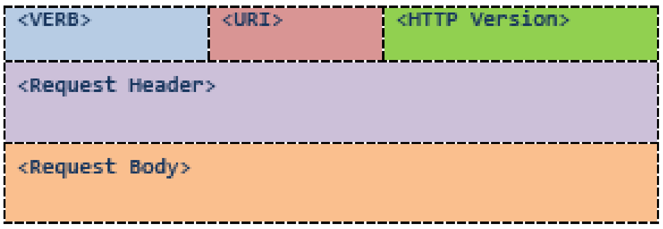


FIGURE 2.6: HTTP request format

low barrier adoption. Restful Web services serve a large number of clients because of the caching and load balancing build of Rest, and have the ability to access resources without registration, and the advantage to optimize data in different formats. (Pautasso, Zimmermann, and Leymann, 2008)

## 2.3 Dynamic Pages

Web application systems are using Web browsers for representing client. Web browsers interprets HTML, CSS and Javascript code, and communicate with server side through the usage of urls and HTTP protocol. In first place, static web pages were send to browsers and server's responsibility was to locate and send files, based on client's requests. Afterwards, dynamic pages were generated by servers via running software, and by clients via executable code. Thus, a lot different software development, such as languages, APIS, libraries or frameworks, were build to support dynamic pages' approach. Ajax, Javascript, Python, and much more other such development tools, are a good example. (Kulesza et al., 2018)

Ajax stands for "Asynchronous JavaScript And XML" and provides asynchronous requests for data transfer. By using Ajax, an application can dynamically update its page's content without the need of reloading the entire DOM all over again. In general, Ajax is a way to get a certain functionality by using web technologies on the client side, in order to create asynchronous applications. Ajax is different than JS frameworks or libraries which include a lot capabilities and functionalities along with Ajax. (Voutilainen, 2017)

JavaScript is the language created to fulfill the need of client-based dynamic pages. JavaScript is being developed really fast, and new technologies are regularly released. So, libraries and frameworks are inclined to make easier JS development and improve its capabilities. JavaScript's libraries exist for a long period of time and in 2006, they got popular for first time with jQuary. AngularJS and Ember, which are other JS frameworks, became known between 2010 and 2011. AngularJS was the first framework that composed routing and data binding in one. Ember followed and provided some improvements on AngularJS, such as better use of routing. Other frameworks and libraries, such as React, Angular, Vue, were announced until 2015. Javascript and its frameworks, CSS and native HTML are more and more powerful today. (Voutilainen, 2017)

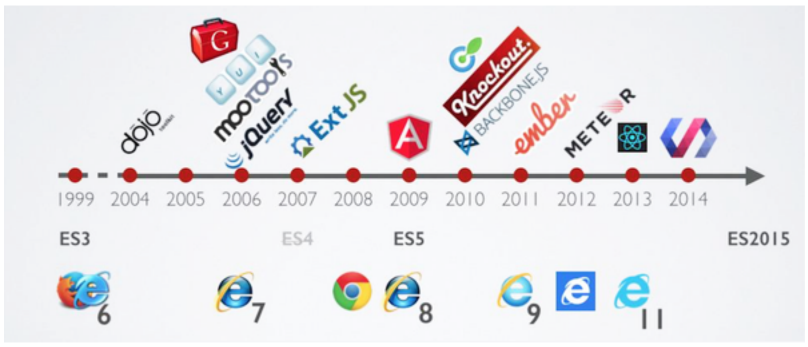


FIGURE 2.7: Historical overview of JS frameworks and libraries

An application framework is defined as a large reusable collection of libraries, functions and tools in order application's main structure to be implemented. Thus, JavaScript frameworks are implementing the whole structure of an application, and make development of JS painless and faster by offering ready, optimized functions in comparison with native JS. Dependency management, file system structure and routing possibilities are also included. The term framework is used to declare that the execution flow of an application is handled by the framework and not the developer. This is the main difference between JS frameworks and libraries, since a library provides only a set of functionalities, while a framework conducts processing and data flow. (Voutilainen, 2017)

