LAMK
Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

HOW TO INTEGRATE REACT NATIVE WITH NOSQL DATABASE (HOSTED ON AZURE)

Case: Napapiiri Jukola 2020 Teaser Mobile Application

LAHTI UNIVERSITY OF APPLIED SCIENCES
Bachelor of Business Administration
Degree program in Business Information Technology
Autumn 2019
Mahyar Mohammadi

Abstract

Author(s) Mohammadi, Mahyar	Type of publication Bachelor’s thesis	Published Autumn 2019
	Number of pages (61)	

Title of publication

How to Integrate React Native with NoSQL Database (Hosted on Azure)

Case: Napapiiri Jukola 2020 Teaser Mobile Application

Bachelor of Business Administration
The degree program in Business Information Technology

Abstract

The primary goal of the study is to familiarize the readers with the developing procedure of a hybrid mobile application. In a more in-depth perspective, the author tries to meet the secondary goal of the study by explaining the reason for the selected technology in case mobile application, Jukola 2020, including React Native platform, NoSQL, and Azure database by elaborating the development process.

The theoretical part of the thesis reviews the previous studies by applying both primary and secondary resources. Chapter 2 firstly focuses on React Native overview and its advantages and disadvantages. Then NoSQL database including its types and features is discussed, followed by Mango as well as Azure databases. The third part of chapter 2 concentrates on the concept of cloud computing, and it is wrapped up by the introduction of Jukola event and application.

The empirical part of the thesis follows the constructive research methodology by analyzing the development process of the Jukola 2020 teaser mobile application and the applied technologies in chapter 3. The author also discusses the developing process of the case application, step by step.

The study successfully answers the research questions by spelling out the logic behind implemented technologies, including React Native, NoSQL database, and Azure for the Jukola 2020 teaser mobile application.

Keywords

Hybrid mobile application, React Native, Mango database, NoSQL, Azure, Cloud computing

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACID	Atomicity, Consistency, Isolation, and Durability
API	Application Program Interface
APP	Application
AWS	Amazon Web Services
CDN	Content Delivery Network
CD	Continuous Deployment
CI	Continuous Integration
DB	Database
FedRAMP	Federal Risk and Authorization Management Program
GPS	Global Positioning System
IaaS	Infrastructure as a Service
iOS	iPhone Operating System
IoT	Internet of thing
JSX	JavaScript XML
MVP	Minimum Viable Product
NoSQL	Not Only SQL
NPM	Node Package Manager
OY	Company
PaaS	Platform as a Service
REST	Representational State Transfer
RDBMS	Relational Databases
RN	React Native
UI	User Interface
XML	Extensible Markup Language

1 INTRODUCTION

1.1 Purpose and Research Methodology

This constructive study aims to elaborate the integration of the React Native platform with the NoSQL (Mongo) database and Azure cloud computing service through a problem-solving approach. In a more in-depth outlook, the thesis author tries to depict how to develop a mobile application using the technologies mentioned above regarding the case study “Napapiiri Jukola 2020 Teaser mobile application”.

The motivation behind the study is the author, as the Jukola 2020 mobile application developing team member of Tietotalo Infocenter Oy, found it inspiring how to integrate technologies such as React Native, Mongo database, and Azure to develop a hybrid mobile application just in three weeks.

As the main aim of this empirical and normative study is providing a real-world and noble problem-solving approach in mobile application development, by answering to the questions of how and what technology to implement through the construction of models, diagrams, plans, etc., this thesis follows the constructive research method as the subbranch of qualitative research methodology through problem-solving approach. (Kasanen; Lukka and Siitonen 1993.) In the theoretical part of the study, both primary and secondary resources are used via desk-research. In the empirical section, the implantation process of a real-life project has been explained by the author with taking into account the confidentiality of project implementation by the consulting company “Tietotalo Infocenter Oy.”

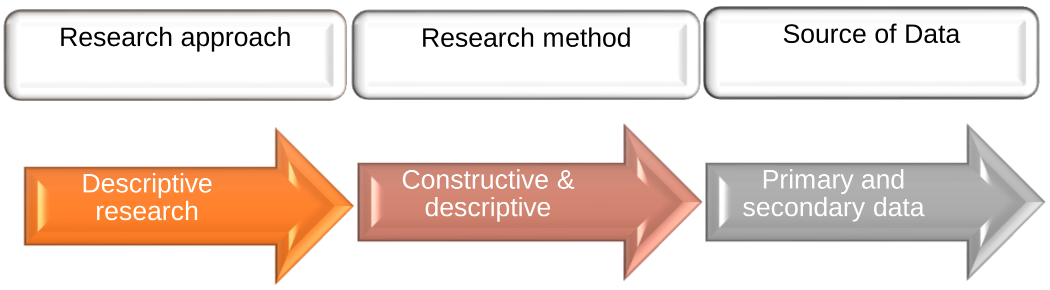


Figure 1 Thesis Methodology

1.2 Thesis Objectives and Research Questions

While deciding to develop an application, the companies face many technical, business, design, marketing, and branding decisions. Each of them has a certain weight and impact on the success of the application. (Blair 2017.) The following research questions are designed to help developers decide better what technologies to select and how to implement those on similar projects:

1. How to develop a hybrid mobile application by React Native, NoSQL database on Azure?

By considering the first research question, the author is going to meet the thesis objectives by the following questions:

2. Why selecting the React Native platform for Jukola Teaser Mobile Application development?
3. Why is the NoSQL database used for storing data?
4. Why is Azure applied for hosting the web application and database?

1.3 Thesis Structure

This thesis consists of five chapters. Figure 2 presents the process of writing this study.

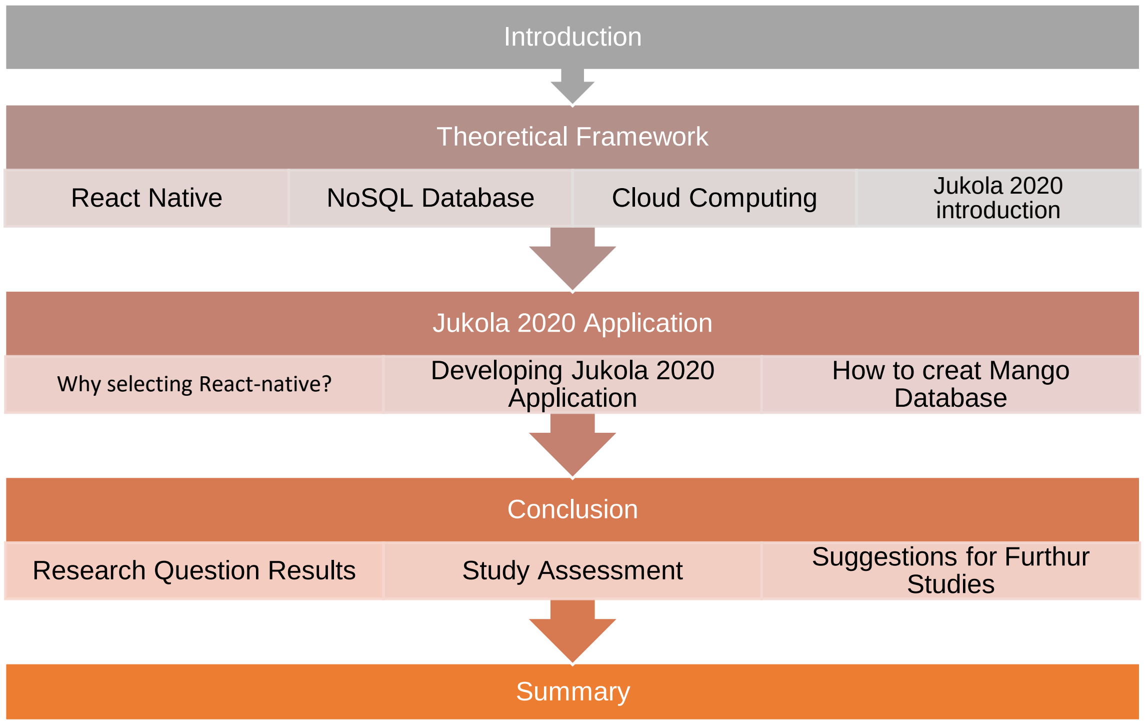


Figure 2 Thesis Structure

The first chapter provides the mind map of the whole research by considering the case mobile application, and it's development process, where the author explains the purpose and motivation behind, followed by research questions and scope of the study. Chapter two focuses on three pillars of the empirical part of the study. The first sub-section concentrates on the introduction of the React Native platform, including its pros and cons; The second sub-section elaborates the NoSQL database, and in a more in-depth study, the author defines NoSQL types and merits and wraps up this section by explaining why to use the Mongo database. In the last part of chapter two, the author investigates cloud computing service by highlighting the power points of the Azure cloud; then, he compares Azure with Amazon Web Services (AWS) as Its most prominent competitor.

Chapter 3 tells the story of the Jukola 2020 project, developed by Tietotalo Infocenter Oy, by describing the reason for selecting React Native as the mobile platform in part one. At the next step, the author reveals how to create a Mongo database and finishes the third part of chapter three by showing how to host the Mongo database on Azure.

Chapter four contains the study results in a nutshell, and the fifth chapter includes the thesis recap and the suggestions for further researches.

Mobile usage grows by 6% every year. The ever-increasing growth of mobile usage has led even small businesses to look for mobile app development services. A common question that developers have in their mind is: Should they go for the native or hybrid or web apps? (Wadhwa 2019.)

2 BACKGROUND OF STUDY

Currently, the three main mobile application types are Web app, Native app, and Hybrid apps. Mobile web applications are used for mobile devices, which require only a web browser to be installed on the device. (Gartner IT Glossary 2019.) Native mobile app refers to a smartphone application that is coded in a specific programming language, such as Objective C for the iPhone Operating System (iOS) or Java for the Android operating system (Techopedia 2019). The hybrid application includes elements of both Native and Web applications. Hybrid application features include:

- The ability to operate whether or not the device is connected
- The ability to integrate with the devices' file system
- The ability to integrate with the web-based services
- An embedded browser to improve access to dynamic online content (Rouse 2011.)

There are several criteria to consider before deciding about the right app for the businesses illustrated in Figure 3 :

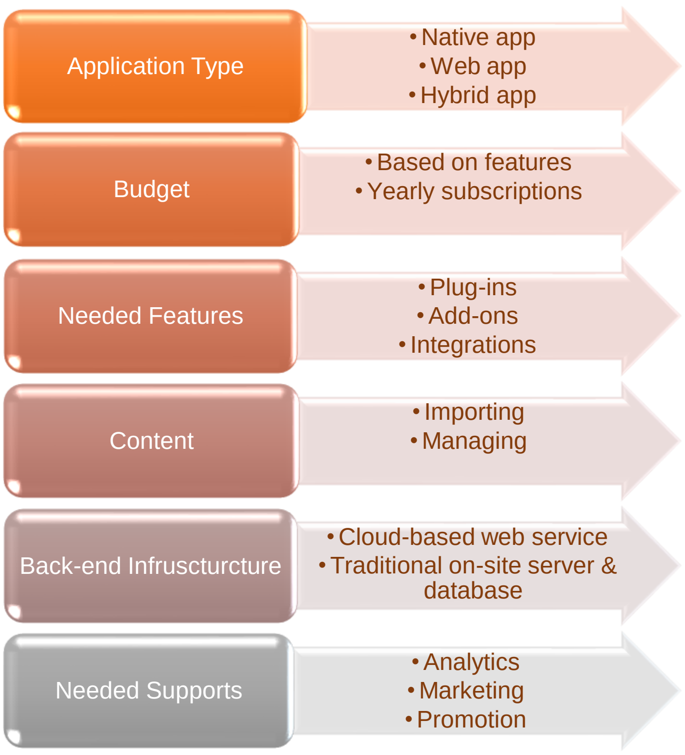


Figure 3 App Development Fundamental Criteria (adapted from Wodehouse 2019)

The other point to consider is that the cloud-based applications benefit elastic resources that can grow and shrink according to the users' needs; thus, the application performance does not deteriorate by migrating to the cloud. The other value that adds up deploying applications to the cloud is its equipment, software, maintenance, and administration cost-efficiency. (Ren 2012.)

2.1 React Native

The focus of this part is on describing React Native platform features. The author firstly points out the React Native background, and then in a further investigation, spells out its advantages and disadvantages.

2.1.1 React Native Overview

React Native is a JavaScript framework. It is used for writing and rendering mobile applications on both iOS and Android platforms. It is based on React, Facebook's JavaScript library for building user interfaces. One of the prominent features of React is providing a platform for web developers to write mobile applications that look like native ones. React Native makes it easy to develop both Android and iOS simultaneously since most of the codes can be shared between the two platforms. (Eisenman 2019.)

React Native projects are written using a mixture of JavaScript and Extensible Markup Language (XML)-sequel Markup, known as JavaScript XML (JSX). The React Native bridge invokes the native rendering Application Program Interface (APIs) in Objective-C for iOS or Java for Android. Since the application will render using real mobile User Interface (UI) components, it looks like other mobile applications. React native apps can access platform features like camera Global Positioning System (GPS). Most of the codes are cross-platform. (Eisenman 2019.)

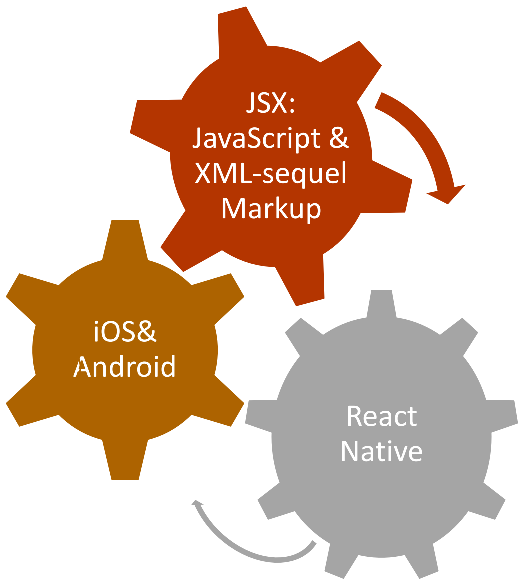


Figure 4 React Native Overview (adapted from Eisenman 2019)

2.1.2 React Native Merits

Malik (2018), Rajput (2018), Chrzanowska (2019) and Gawron (2019) count the reasons behind React Native popularity among developers:

Time Efficiency

Czermiewski, in his blog, mentions considering the React Native's cross-platform technology advantages, by which the source code can be rendered into both native Android and iOS components, RN is more attractive as cost and time-efficient platform (Czerniewski 2019). React Native has a great feature called "Hot Reload," by which the results of developing can be checked immediately (Malik 2018). React Native accelerates development while developers only need to build applications once using JavaScript as a purely native approach to develop for both iOS and Android apps (Dyvliash 2018).

Hot Reloading

The developer can keep the app running while developing new versions by this feature, and there is no need to rebuild the app (Chrzanowska 2019). React Native reflects the effect of developers works in real-time, without the need to compile code. This is done by Hot Reloading, which allows the changes in code to be seen right away, without building the app. (Czerniewski 2019.)

Performance

User Experience and User Interface, as the principal features of produced apps, are the real values that React Native provides by Native looking components. Accordingly, the applications developed by RN look identical to those developed natively for either Android or iOS. (Czerniewski 2019.) React Native uses the native components, modules, and APIs, and it improves performance (Gawron 2019).

Building Faster

Since there are many ready-to-apply components available in this framework, the development time is much shorter. For example, developing the same application with React Native would be about 30% faster than Swift for iOS. (Chrzanowska 2019.) In 80% of real business cases, RN can increase mobile development speed 30-40% (Polidea 2019).

Growing Community

React Native is an open-source platform, and every developer can contribute to the framework and easily ask help from the community members (Rajput 2018). The open-source nature of React Native attracts contribution. In July 2019, the React Native repository was the 14th most starred repository on GitHub. Facebook, of course, has a determinant role in the React Native community. (Ratner 2019.)

Ready-made Components

Thanks to React Native “ready to use” components, the development on this platform is more straightforward and faster (Malik 2018). React Native also facilitates integration with web development by cross-platform compatibility. By applying React Native, developers can use git submodules to share various types of codes across platforms such as Hooks and Helpers for components and assets. It should be considered that when using React Native, developers need to set up CI&CD and analytics twice, once for Android and once for iOS. Some tools like Nevercode, Fabric, and Firebase analytics can be applied to this aim. (Pruulmann 2019.)

Cost Efficiency

Since it is possible to use the same code on both Android and iOS platforms, React Native helps cutting development costs (Rajput 2018). Furthermore, the codes already written for a website made with React can be reused in React Native. This will save almost 30% of developing costs. (Dyvliash 2018.) RN reduces the app development costs by 25-30%, as one development team is needed, instead of two Native developer teams. Thus, project management would be easier and more efficient, as two teams of iOS and Android will never work at the same pace, but applying RN makes the app production process unified by the same development rate. (Czerniewski 2019.)

Smaller Team Size

Having a single team vs. two different teams of Android and iOS developers, helps project management, communication, time, cost and developing with a faster speed while reacting to project changes more efficiently (Malik 2018). By using React Native, one developing team would be enough instead of two separate ones for developing Android and iOS applications (Chrzanowska 2019).

Reliable and Stable Applications

Being utilized by large companies like Facebook and Skype endorses React Native's quality and stability (Rajput 2018). React Native was developed by Facebook in 2015. By 2018, world-leading companies like Instagram, Tesla, Skype, Wix, Uber, Walmart, Airbnb, etc. utilized React Native advantages. (Dyvliash 2018.)

Less Native Codes

About 90% of the codebase can be applied to cross platforms. This can meditate the complexity of platform-specific features; for instance, many same codes are applied for adding new features and bug fixing. (Dyvliash 2018.) Compared to other hybrid tools, the React Native platform needs less native code (Malik 2018). To recap the mentioned above points, Figure 5 shows the advantages of React Native.

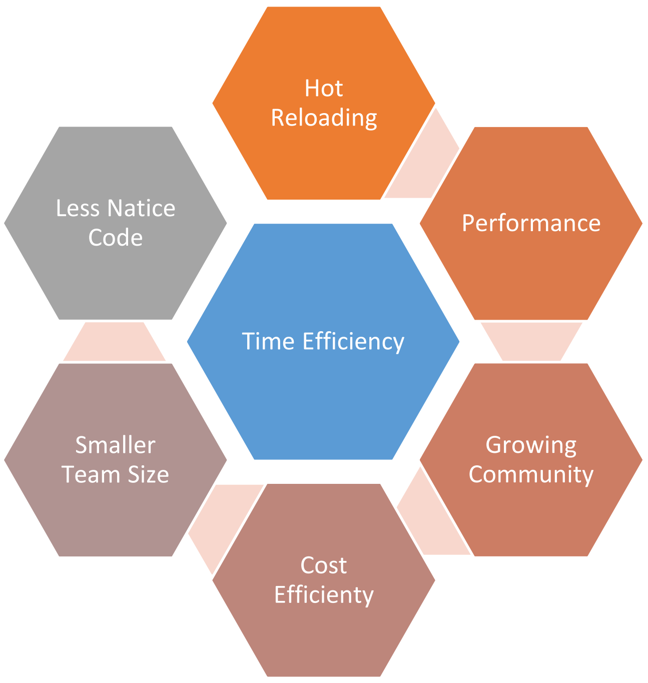


Figure 5 React Native Merits

2.1.3 React Native Demerits

Even though React Native offers outstanding features, it also has some shortcomings:

A Small Collection of Ready-made Components

Owning a new framework, React Native contains a small collection of ready-made components. React Native Still has code and UI limitations. It needs native coding for the use of hardware such as Bluetooth, accelerometers, gyroscopes, etc. In fact, in some cases, React Native can add complexity to the project. (Chrzanowska 2019.)

Native Code Requirement

Although the platform provides a variety of prefabricated features, React Native still needs native codes for some specific features. (Gawron 2019). While applying RN, some third-party components might not work as expected, to solve the issue, Java/Objective-C/Swift expertise is needed (Churylov 2019).

Performance Deficiency

Compared to native apps, React Native performance at some points might be drained (Malik 2018). In case of hardware-intensive apps by React Native, performance in older models of phones like Samsung S8 or iPhone 7 might quickly drop. This is more sensible in running transitions and animations smoothly with React Native. Optimizing these features will take a longer time compared to Native platforms regarding React Native longer learning curve. React Native is recommended for lighter projects that do not need mostly display information about products. (Pruulmann 2019.)

Component Quality

Since official developers do not write the ready-made components of React Native, there might be some bugs while applying them (Malik 2018). Cross-platform development of high-performance requirements with RN is slower than Native (Czerniewski 2019).

Long Learning Curve

As RN is new and not 100% tested, the ongoing changes and improvements, as well as the probable errors on the platform, requires time to be expert while applying the platform for any real-life with different levels of complexity (Czerniewski 2019). For the developers who are new to React, the logic of React Native might be challenging to learn (Malik 2018).

Low Security

JavaScript is famous for fragility. For highly secure apps, developers applying RN need to be careful about malicious code snippets that can distort the app's core functionality. (Malik 2018.) Considering the point that React Native is library-based, it is not suitable for secure apps like banking ones (Rajput 2018).

Memory Management

React Native does not contain the memory management tools as it is based on JavaScript (Rajput 2018). To develop high computation-intensive apps, RN does not handle memory management, performance and speed efficiently (Malik 2018).

Cross-Platform Debugging

Compared to native platforms, debugging takes longer with React Native as it needs extensive knowledge of Android, iOS, and web. Moreover, React Native utilizes JavaScript, which can make debugging even more complicated. (Pruulmann 2019.) Figure 6 summarizes React Native Disadvantages.

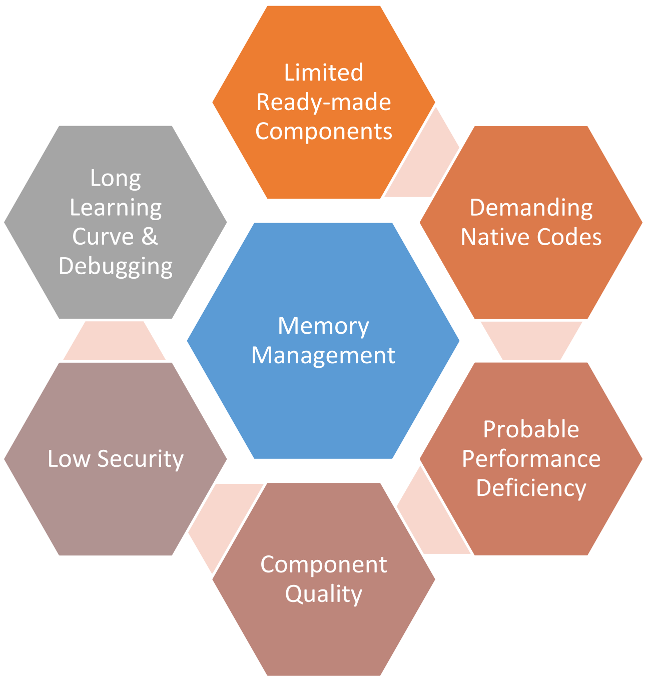

Limited Ready-made Components
Long Learning Curve & Debugging
Demanding Native Codes
Memory Management
Low Security
Probable Performance Deficiency
Component Quality

Figure 6 React Native Cons

2.2 NoSQL Database

The scalability of NoSQL with the power of SQL turned it one of the most popular databases modern web, mobile, and IoT applications. NoSQL database equips developers with easier deployment and data management features. (Couchbase 2019.) NoSQL database is a wide variety of different database technologies developed for modern applications (NOSQL 2019).

2.2.1 NoSQL Database Overview

What makes NoSQL databases different from common relational databases is that NoSQL is a form of unstructured storage. In other words, NoSQL databases do not have a solid structure similar to relational databases. (Altarad 2019.) Developers need to develop applications that can deal with a massive volume of new and rapidly changing data; for instance, the applications with finite services and limited users, now serve globally with millions of users. Bearing in mind that the waterfall app development process changed to agile development, the organizations are now turning to scale-out architecture instead of huge monolithic server and storage infrastructure since the relational databases do not support scale and agility. (MongoDB 2019.)

2.2.2 NoSQL Database Types

There are four types of NoSQL database:

1. **NoSQL Document Database**

NoSQL document database provides each key with a complex data structure. It contains a variety of key-value pairs. The value can be arrays or even nested documents. (MongoDB 2019.)

2. **NoSQL Graph Stores**

The role of NoSQL graph stores is accumulating information about networks of data such as social connections (MongoDB 2019).

3. **NoSQL Key-value Stores**

NoSQL databases are based on key-value pairs (Pandorafms 2019). It contains different items, and every element has an attribute key, including its value. The key-value store is one of the most straightforward NoSQL databases. (MongoDB 2019.)

4. **Wide-column Stores**

Wide-column stores are optimized to store queries or columns of data (Figure 7) together (MongoDB 2019).

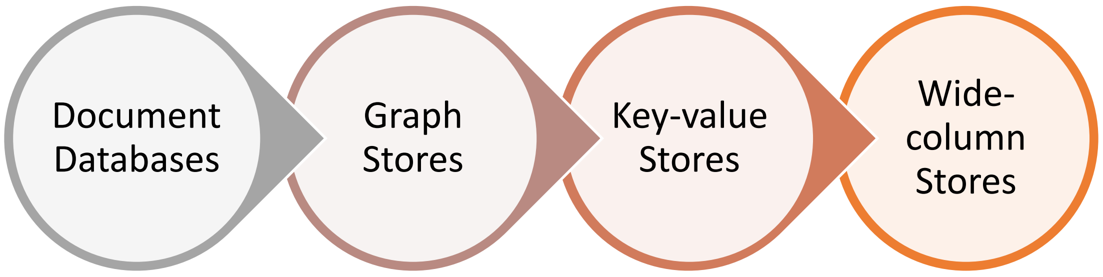


Figure 7 NoSQL Database Types (adapted from MongoDB 2019)

NoSQL databases provide more agility, easier scalability, higher performance, more availability, and lower costs. In NoSQL databases, working with the JSON data model is smoother due to ease of scaling without any extra costs. (Couchbase 2019.)

2.2.3 The Merits of NoSQL Databases

What makes a distinction between NoSQL and traditional databases are the prominent features it provides. NoSQL is non-rational, distributed, open-source, and horizontally scalable. (NOSQL 2019.)

Performance and Scalability

The horizontally scaled and distributed loads on all nodes make scalation with NoSQL databases, either open source or proprietary, easier, and cheaper than relational databases (Pandorafms 2019). NoSQL databases serve more scalable and superior performance compared to relational databases. The main goals behind NoSQL database design were providing a large volume of rapidly changing structured, semi-structured, and unstructured data aligned with agile sprints, quick schema iteration, and recurring code push as well as using object-oriented programming and employing the geographically distributed scale-out architecture. (MongoDB 2019.)

Flexible Data Models

Minor changes to the data model of SQL databases should be managed carefully. NoSQL databases have fewer data model restrictions. NoSQL Key-Value Stores and Document databases can store any data structure. (Harrison 2010.) NoSQL Databases can include a variety of formats such as column-store, key-value store, graph store, object store, XML store, and other store modes (Pandorafms 2019).

Low Cost

NoSQL databases apply clusters of cheap commodity servers to manage the exploding data and transactions volumes, compared to RDBMS that relies on expensive proprietary servers and storage platforms, as a result, the cost per gigabyte or transaction /second for NoSQL can be far less than the cost for RDBMS. Therefore, NoSQL DBs provide storing and processing more data at much lower price. (Harrison 2010.) Open-source NoSQL database can run on low resource hardware and render the deployment while it does not contain expensive licensing fees (Pandorafms 2019). NoSQL database is an excellent solution for small organizations since it is an open-source database, developer-friendly, and cost-effective to install (Jamsheer 2019).

Store Massive Amount of Data

The transaction rates of data are rising, and the volume of the stored data is increasing massively. NoSQL models can easily handle a massive quantity of data. (Regoli 2017.)

2.2.4 The Limitations of NoSQL Databases

Although NoSQL databases are the leader of new data storage technology, they are not perfect.

Security

NoSQL database is vulnerable to a long list of security problems. NoSQL database continues to develop to solve most of the issues. Still, security is a limiting factor for NoSQL deployment. (Tozzi 2016.)

Not Mature

RDBMS systems are stable and highly functional, while most NoSQL DBs are on pre-production versions with key features that still need to be implemented (Harrison 2010). Compared to SQL databases, NoSQL models are much up to date. SQL databases have grown to be more functional and stable systems over the years. (Regoli 2017.)

Data Consistency

NoSQL databases can handle the volumes of big data, while the transaction rates and limitations of data volume that can practically be managed by a single relational database is not satisfying (Bigdata 2014). ACID transactions, a technique for guaranteeing the data remains consistent, are not supported by most NoSQL databases. NoSQL depends on the ‘eventual consistency’ which presents some performance benefits, but there is the risk of losing the async among one database node with data on the other nodes. (Tozzi 2016.)

Less Support

RDBMS vendors provide a high level of enterprise support, yet since most NoSQL databases are open source projects, a few firms offer support for NoSQL databases. Furthermore, business mine information such as business intelligence (BI) is a key IT issue for all medium to large companies. NoSQL DBs provide few facilities for ad-hoc queries and analytics. (Harrison 2010.) All SQL database vendors provide 24 hours support, which is not guaranteed by NoSQL database vendors yet (Regoli 2017).

Lack of Standardization

NoSQL needs a lot of effort to install and maintain (Harrison 2010). The other problem with the NoSQL database is incompatible with SQL queries, as manual or proprietary querying is needed which makes it complex and takes more time. The query and design languages of NoSQL databases change broadly among different NoSQL products. The learning curve for NoSQL databases is more complicated than SQL databases. (Tozzi 2016.)

Less Community Support

The NoSQL database community support is less than the SQL database since it is relatively new (Jamsheer 2019).

Scalability

There are several scalability challenges. Since some NoSQL databases cannot fragment automatically, they cannot scale up or down automatically. (Pandorafms 2019.) The NoSQL databases are not entirely scalable in all situations (Tozzi 2016). Even though NoSQL databases are highly popular in high performance, high scalability, and ease of access, they still lack consistency and reliability (Altarad 2019).

Reliability

Reliability features such as atomicity, consistency, isolation, and durability are not supported by most NoSQL databases that are natively supported by relational databases. Thus, the developers must implement their codes that make the system more complex, and accordingly, the applications are highly secure and reliable, like banking systems or personal data management ones, cannot rely on NoSQL databases. Table reviews NoSQL and RDBs features. (Pandorafms 2019.)

Table 1 NoSQL vs. Relational Databases (adapted from Pandorafms 2019)

Feature	NoSQL Data Bases	Relational Databases
Performance	High	Low
Reliability	Poor	Good
Availability	Good	Good
Consistency	Poor	Good
Data Storage	Optimized for huge data	Medium size to large
Scalability	High	High (but more expensive)

2.2.5 MongoDB

MongoDB is a NoSQL database that is both scalable and flexible. Data is stored on MongoDB in JSON-like documents that refer to the objects in the applications code. MongoDB offers substantial approaches to access and analyze the data; besides, it is appropriate for horizontal scaling. Above all, MongoDB is free to use. (MongoDB 2019.)

2.2.6 Azure Cosmos DB

In the current high technology age, applications need to be responsive and consistently online. The datacenters should be close to users for reaching low latency and high availability. The applications for distributing globally, need a globally distributed database which replicates the data everywhere. Azure Cosmos DB is a globally distributed database service that offers low latency, flexible scalability, the straightforward definition for data consistency, and high availability. It provides fast response time everywhere, being always online, unlimited and flexible scalability. The databases can be configured to distribute globally and available in all of the Azure regions. Distributing the data near to users reduces the latency. The regions that are linked with the account can be added or removed any time without pausing or redeploying the application. (Microsoft Azure 2019.)

Cosmos Database is a NoSQL database platform that is offered by Microsoft and running in Azure. Four API models are supported, such as Key-Value pairs and documents. The data can be distributed globally by Azure regions easily. Cosmos DB is scalable to handle millions of reads and writes transactions per second. Cosmos DB indexes all the data automatically, Figure 8. (Penchal Reddy 2019.)