Nowadays, widely scoped, full and continually improved optimized JS frameworks and libraries, i.e Angular and ReactJS which will be analyzed in detail later on, their popularity have been significantly increased over time in front-end development. However, there is no framework considered to be as the best, a group of great choices exist that do have different functionalities and are selected based on some parameters that will be investigated in this thesis. (Voutilainen, 2017)

# Chapter 3

# ReactJS

## 3.1 Definition

## 3.2 History

## 3.3 Notable Features

### 3.3.1 Stateful Components

### 3.3.2 JSX

### 3.3.3 Architecture Beyond HTML

## 3.4 Case Study

# Chapter 4

# Angular

## 4.1 Definition

## 4.2 History

## 4.3 Typescript

## 4.4 MVC Architecture

### 4.4.1 Model

### 4.4.2 View

### 4.4.3 Controller

## 4.5 Case Study

# Chapter 5

# Comparison

## 5.1 In which case to use ReactJS

## 5.2 In which case to use Angular

# Chapter 6

# Conclusion

## 6.1 Future Work

# References

Cemus, Karel et al. (Oct. 2015). “Enterprise Information Systems : Comparison of Aspect-driven and MVC-like Approaches”. In: *Proceedings of the 2015 Conference on research in adaptive and convergent systems (RACS). ACM*, pp. 330–336. URL: https://dl.acm.org/citation.cfm?doid=2811411.2811477.

Chen, Xuebin and Shufen Zhang (Apr. 2013). “Experiment teaching management system based on three-layer architecture”. In: *8th International Conference on Computer Science and Education Computer Science and Education (ICCSE)*, pp. 1298–1302. URL: http://eds.a.ebscohost.com/eds/detail/detail?vid=0&sid=1e19995b-4b66-4809-a426-e0a4ac683f32%40sessionmgr4008&bdata=JnNpdGU9ZWRzLWxpdmU%3d#AN=edseee.6554122&db=edseee.

Gallaugher, John M. and Suresh C. Rarnanathan (1996). “Choosing a client/server architecture: A comparison of Two-and Three-Tier systems”. In: *Information Systems Management* 13, pp. 7–13. URL: http://eds.a.ebscohost.com/eds/detail/detail?vid=1&sid=64174bb1-2f01-4844-b4a0-8eb1916d5792%40sessionmgr4008&bdata=JnNpdGU9ZWRzLWxpdmU%3d#AN=edselc.2-52.0-0642312740&db=edselc.

Kulesza, Raoni et al. (Apr. 2018). “Evolution of software architectures: From Web 1.0 to Web 3.0 systems”. In: *WebMedia 2018 - Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, pp. 11–13.

Padmanaban, R. et al. (2018). “Computability evaluation of RESTful API using Primitive Recursive Function”. In: *Journal of King Saud University - Computer and Information Sciences*. URL: https://doi.org/10.1016/j.jksuci.2018.11.014.

Pautasso, Cesare, Olaf Zimmermann, and Frank Leymann (2008). “RESTful Web Services vs . “ Big ” Web Services : Making the Right Architectural Decision Categories and Subject Descriptors”. In: *Proceedings of the 17th international conference on World Wide Web*, pp. 805–814. URL: http://wwwconference.org/www2008/papers/pdf/p805-pautassoA.pdf.

Ramirez, Ariel Ortiz (July 2000). “Three-Tier Architecture”. In: *Linux Journal*. URL: https://www.linuxjournal.com/article/3508.

Vaqqas, M. (Sept. 2014). “RESTful Web Services: A Tutorial”. In: *Dr. Dobb’s Journal*. URL: http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069.

Voutilainen, Jaakko (Dec. 2017). “Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development”. In: *Metropolia University of Applied Sciences*, pp. 1–50. URL: https://www.theseus.fi/bitstream/handle/10024/138668/Voutilainen_Jaakko.pdf?sequence=1&isAllowed=y.