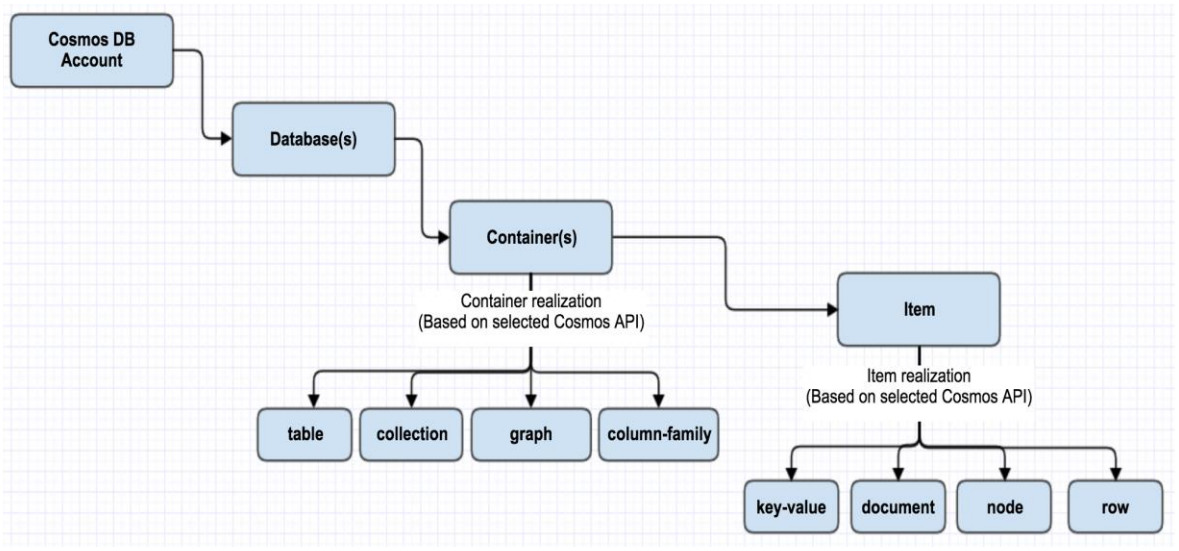


Figure 8 Azure Cosmos DB Relations (Penchal Reddy 2019)

It is a multi-model database that stores data in Key-value Pair, Document-based, Graph-based, Column Family-based databases. Azure Cosmos DB supports multi-language such as Java, .NET, Python, Node.js, and JavaScript. Also, it has multi-API support. For example, SQL API, Cosmos DB Table API, MongoDB API, Graph API, Cassandra API, and Gremlin API. It has multi-Consistency level support such as Eventual, Prefix, Session, Bounded, and Strong. Cosmos DB indexes all the data on all fields automatically. It provides 99.999% availability for both reads and writes for multi-region accounts and 99.99% for single-region accounts. Azure Cosmos DB guarantees ten milliseconds latency for reads and writes for all consistency levels. The container is a unit of scalability. It is replicated across multiple regions. Any application that needs to work with a vast amount of data at a global scale with close to real response time will benefit from Cosmos DB, such as web, mobile, gaming, and Internet of thing (IoT) applications. (Microsoft Azure 2019.) Figure 9 illustrates the Azure Cosmos DB Global Distribution

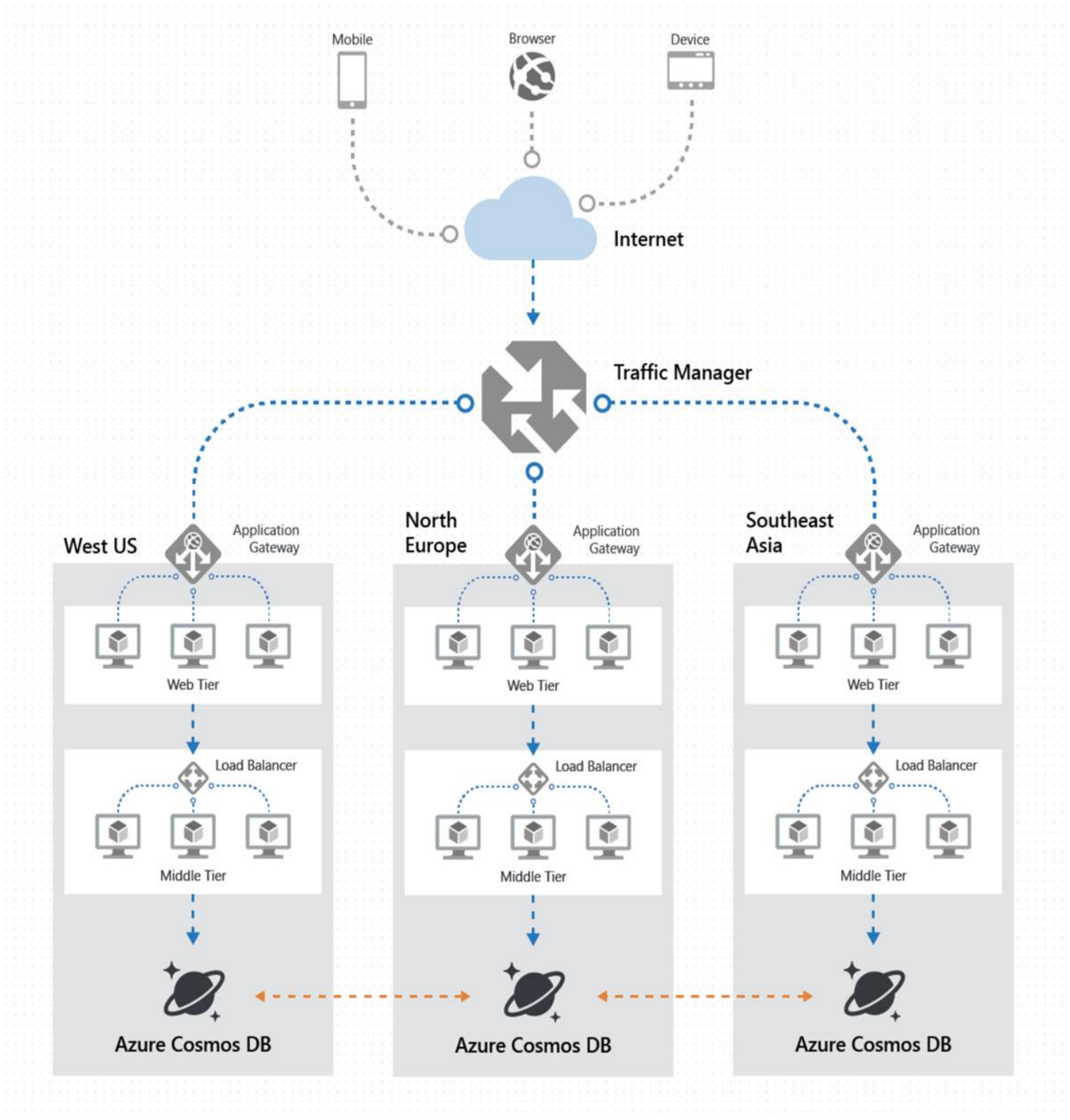


Figure 9 Azure Cosmos DB Global Distribution (Microsoft Azure 2019)

2.3 Cloud Computing

By considering the importance of cloud computing in the modern technology age, the author is going to discuss the concept of cloud computing and then profoundly explores Azure in a comparative study.

2.3.1 The Concept of Cloud Computing

Cloud computing refers to accessing computing services like servers, storage, networking, software over the internet from a provider like Azure (Microsoft Azure 2019). Hybrid cloud file transfer manages file transfer within a private or public cloud, and linking them provides control over the hybrid cloud environment. This allows securely transfer files between local file servers and storage cloud services like Azure or AWS. (Stonebranch 2019.)

2.3.2 The Highlights of Cloud Computing

Cloud computing includes valuable characteristics, including the most affordable price, higher security, more efficiency, and productivity on an international scale. Figure 10 contains the highlights of cloud computing as an on-demand resource.

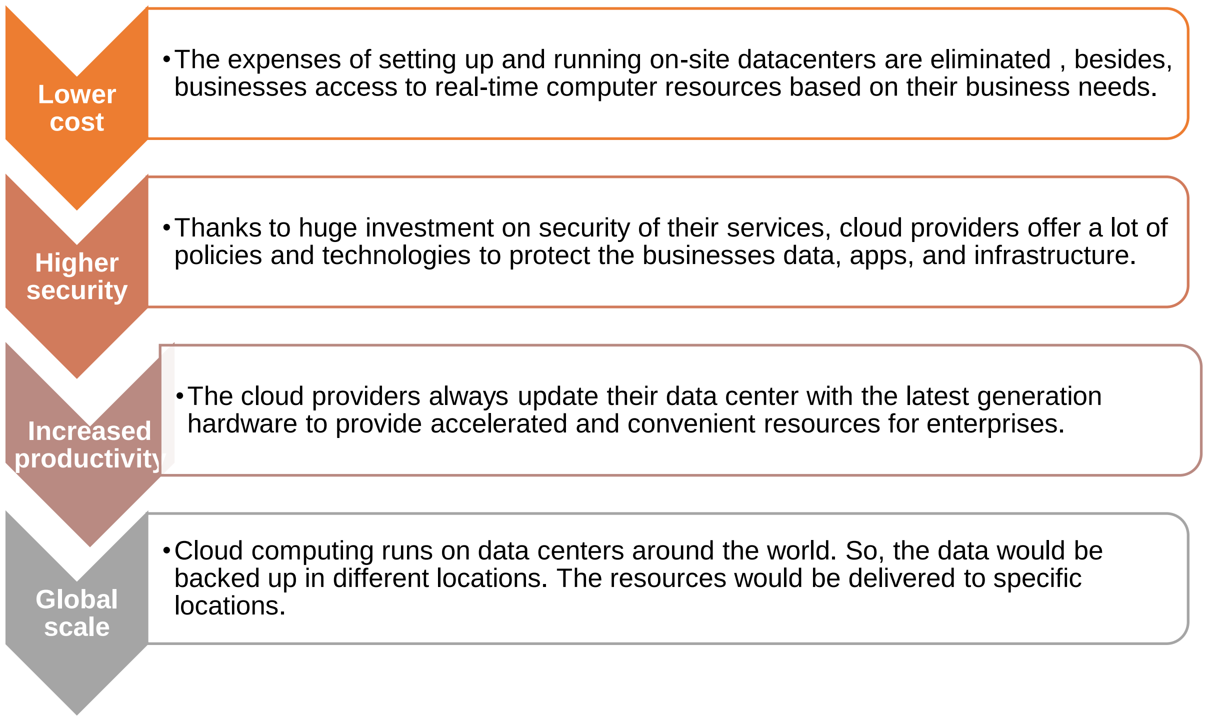

Lower cost
•The expenses of setting up and running on-site datacenters are eliminated , besides, businesses access to real-time computer resources based on their business needs.
Higher security
•Thanks to huge investment on security of their services, cloud providers offer a lot of policies and technologies to protect the businesses data, apps, and infrastructure.
Increased productivity
•The cloud providers always update their data center with the latest generation hardware to provide accelerated and convenient resources for enterprises.
Global scale
•Cloud computing runs on data centers around the world. So, the data would be backed up in different locations. The resources would be delivered to specific locations.

Figure 10 Cloud Computing Highlights (adapted from Microsoft Azure 2019)

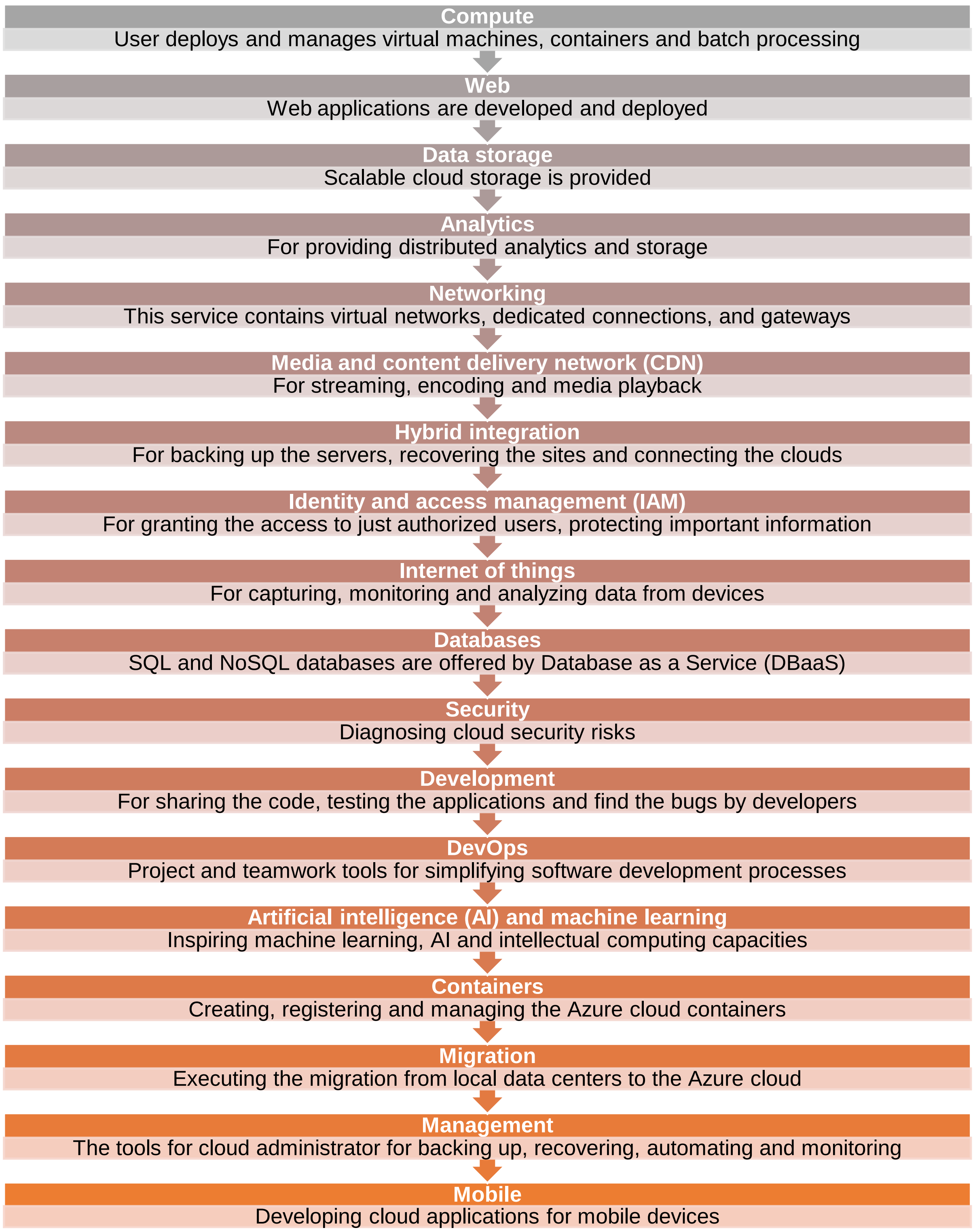
2.3.3 Azure vs. AWS

Azure Opening Remarks

There are a variety of cloud providers such as Amazon, Rackspace, and Microsoft Azure as developing and deploying application platform. Microsoft Azure delivers powerful hardware-level virtualization such as compute and storage resources. The possibility of computing on virtual parallel clusters adds up to Microsoft Azure. (Jackson 2010.)

Azure is supported by Microsoft and contains a set of cloud computing services for building, managing, and deploying the applications on a global network by different tools and frameworks (Microsoft Azure 2019). Azure offers several cloud services, such as computing, analyzing, storing, and networking for developing, scaling, or running applications.

Azure includes 18 categories, as shown in Figure 11:


Compute
User deploys and manages virtual machines, containers and batch processing

Web
Web applications are developed and deployed

Data storage
Scalable cloud storage is provided

Analytics
For providing distributed analytics and storage

Networking
This service contains virtual networks, dedicated connections, and gateways

Media and content delivery network (CDN)
For streaming, encoding and media playback

Hybrid integration
For backing up the servers, recovering the sites and connecting the clouds

Identity and access management (IAM)
For granting the access to just authorized users, protecting important information

Internet of things
For capturing, monitoring and analyzing data from devices

Databases
SQL and NoSQL databases are offered by Database as a Service (DBaaS)

Security
Diagnosing cloud security risks

Development
For sharing the code, testing the applications and find the bugs by developers

DevOps
Project and teamwork tools for simplifying software development processes

Artificial intelligence (AI) and machine learning
Inspiring machine learning, AI and intellectual computing capacities

Containers
Creating, registering and managing the Azure cloud containers

Migration
Executing the migration from local data centers to the Azure cloud

Management
The tools for cloud administrator for backing up, recovering, automating and monitoring

Mobile
Developing cloud applications for mobile devices

Figure 11 Azure Categories (adapted from Rouse 2018)

There is an outstanding difference between using Azure and IaaS infrastructures respecting the learning curve, particularly for developers experienced in developing and writing software for Linux systems. (Shanahan, Owen and Harrison 2014).

AWS Foreword

Amazon Web Services (AWS) is the most inclusive cloud platform which provides 165 different services from global data centers. AWS is a cloud service that offers database storage, content delivery, compute power, and other services for different businesses. (Amazon 2019.) Figure 12 shows the AWS provided services:

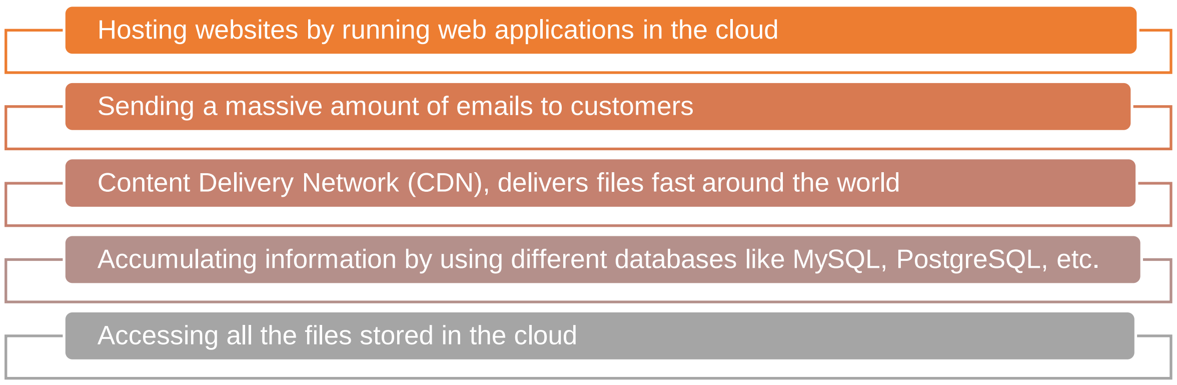

Hosting websites by running web applications in the cloud
Sending a massive amount of emails to customers
Content Delivery Network (CDN), delivers files fast around the world
Accumulating information by using different databases like MySQL, PostgreSQL, etc.
Accessing all the files stored in the cloud

Figure 12 Provided Services by AWS (adapted from Yadav 2018)

Comparing Azure with AWS

The cloud service providers are compared based on business variations, available services, and facilities. Microsoft entered the cloud Infrastructure as a Service (IaaS) market in June 2012 by introducing Azure Virtual Machines. AWS started the cloud IaaS market in 2006. Azure and AWS can be compared based on the following features: (Bala, Gill, Smith, and Wright 2019).

Offering

Both AWS and Azure are integrated IaaS + Platform as a Service (PaaS).

Locations

The AWS data centers are grouped into regions, and region partially involves two data centers. There are numerous regions in many countries. The Azure data centers are called regions, and there are multiple Azure regions around the world.

Profile acceptance

AWS buys agile operations and uses classic designs of IT operations; Azure proposes two different groups of customers. Group one plans to extend infrastructure. Group two intends to integrate with Microsoft applications.

Recommended uses

AWS and Azure are suitable for all cases which need a virtual environment. Figure 13 compares the AWS and Azure specifications:

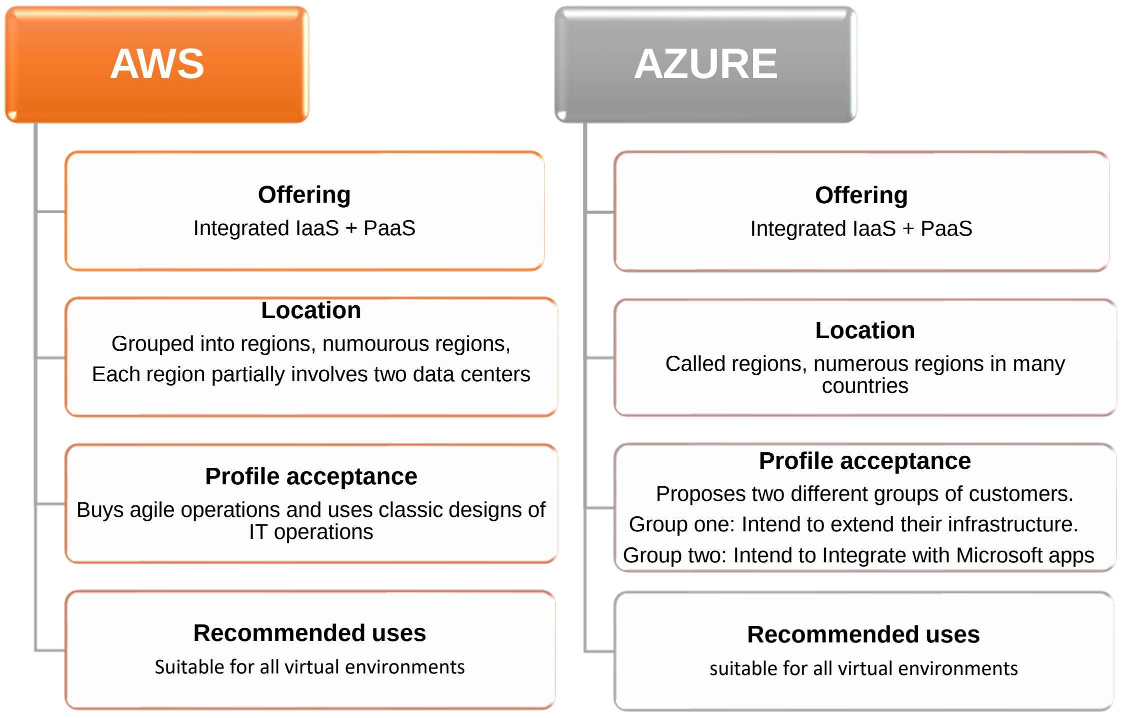

AWS
Offering
Integrated IaaS + PaaS
Location
Grouped into regions, numourous regions,
Each region partially involves two data centers
Profile acceptance
Buys agile operations and uses classic designs of IT operations
Recommended uses
Suitable for all virtual environments
AZURE
Offering
Integrated IaaS + PaaS
Location
Called regions, numerous regions in many countries
Profile acceptance
Proposes two different groups of customers.
Group one: Intend to extend their infrastructure.
Group two: Intend to Integrate with Microsoft apps
Recommended uses
suitable for all virtual environments

Figure 13 AWS Vs. AZURE (adapted from Bala, Gill, Smith & Wright 2019)

Gartner Research categorized the cloud service provider to four areas: Niche Players, Visionaries, Challengers, and Leaders. Alibaba Cloud, Oracle, and IBM are in the Niche Players section. Amazon Web Services, Microsoft, and Google are in the” Leader’s” part. AWS is the best in the ability to execute and completeness of vision. To have a better insight, cloud service infrastructures is compared in Figure 14.

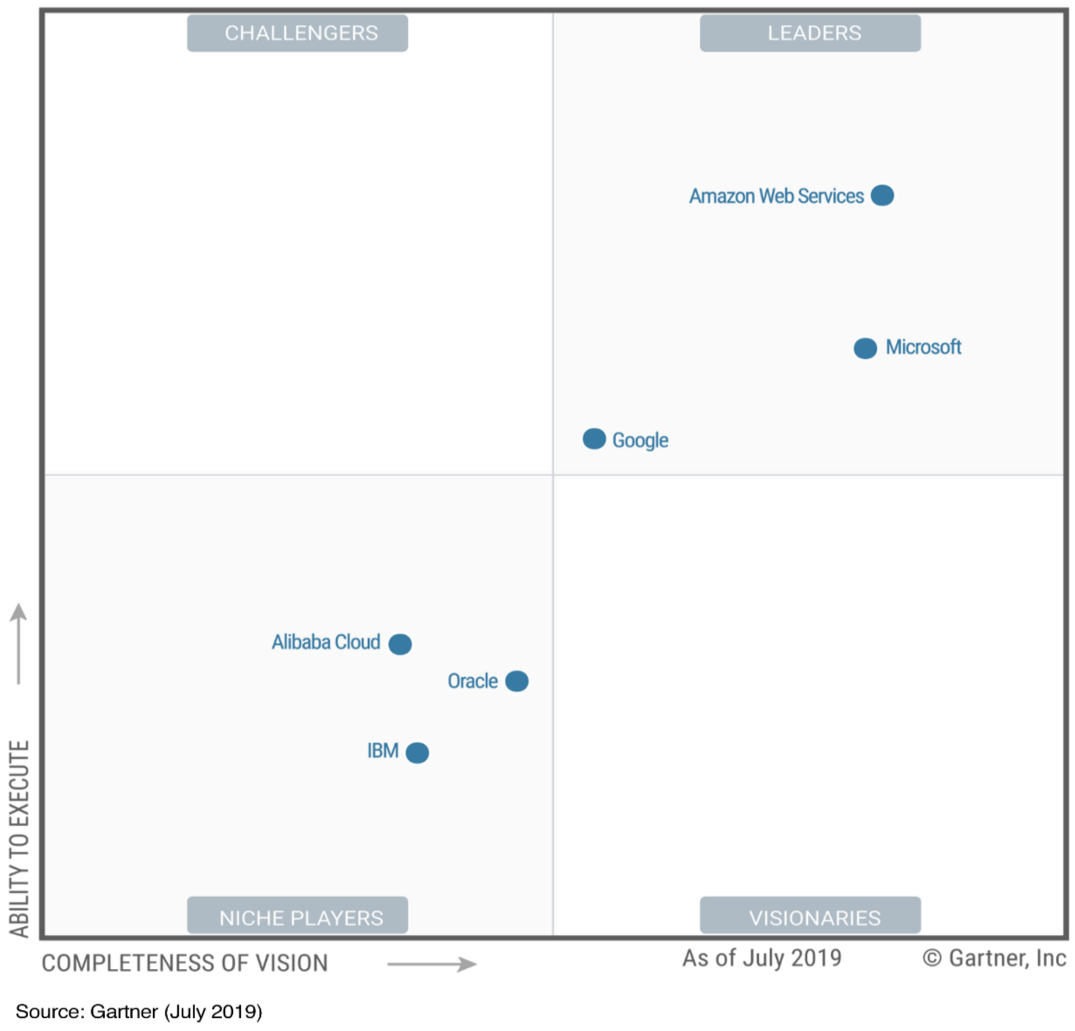


Figure 14 Comparing Cloud Service Infrastructure (Gartner 2019)

After brushing up the React Native, NoSQL, and Cloud computing background, the author is going to introduce the Jukola project on the next part.

2.4 Jukola Project

2.4.1 Jukola Background

Launched in 1949 in Finland, Jukola is the biggest and most popular world relay orienteering competition. Jukola is organized by a unique group of selected Finnish orienteering clubs. In the orienteering competition center, usually, a small town with full service including restaurants, sport equipment shops, sauna, self-service laundry, electricity, sanitary facilities, etc. is built. (Jukola 2019.) Since 2018, the Jukola event ought to have one mobile app for each annual event. Avenla Oy developed Jukola 2018 mobile application. Jukola 2019 mobile application was developed by Aprikoodi Oy. Currently, Tietotalo Infocenter Oy develops the Jukola 2020, and the thesis author, as a member of its development team, after a preface about the Jukola event, explains more about the project and its developing process in next parts.

2.4.2 Jukola Event

Jukola event as the world's biggest competitions of relay orienteering annually attracts over nearly 30,000 – 40,000 spectators; it always takes place on the third Saturday of June. “JUKOLA” is composed of “Jukola” relay, (7 men in a team) and the “Venla” relay (4 women in a team).

Figure 15 Jukola 2019 Tents (Jukola 2019) Photo by Jusso Lahto

The first Jukola relay attracted 41 teams in total; Nowadays, there are over 1,500 teams in the male Jukola relay and 1,100 teams in Venla relay from 20 countries. The distance for the male relay is 7 to 15 kilometers; the competition starts at 11 pm, and the winning team usually crosses the finish line next Sunday 6-7 am, and the last team comes to finish just before 2 pm. The female Venla relay starts at approximately 3 pm on Saturday. The distance is 5-8 kilometers, and the best teams use nearly three hours before crossing the finish line, and the last team usually finishes at the same time as the male Jukola relay. (Jukola 2019.)

2.4.3 Jukola Project

From 2018, There is one mobile app for the event each year. Avenla Oy developed Jukola 2018 mobile application. As mentioned above, Aprikoodi Oy developed Jukola 2019 application. Tietotalo Infocenter Oy is developing the Jukola 2020 application. The development of the teaser app (MVP) started in May 2019 and finished in June 2019. The app was used in Jukola 2019 event for registering the users for the Jukola 2020 event. The experience was successful, and about 250 emails gathered. The development team continued its development. The deadline for the Jukola 2020 application project is May 2020.

3 DEVELOPING JUKOLA PROJECT

In this study, the MVP of the Jukola applications will be explained in detail. It contains the following features:

1. Video splash screen
2. Intro page
3. Email registration
4. Main View

The front and back were developed simultaneously. The thesis author developed the backend part as well as the email registration page of MVP. Figure 16 shows the Jukola 2020 app home screen.

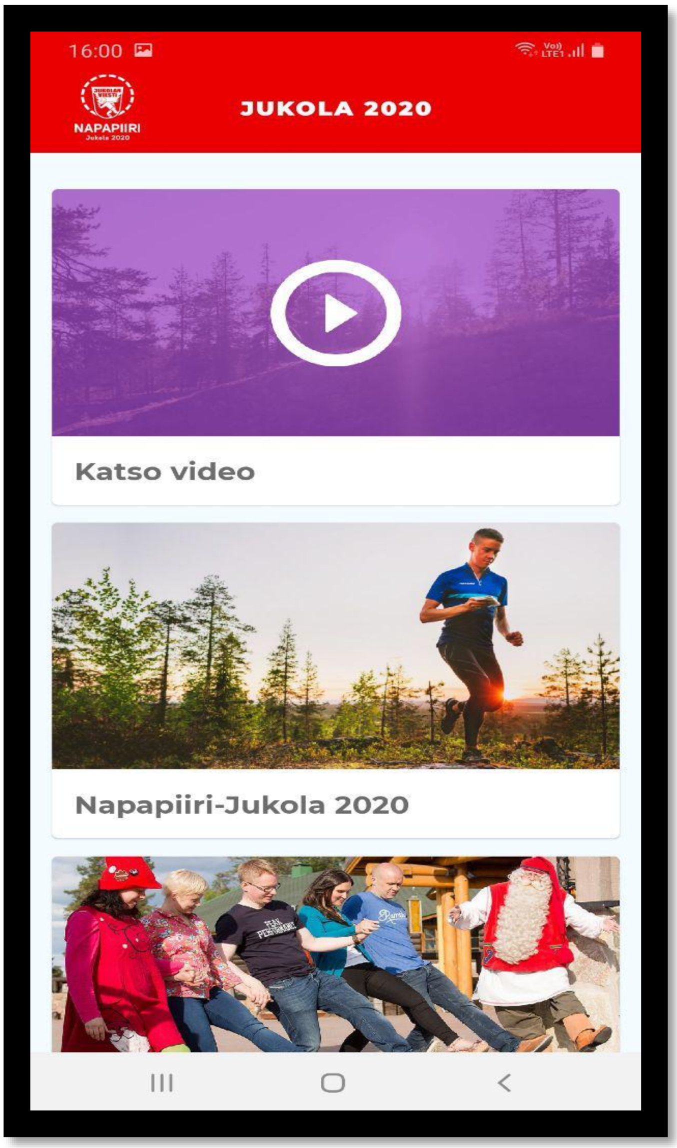


Figure 16 Jukola 2020 Application Home Screen

3.1 Why React Native?

The Jukola project includes developing the application for both Android and iOS platforms. As the leading developer of the project team, the thesis author was free to choose the developing framework from available cross-platform development tools. There were a variety of cross-platform development tools such as Xamarin, React Native, and Flutter. Before highlighting the reasons for selecting React Native, the author will briefly review the specifications of Flutter, Xamarin then React Native.

3.1.1 Flutter

Flutter is Google’s UI toolkit for developing natively compiled applications from a codebase for mobile, web, and desktop (Flutter 2019). Flutter released in December 2018. It collects the source code to native code. It also provides shared codes for platform-specific such as Android and iOS UI design. It looks fascinating and worth to learn. (Vries 2019.) Since the author was not familiar enough with Flutter, and the deadline was very tights (two weeks), the author skipped using Flutter.

3.1.2 Xamarin

Xamarin is a cross-platform mobile application development tool, owned by Microsoft since 2016. Xamarin uses the programming language C# to build apps for all mobile platforms. About 90 percent of shared codes are used across the Android and iOS platforms. It is more challenging to use the Xamarin community help compared to the React Native community. The Xamarin community is smaller than React and Native iOS and Android communities. Stack Overflow survey shows 7.4 percent of developers use Xamarin comparing to 28.3 percent of React developers. (AltexSoft 2019.)

Since the thesis author, as the leading developer of Jukola 2020 development team, is already experienced in maintenance tasks for a Xamarin mobile app with many users, he compared Hot-reload, application performance, libraries and tooling, the third-party support, UI and the community of each of these development platforms to select the best fitting one for the Jukola project. Figure 17 compares the Xamarin application with other frameworks.

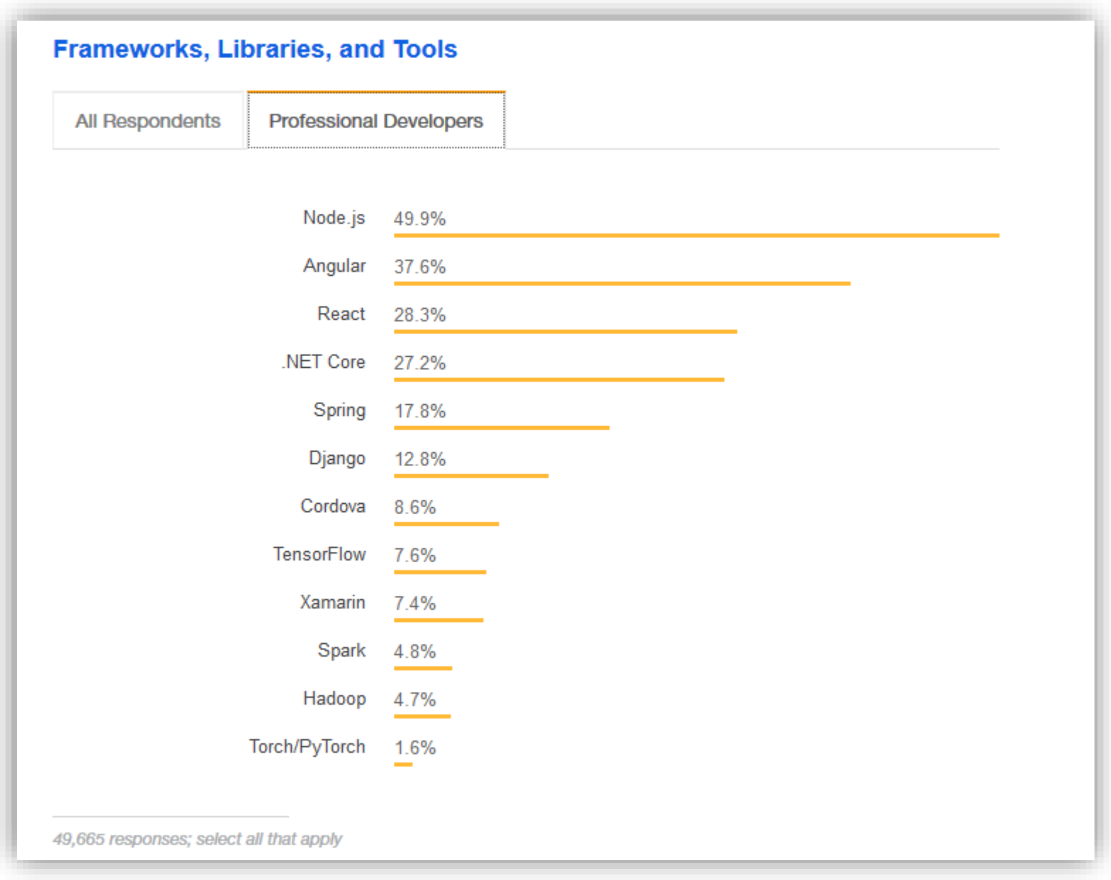
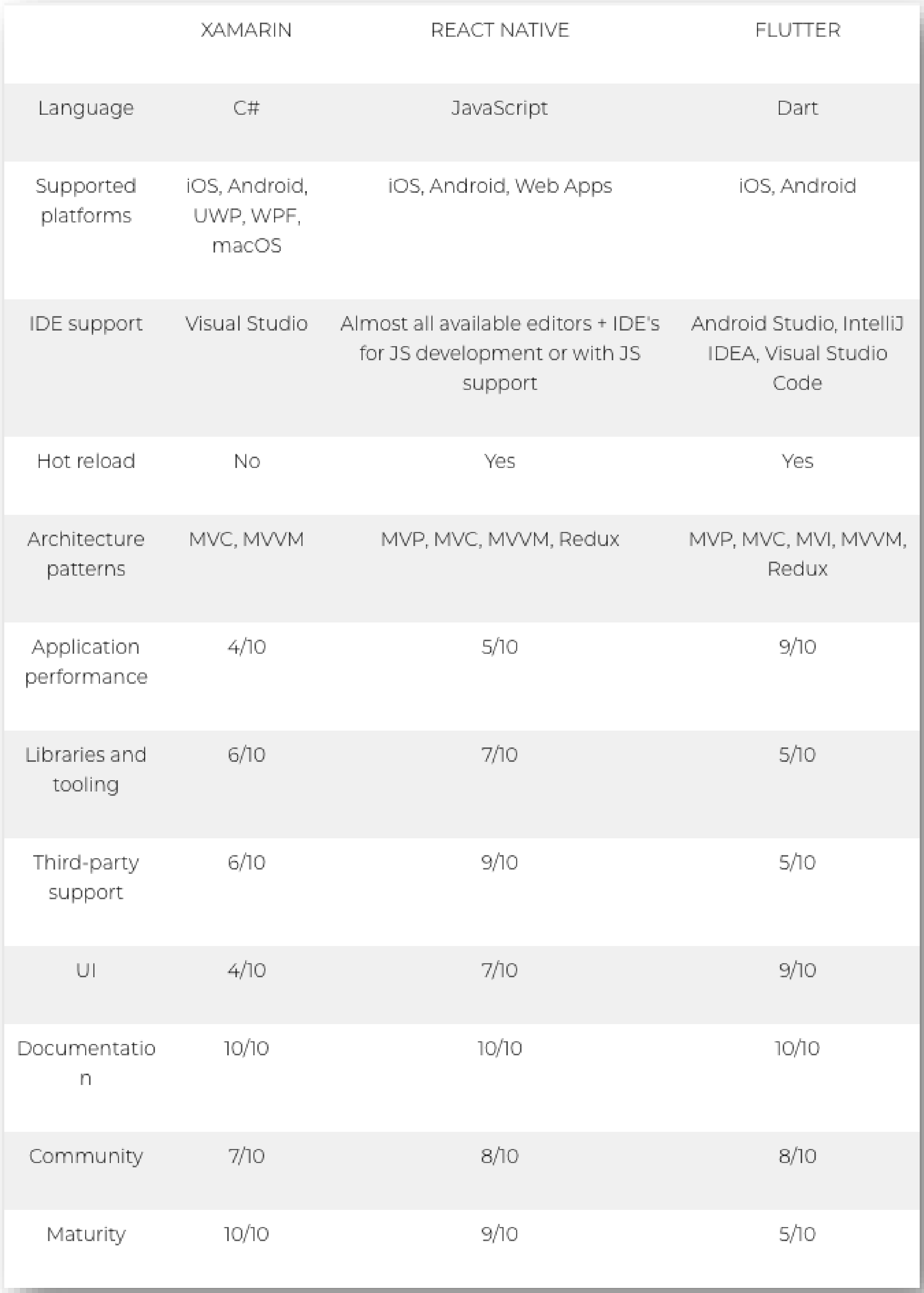


Figure 17 Xamarin Usage Compared to Other Frameworks (AltexSoft 2019)

Considering Xamarin specifications, the author compares Flutter, Xamarin, and React Native features to select the appropriate developing platform. According to table 2, React Native is preferred to Xamarin regarding the following features:

- Hot reload
- Application performance
- Libraries and tooling
- Third-party support
- UI
- Community

Table 2 Mobile Application Development Platforms Comparison (Hackernoon. 2018)



	XAMARIN	REACT NATIVE	FLUTTER
Language	C#	JavaScript	Dart
Supported platforms	iOS, Android, UWP, WPF, macOS	iOS, Android, Web Apps	iOS, Android
IDE support	Visual Studio	Almost all available editors + IDE's for JS development or with JS support	Android Studio, IntelliJ IDEA, Visual Studio Code
Hot reload	No	Yes	Yes
Architecture patterns	MVC, MVVM	MVP, MVC, MVVM, Redux	MVP, MVC, MVI, MVVM, Redux
Application performance	4/10	5/10	9/10
Libraries and tooling	6/10	7/10	5/10
Third-party support	6/10	9/10	5/10
UI	4/10	7/10	9/10
Documentatio n	10/10	10/10	10/10
Community	7/10	8/10	8/10
Maturity	10/10	9/10	5/10

3.2 Creating Jukola 2020 React Native Application

The Jukola 2020 React Native application was created by the following commands:

```
react-native init JukolaApp
```

For running the project (Android platform):

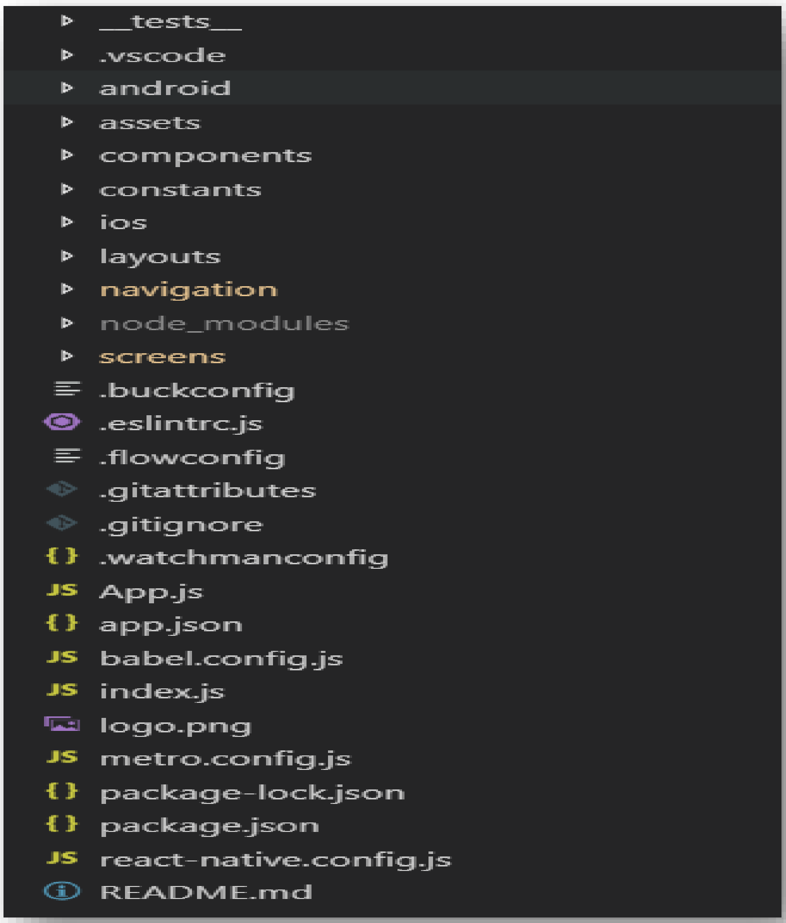
```
cd JukolaApp
react-native run-android
```

For running the project (iOS platform):

```
cd JukolaApp
react-native run-ios
```

3.2.1 Jukola Project Structure

Tietotalo development team defined Jukola 2020 project folder with the following contents (Figure 18):



```
__tests__
.vscode
android
assets
components
constants
ios
layouts
navigation
node_modules
screens
.buckconfig
.eslintrc.js
.flowconfig
.gitattributes
.gitignore
.watchmanconfig
App.js
app.json
babel.config.js
index.js
logo.png
metro.config.js
package-lock.json
package.json
react-native.config.js
README.md
```

Figure 18 Jukola 2020 Project Folder Structure

The project folder contains different subfolders, such as android, assets, components, constants, iOS, layouts, navigation, node-modules, and screens.

Components folder

The shared components which are applied in both iOS and Android application are placed in this directory such as Buttons, Status bar, Vector Assets, and Touchable components.

Constants directory

It contains the application styling, such as colors and components styling.

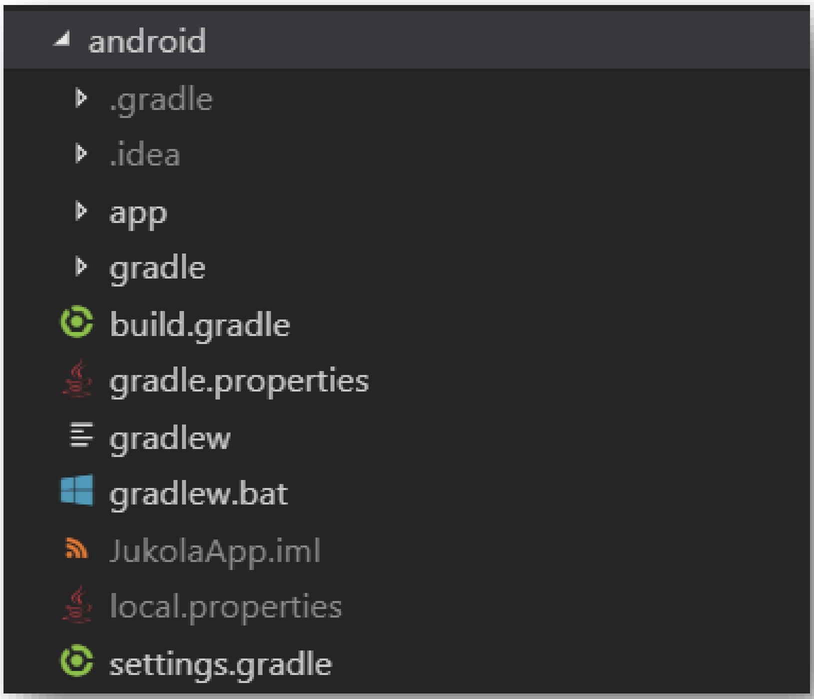
Navigation folder

It includes application navigation files.

Nod-modules folder

It consists of the Node Package Manager and Node Package Manager (NPM) packages.

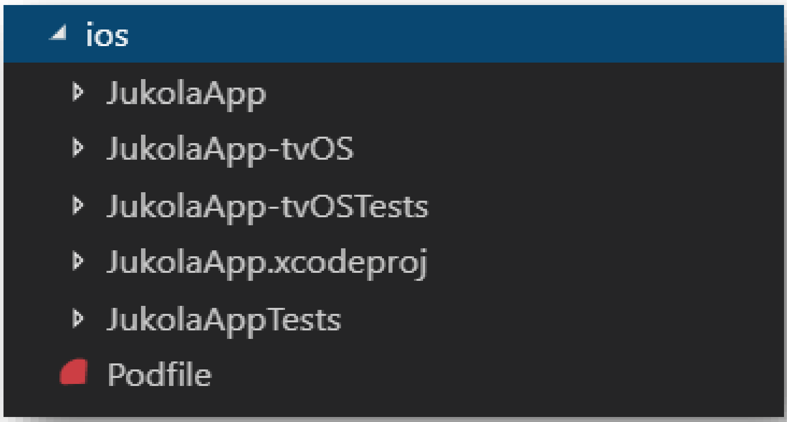
The Android folder (Figure 19) contains the following files and folders:



```
◢ android
   ▸ .gradle
   ▸ .idea
   ▸ app
   ▸ gradle
   build.gradle
   gradle.properties
   gradlew
   gradlew.bat
   JukolaApp.iml
   local.properties
   settings.gradle
```

Figure 19 Jukola 2020 Android Folder

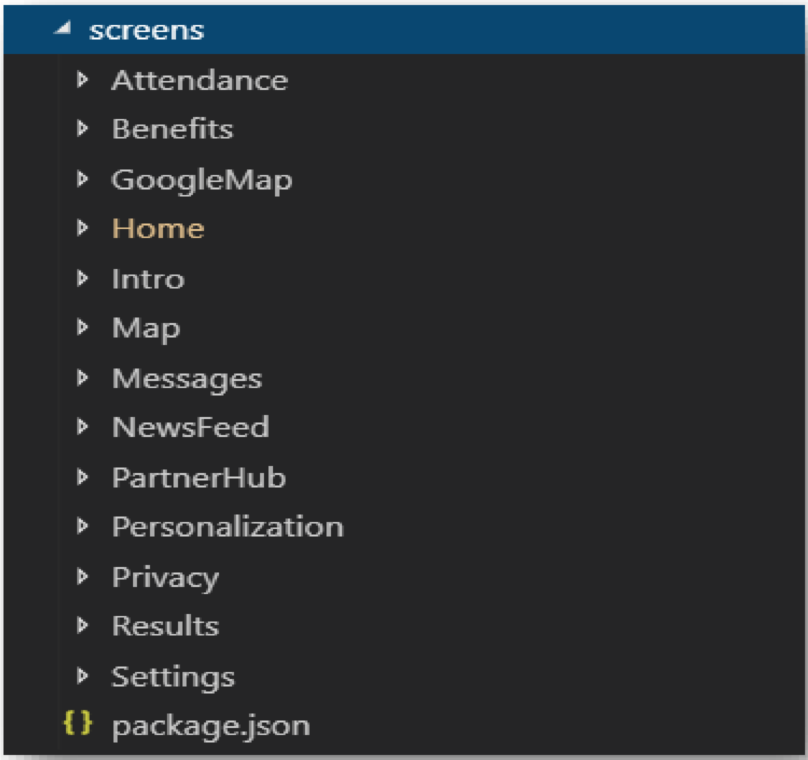
Jukola 2020 iOS folder (Figure 20), includes the following content:



```
ios
  JukolaApp
  JukolaApp-tvOS
  JukolaApp-tvOSTests
  JukolaApp.xcodeproj
  JukolaAppTests
  Podfile
```

Figure 20 Jukola 2020 iOS Folder

The assets folder is used for managing the fonts, images, and videos in the iOS and Android applications. It includes fonts, images, and video subfolders. The screen directory, which contains all of the screens, is illustrated in Figure 21.

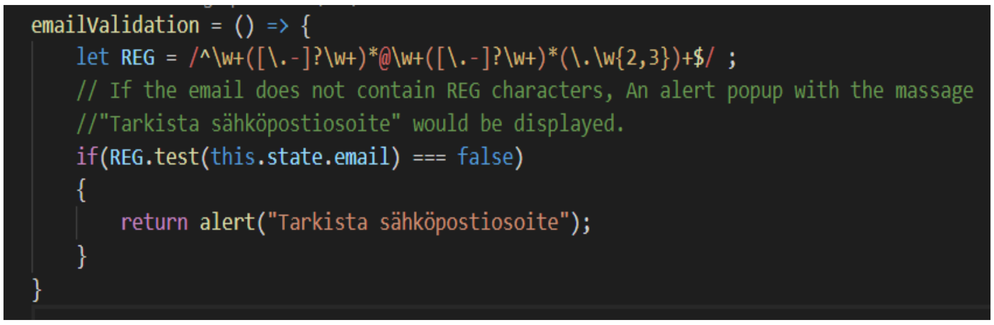


```
screens
  Attendance
  Benefits
  GoogleMap
  Home
  Intro
  Map
  Messages
  NewsFeed
  PartnerHub
  Personalization
  Privacy
  Results
  Settings
  package.json
```

Figure 21 Jukola 2020 Screens Folder

3.2.2 Intro Screen

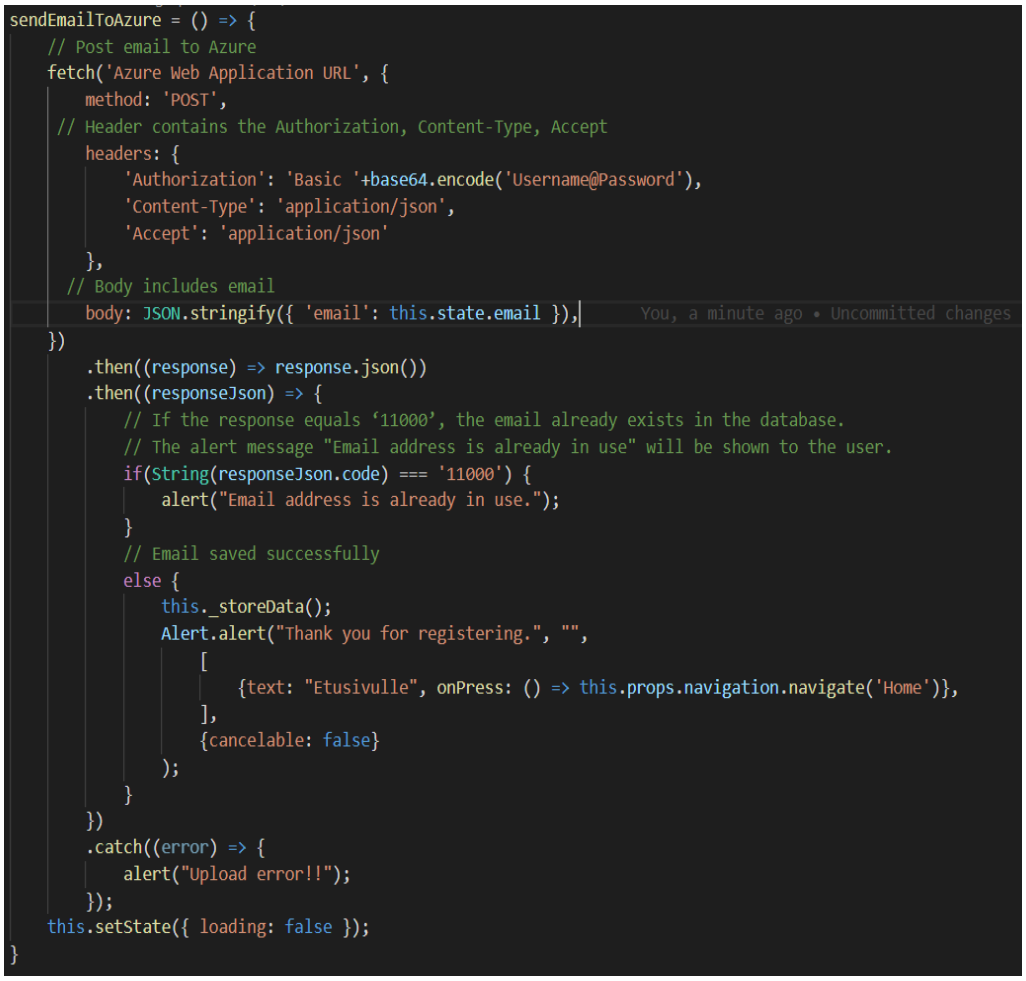
In the Intro screen, there is one input in which the user can enter the email. The email is validated by the function “emailValidation” which is illustrated in Figure 22. If the email format is not valid, the message “Check the email” will be displayed.



```
emailValidation = () => {
    let REG = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/ ;
    // If the email does not contain REG characters, An alert popup with the massage
    //"Tarkista sähköpostiosoite" would be displayed.
    if(REG.test(this.state.email) === false)
    {
        return alert("Tarkista sähköpostiosoite");
    }
}
```

Figure 22 Email Validation Function

If the email validation is true, the email will be sent to Azure by the function “sendEmailToAzure” Illustrated in Figure 23:



```
sendEmailToAzure = () => {
    // Post email to Azure
    fetch('Azure Web Application URL', {
        method: 'POST',
     // Header contains the Authorization, Content-Type, Accept
        headers: {
            'Authorization': 'Basic '+base64.encode('Username@Password'),
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        },
      // Body includes email
        body: JSON.stringify({ 'email': this.state.email }),        You, a minute ago • Uncommitted changes
    })
        .then((response) => response.json())
        .then((responseJson) => {
            // If the response equals ‘11000’, the email already exists in the database.
            // The alert message "Email address is already in use" will be shown to the user.
            if(String(responseJson.code) === '11000') {
                alert("Email address is already in use.");
            }
            // Email saved successfully
            else {
                this._storeData();
                Alert.alert("Thank you for registering.", "",
                    [
                        {text: "Etusivulle", onPress: () => this.props.navigation.navigate('Home')},
                    ],
                    {cancelable: false}
                );
            }
        })
        .catch((error) => {
            alert("Upload error!!");
        });
    this.setState({ loading: false });
}
```

Figure 23 ”Send Email to Azure Function”

3.3 Creating Jukola 2020 MongoDB

3.3.1 From the Jukola 2020 to MongoDB

In the Jukola 2020 Intro screen, figure 24, there is an input for entering the email. When the user clicks the “save” button, the value is sent to the MongoDB located in the Azure hosting service and stored on it.

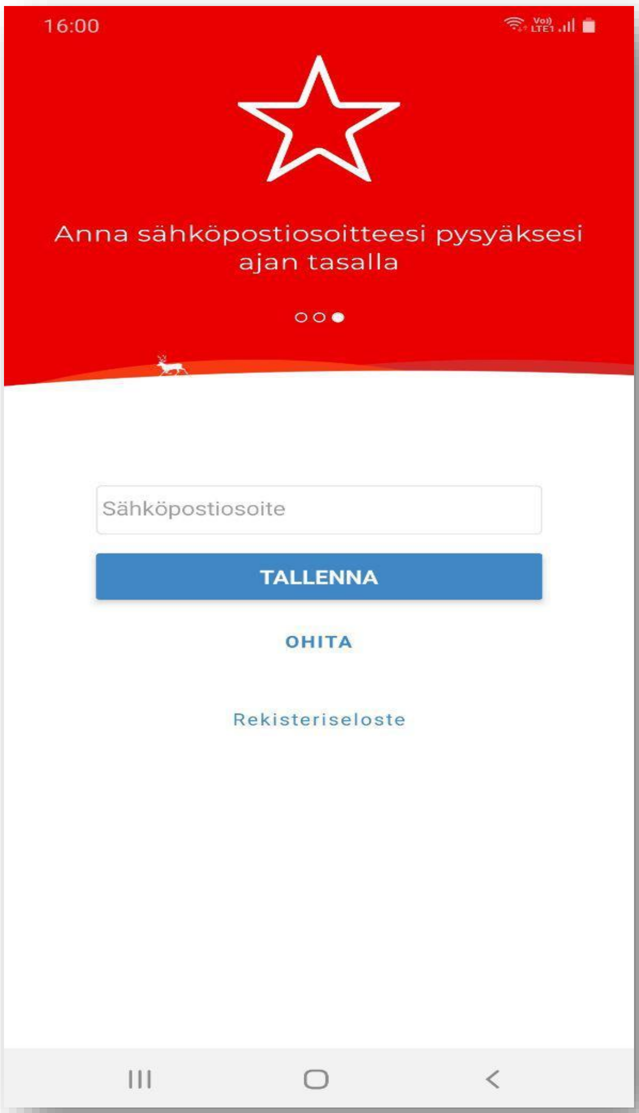


Figure 24 Jukola 2020 Intro Screen

3.3.2 Creating the Jukola Backend project

First, a directory was created called ‘JukolaBack.’ In that folder, the below command exe-cuted in the terminal:

```
C:\Mahyar\jukolaBack>npm init
```

Then, ExpressJS and Mongoose packages should be installed:

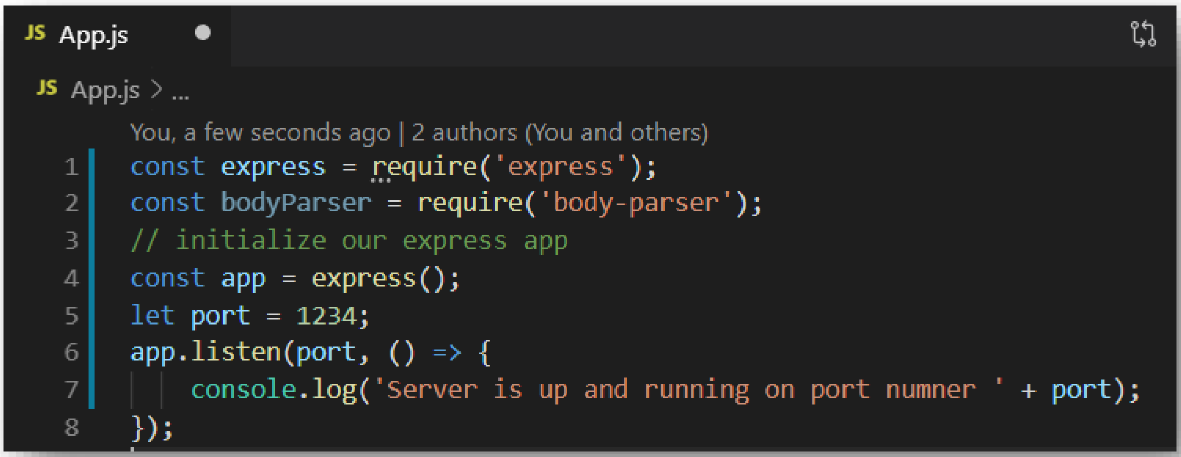
```
C:\Mahyar\jukolaBack>npm install -save express mongoose
```

3.3.3 Initializing the Server

To initialize the server, the file ‘app.js’ was created inside the ‘JukolaBack’ folder.

```
C:\Mahyar\jukolaBack>touch app.js
```

The next step is adding the following code (Figure 25), to the ‘app.js’ file:



```
const express = require('express');
const bodyParser = require('body-parser');
// initialize our express app
const app = express();
let port = 1234;
app.listen(port, () => {
    console.log('Server is up and running on port numner ' + port);
});
```

Figure 25 App.js file content

Now the server is ready and should be tested by the following command:

```
C:\Mahyar\jukolaBack>node app.js
```

3.3.4 Organizing the Application

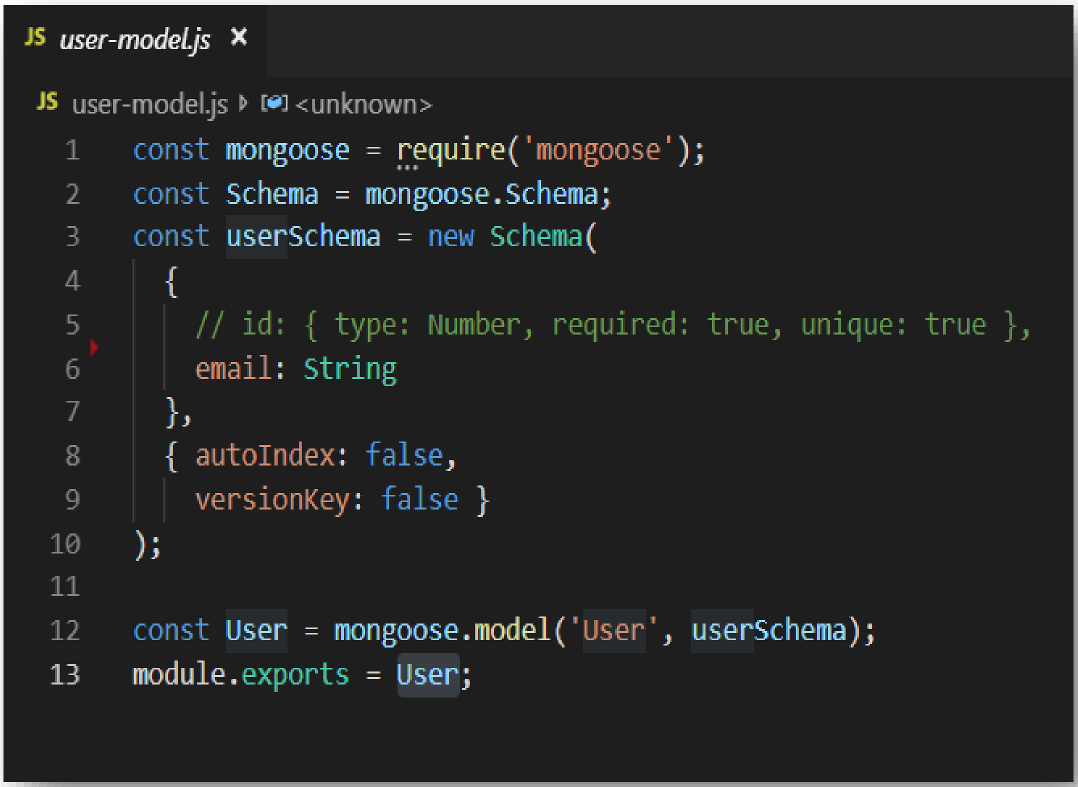
According to the app architecture, the following three sub-directories were created in the leading directory:

- env
- routes

- view

At the next step, the data model was defined. It was called ‘user-model.js’:

‘userSchema’ described on line 3 (Figure 26), which includes ‘email.’ ‘email’ defined as a string variable and can contain letters, numbers, and other characters.



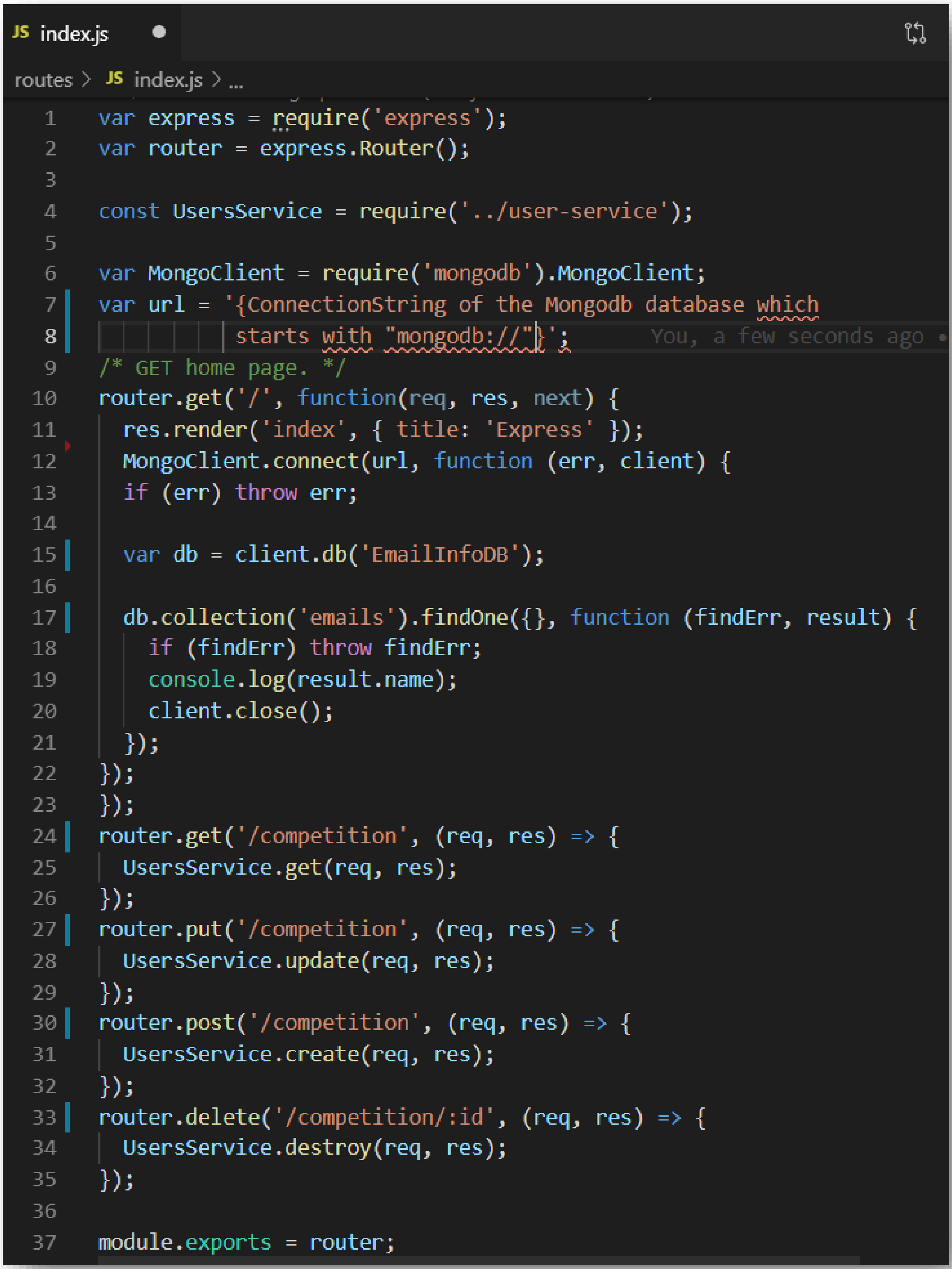
```
1   const mongoose = require('mongoose');
2   const Schema = mongoose.Schema;
3   const userSchema = new Schema(
4     {
5       // id: { type: Number, required: true, unique: true },
6       email: String
7     },
8     { autoIndex: false,
9       versionKey: false }
10  );
11
12  const User = mongoose.model('User', userSchema);
13  module.exports = User;
```

Figure 26 User-model.js File

Inside the routes folder, the ‘index.js’ file was created. This file includes the routes of the application. The following content was added to the file:

The variable ‘URL’ is declared on line 7. Its value is Mongo database Connection String and can be found in the Azure Cosmos database dashboard.

The functions ‘get’, ‘put,’ ‘post,’ and ‘delete’ are defined on lines 24, 27, 30, and 33 in Figure 27:



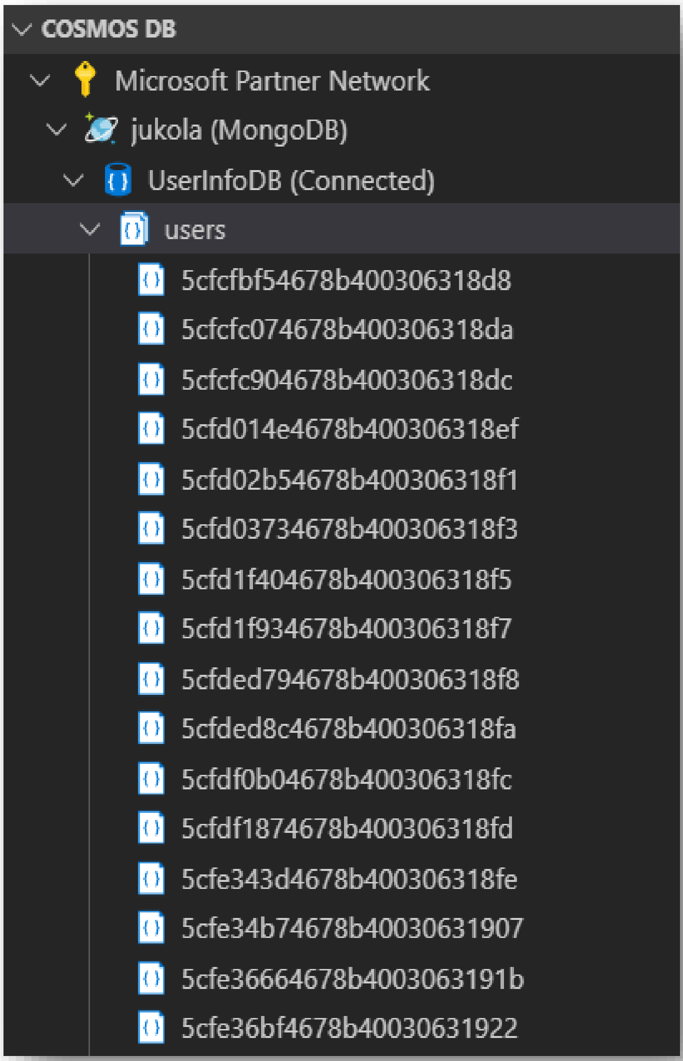
```
var express = require('express');
var router = express.Router();

const UsersService = require('../user-service');

var MongoClient = require('mongodb').MongoClient;
var url = '{ConnectionString of the Mongodb database which
          starts with "mongodb://"}';        You, a few seconds ago •
/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
  MongoClient.connect(url, function (err, client) {
  if (err) throw err;

  var db = client.db('EmailInfoDB');

  db.collection('emails').findOne({}, function (findErr, result) {
    if (findErr) throw findErr;
    console.log(result.name);
    client.close();
  });
});
});
router.get('/competition', (req, res) => {
  UsersService.get(req, res);
});
router.put('/competition', (req, res) => {
  UsersService.update(req, res);
});
router.post('/competition', (req, res) => {
  UsersService.create(req, res);
});
router.delete('/competition/:id', (req, res) => {
  UsersService.destroy(req, res);
});

module.exports = router;
```

Figure 27 Index.js File

As illustrated in figures 25 and 26, the model and routes of Jukola 2020 were organized aligned with the app architecture.

3.3.5 MongoDB Directory Structure

For storing the data, the Mongo database is used, as presented in figure 28. The Azure Cosmos DB account is 'jukola.' The database is 'UserInfoDB,' and the collection ID is 'users' as illustrated in figure 27.



```
COSMOS DB
  Microsoft Partner Network
    jukola (MongoDB)
      UserInfoDB (Connected)
        users
          5cfcfbf54678b400306318d8
          5cfcfc074678b400306318da
          5cfcfc904678b400306318dc
          5cfd014e4678b400306318ef
          5cfd02b54678b400306318f1
          5cfd03734678b400306318f3
          5cfd1f404678b400306318f5
          5cfd1f934678b400306318f7
          5cfded794678b400306318f8
          5cfded8c4678b400306318fa
          5cfdf0b04678b400306318fc
          5cfdf1874678b400306318fd
          5cfe343d4678b400306318fe
          5cfe34b74678b40030631907
          5cfe36664678b4003063191b
          5cfe36bf4678b40030631922
```

Figure 28 Jukola 2020 Local MongoDB

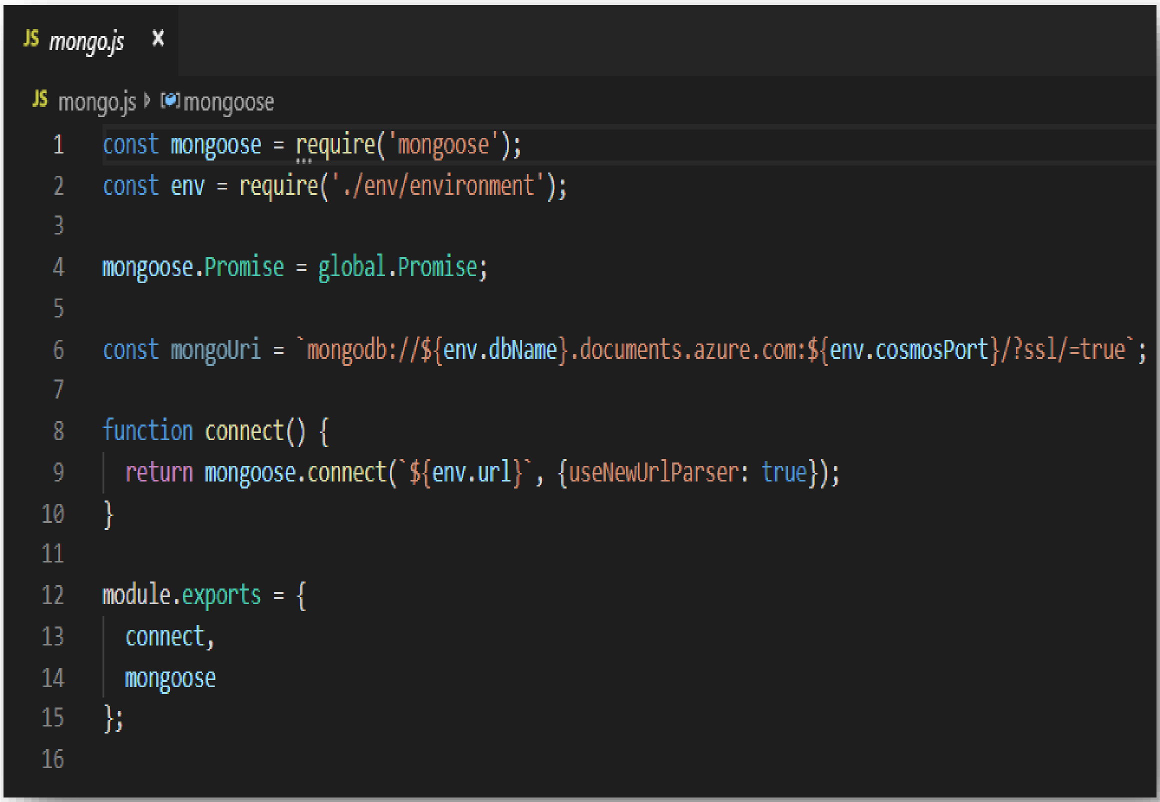
3.3.6 Connecting to the MongoDB

The "mongo.js" file was created for connecting to MongoDB.

The credentials values should be defined as illustrated in Figure 29, in the "environment.js" file to allow the application to function differently based on the environment (Local or Azure), such as the below variables:

- const cosmosPort

- const dbName
- const key
- const URL



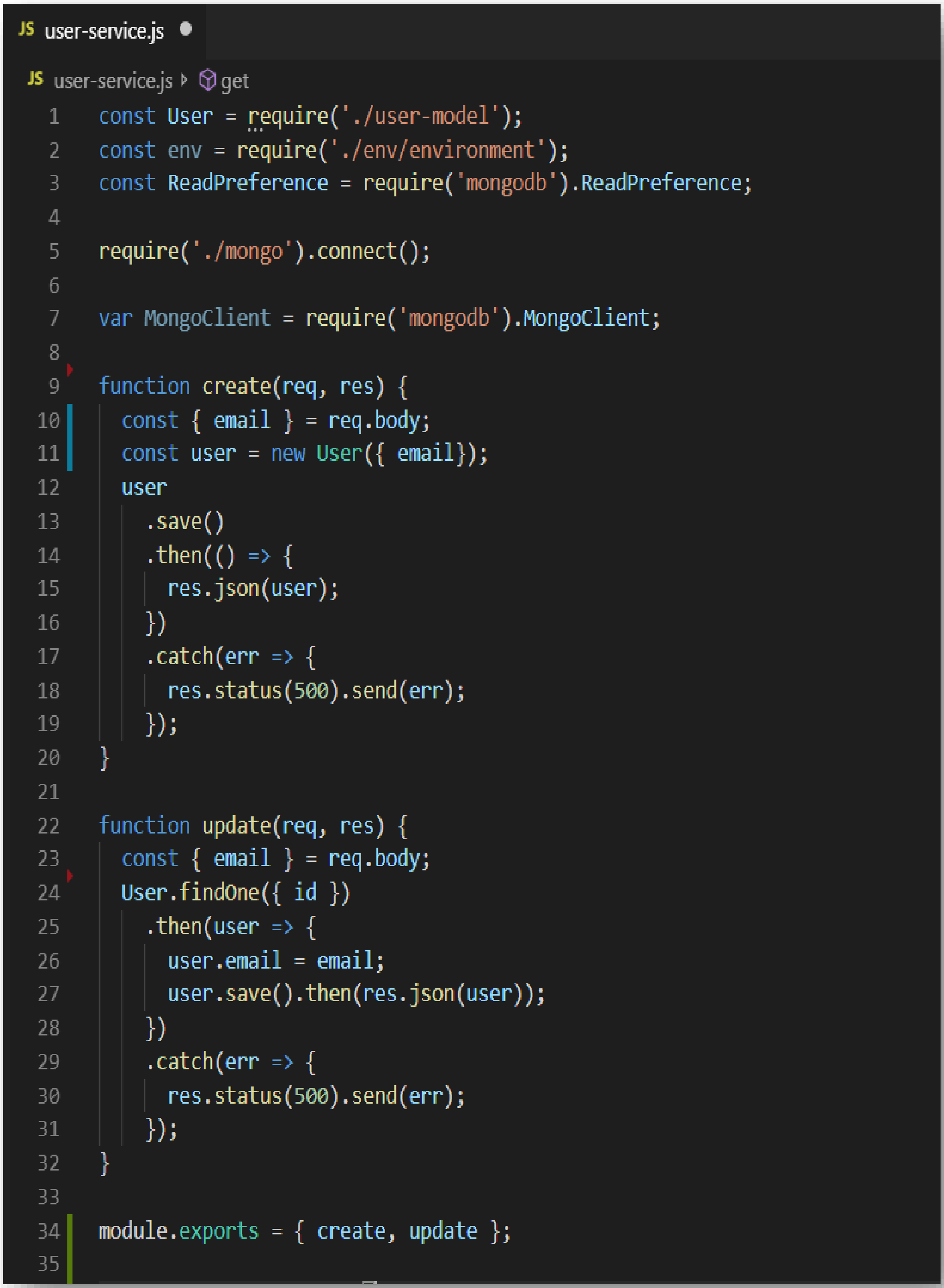
```
const mongoose = require('mongoose');
const env = require('./env/environment');

mongoose.Promise = global.Promise;

const mongoUri = `mongodb://${env.dbName}.documents.azure.com:${env.cosmosPort}/?ssl/=true`;

function connect() {
  return mongoose.connect(`${env.url}`, {useNewUrlParser: true});
}

module.exports = {
  connect,
  mongoose
};
```

Figure 29 Mongo.JS

For the data model, user-model.js is used. The user is defined by userSchema, and it contains email. The id will be set automatically.

3.3.7 Managing the Data

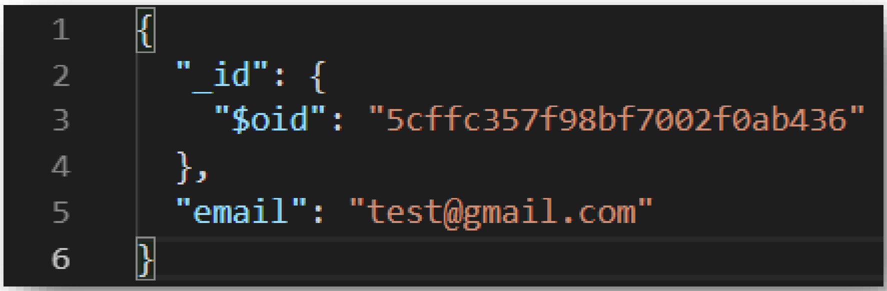
To manage the data, 'user-service.js' is applied, which is located in the root directory. For connecting to the Mongo database, ‘mongo.js’ was used (Figure 30, line 5). 'MongoClient' is defined by ‘mongodb’ (Figure 30, line 7). To add a new unique email to a database, the function 'update' updates the email value in the database (Figure 30, line 22).



```javascript
const User = require('./user-model');
const env = require('./env/environment');
const ReadPreference = require('mongodb').ReadPreference;

require('./mongo').connect();

var MongoClient = require('mongodb').MongoClient;

function create(req, res) {
  const { email } = req.body;
  const user = new User({ email});
  user
    .save()
    .then(() => {
      res.json(user);
    })
    .catch(err => {
      res.status(500).send(err);
    });
}

function update(req, res) {
  const { email } = req.body;
  User.findOne({ id })
    .then(user => {
      user.email = email;
      user.save().then(res.json(user));
    })
    .catch(err => {
      res.status(500).send(err);
    });
}

module.exports = { create, update };
```

Figure 30 User-Service.JS

Every data item in the database has one unique key (id) and one unique value (email).



```
{
  "_id": {
    "$oid": "5cffc357f98bf7002f0ab436"
  },
  "email": "test@gmail.com"
}
```

Figure 31 Data Item

3.3.8 How to Host the MongoDB on Azure?

The Visual Studio Code (2019) as a code editor is applied for building and debugging. This tool is free and available for Windows, Mac, and Ubuntu. It offers several helpful features:

- Smart completion based on imported modules and variables types.
- Debugging, by launching or attaching to running applications with breakpoints and debug console.
- Git commands, including staging files, making commits, push, pull, and other Git functions.
- Lots of extensions available for new programming languages, themes, debuggers, and additional services. (Visual Studio 2019.)

For connecting to Azure service, Azure extensions are utilized. There are several helpful Azure extensions for Visual Studio Code, including Azure API management, Azure App Service, Azure Cosmos DB, Azure Functions, Azure Cosmos DB emulator, etc.

Azure API Management which published and supported by Microsoft

This extension is used for performing management operations of Azure in the Visual Studio Code. The APIs can be exposed securely to external and internal consumers. It provides the following features:

- API Management instance can be deleted or created.
- An API can be created by importing an OpenAPI specification.
- APIs and operations in Azure Resource Manager or OpenAPI formats can be edited.

- Policies of any scope can be edited, as well. (Microsoft Azure 2019.)

Azure App Service offered and supported by Microsoft

Web, mobile, and API apps can be created, managed, deployed, and scaled quickly by this tool (Microsoft Azure 2019).

Azure Cosmos DB extension provided by Microsoft

MongoDB databases could be browsed and queried, both locally and in the cloud. The developer can manage the Cosmos DB databases by connecting to Azure. The extension offers the following features:

- A Cosmos DB account can be created easily by clicking the plus button in the title.
- The Azure Cosmos DB accounts can be viewed and opened in the portal directly.
- The databases, collections, graphs can be viewed, created, and deleted.
- A Mongo server can be attached by clicking the plug icon in the title. (Microsoft Azure 2019.)

Azure Functions extension developed by Microsoft

Azure Functions can be created, debugged, managed, and deployed by the extension (Figure 32). It supports the following functions:

- New functions projects and new functions from a template can be created.
- Function projects can be debugged locally and deployed to Azure.
- Azure Function Apps can be viewed, created, deleted, started, stopped, and restarted.

- Application settings can be viewed, edited, uploaded, and downloaded. (Microsoft Azure 2019.)

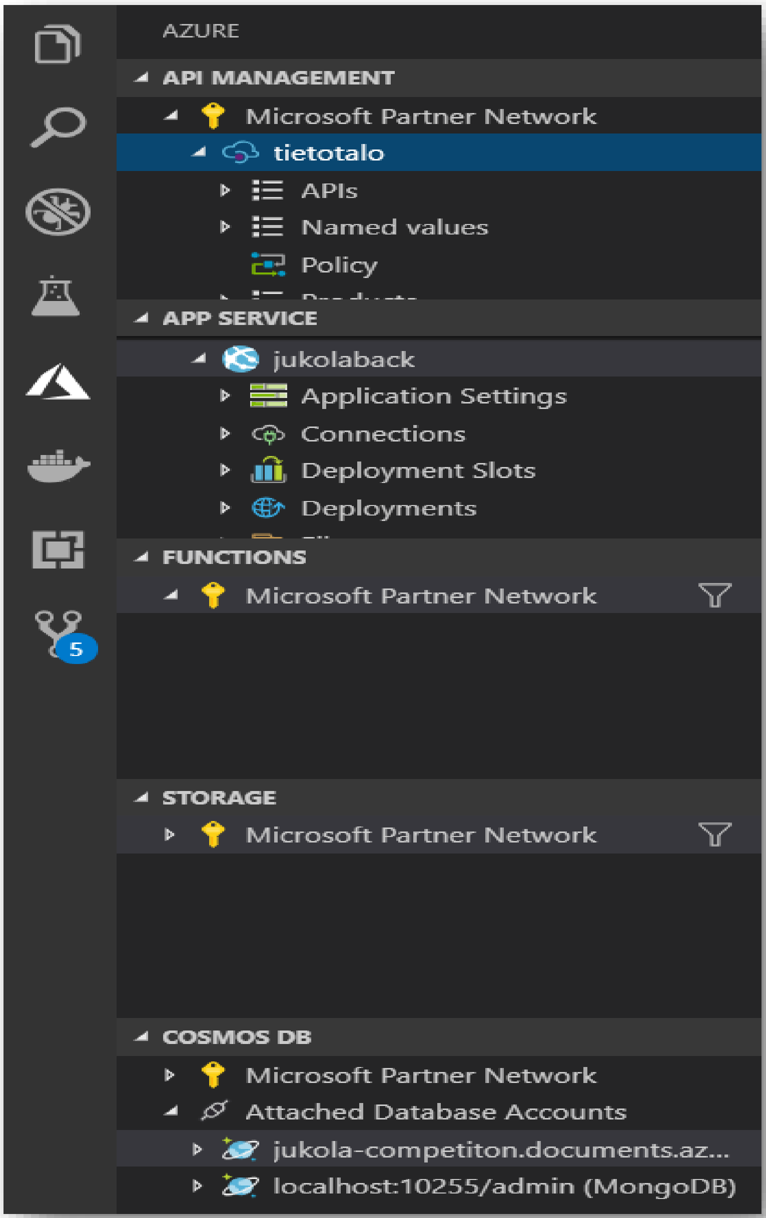

AZURE
API MANAGEMENT
Microsoft Partner Network
tietotalo
APIs
Named values
Policy
APP SERVICE
jukolaback
Application Settings
Connections
Deployment Slots
Deployments
FUNCTIONS
Microsoft Partner Network
STORAGE
Microsoft Partner Network
COSMOS DB
Microsoft Partner Network
Attached Database Accounts
jukola-competiton.documents.az...
localhost:10255/admin (MongoDB)
5

Figure 32 The Azure Extensions

Azure Cosmos DB Emulator

Azure DB provides a local environment that emulates the Azure Cosmos DB service for development purposes. The application can be developed and tested locally without Azure subscription and any costs. The developer switches the application from Azure Cosmos Emulator to Azure Cosmos account in the cloud, when it is ready. (Microsoft Azure 2019.)

Azure Cosmos Emulator can be downloaded and installed from the Microsoft Download Center. For starting the Azure Cosmos Emulator on Windows, the Windows key should be pressed and search for ‘Azure Cosmos Emulator’ and select it from the list of applications. When the emulator is running, it’s icon will be displayed in the Windows taskbar notification area. The Azure Cosmos Emulator opens the Azure Cosmos Data Explorer in the browser automatically (Figure 33), when it launches. (Microsoft Azure 2019.)

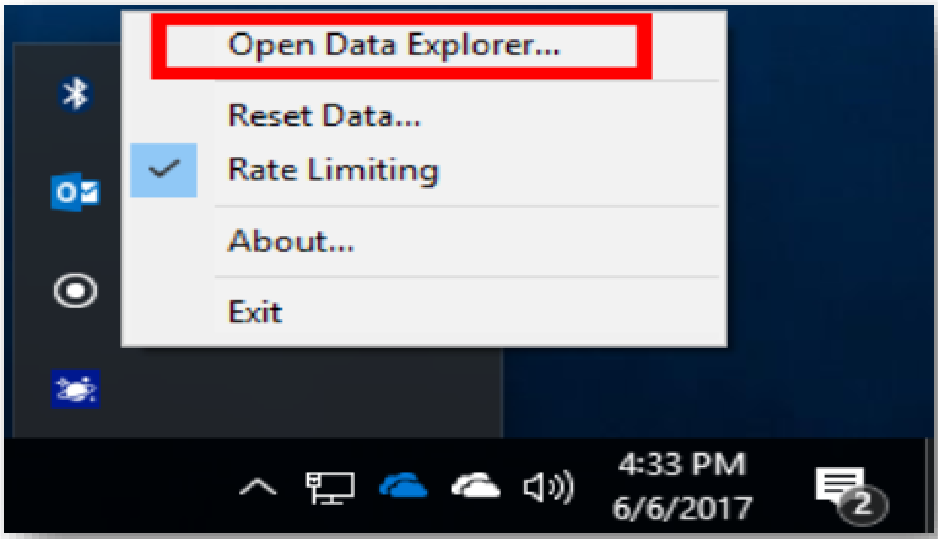


Figure 33 Azure Cosmos Emulator (Microsoft Azure 2019)

3.4 Creating Azure Database

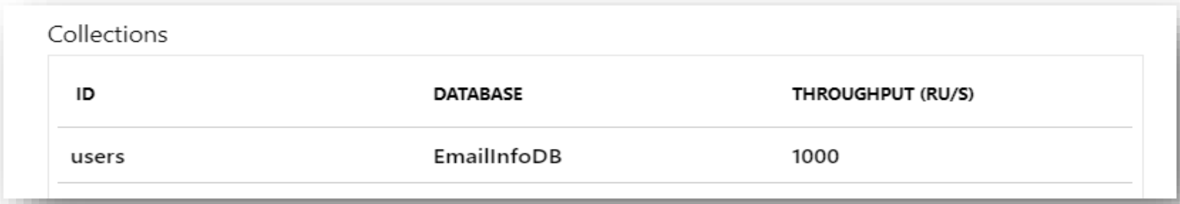
For creating the Azure database, the author applied the Azure Microsoft dashboard from the following link: portal.azure.com. In the dashboard, a new Resource group created in Azure with the name of ‘JukolaApp.’ It takes a couple of minutes to be built. Then it is time to add two resources:

1. App service
2. Azure Cosmos DB account

For the visibility of the missing updates, OS security settings, endpoint protection, and threat detections, the data collection is needed. Data collection is necessary for Compute resources such as virtual machines and non-Azure computers. (Microsoft Azure 2019.)

As illustrated in Figure 34, a new data collection can be added in the ‘overview’ part, which contains:

1. Id: users
2. Database: EmailInfoDB



Collections

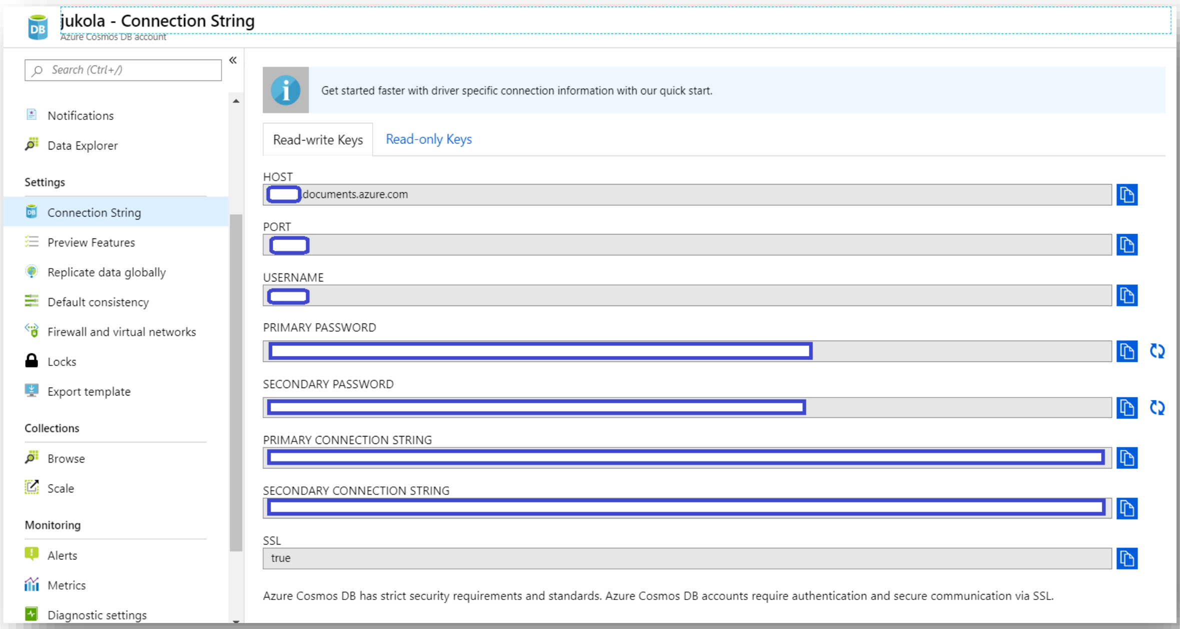
ID	DATABASE	THROUGHPUT (RU/S)
users	EmailInfoDB	1000

Figure 34 Azure Cosmos DB Collection

After creating the Azure Cosmos DB account, the Connection String is created automatically, including the host, port, username, primary password, secondary password, primary connection string, secondary connection string, and SSL information (Figure 35).

The Connection String is accessible in the following address:

Home => Resource groups => 'Azure Cosmos DB account Name' => Connection String



jukola - Connection String
Azure Cosmos DB account

Get started faster with driver specific connection information with our quick start.

HOST: [redacted]documents.azure.com
PORT: [redacted]
USERNAME: [redacted]
PRIMARY PASSWORD: [redacted]
SECONDARY PASSWORD: [redacted]
PRIMARY CONNECTION STRING: [redacted]
SECONDARY CONNECTION STRING: [redacted]
SSL: true

Azure Cosmos DB has strict security requirements and standards. Azure Cosmos DB accounts require authentication and secure communication via SSL.

Figure 35 Azure Cosmos DB Account Connection String

The connection string is used for connecting to the Cosmos database.

Adding New Collection to Cosmos Database

For adding a new collection, the developer should navigate the Azure Cosmos DB account Dashboard, Figure 36:

Collections => Browse => Add Collection

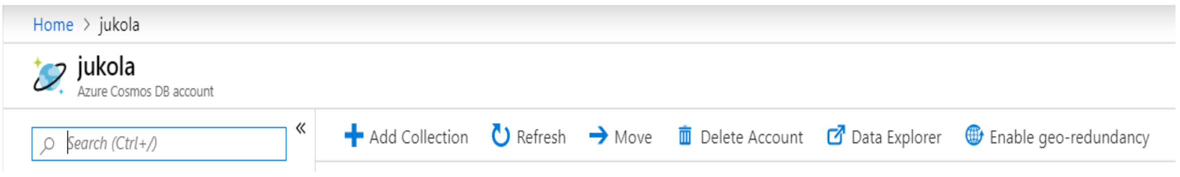
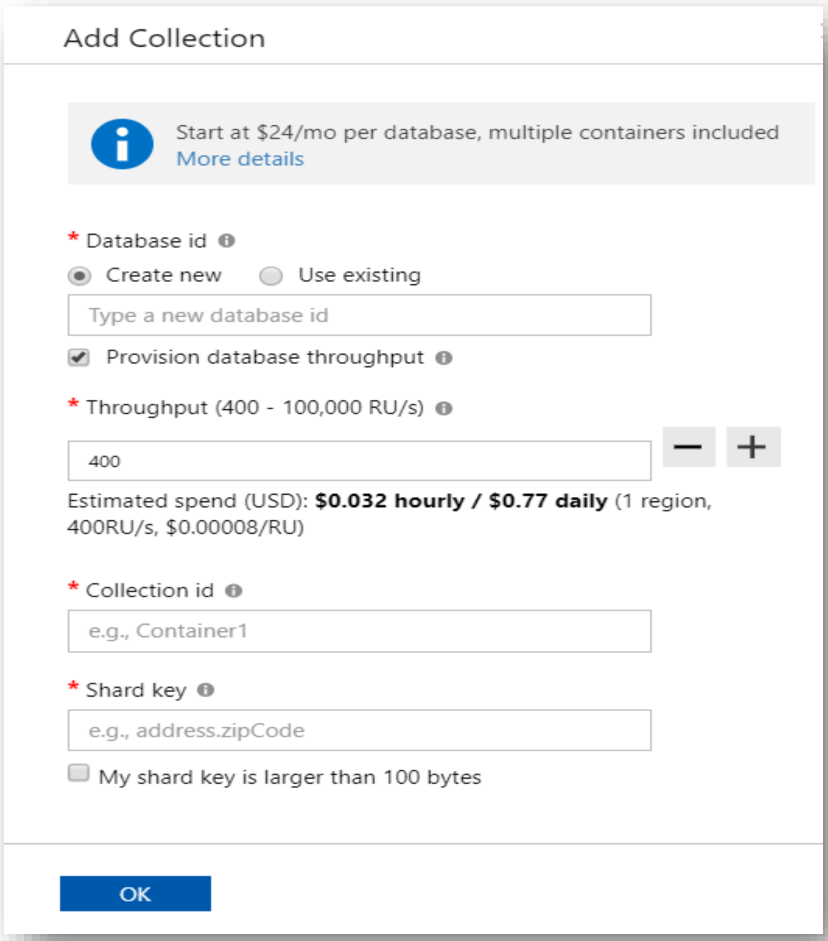


Figure 36 Azure Portal Add New Collection

After clicking the ‘Add Collection,’ a form for adding collection will be opened. This form, as depicted in Figure 37, includes the following mandatory fields: Database id, throughput, collection id, and shared key fields.

- Database id: It is the unit of management for a set of containers
- ‘Create new’ or ‘Use existing’ can be selected
- Throughput: The range is between 400 – 100,000 request unit per second (RU/s) The thesis author selected 400 for decreasing the costs.
- Collection id: It is the unit identifier for container and is used for id-based routing, for instance: Container1
- Shared Key: It is used for scalability for example address, zipCode



Add Collection

Start at $24/mo per database, multiple containers included
More details

* Database id
● Create new ○ Use existing
Type a new database id
☑ Provision database throughput

* Throughput (400 - 100,000 RU/s)
400
Estimated spend (USD): **$0.032 hourly / $0.77 daily** (1 region, 400RU/s, $0.00008/RU)

* Collection id
e.g., Container1

* Shard key
e.g., address.zipCode
☐ My shard key is larger than 100 bytes

OK

Figure 37 Adding Collection to Cosmos DB

3.5 Testing

Once the Cosmos DB is created on Azure, the Azure app, database, and mobile app are ready for testing. The saving data process can be tested either by both the emulator and the real device by sending the request to Azure. At this point, the new user can be added by inserting the email address. The latest user data should be updated to the Azure database immediately.

The testing phase brings us to the final development process of the Jukola 2020 teaser mobile application. Accordingly, the author discusses the study results in chapter 4.

4 CONCLUSIONS

4.1 Key Findings

After recapping the development procedure of the Jukola 2020 Teaser mobile application, the author is firstly going to answer all research questions in the first part of this chapter, and on the next step, he proposes suggestions for further researches regarding the study results.

This study successfully managed to answer all the research questions. The main question of the thesis was:

4.1.1 How to develop a hybrid mobile application by React Native, NoSQL database on Azure?

The developing process is explained in chapter 3 in detail. The author firstly started the project by developing the mobile application using the React Native platform. Then the Cosmos database as a NoSQL database was developed in the second phase. In the third phase, the Cosmos DB and Web Application hosted on Microsoft Azure. In the final step, the author tested the application. Figure 38 shows the Jukola 2020 Teaser Application Process:

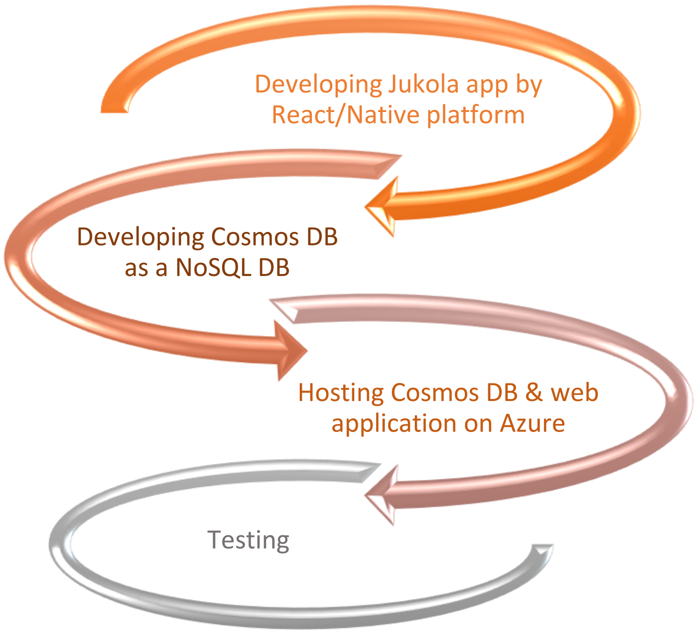

Developing Jukola app by React/Native platform
Developing Cosmos DB as a NoSQL DB
Hosting Cosmos DB & web application on Azure
Testing

Figure 38 Developing Jukola 2020 Teaser Application Process

4.1.2 Why React Native platform is applied for mobile development?

Considering the previous studies about the specifications of React Native compared to other platforms, this study admits the selected platform was a proper technology for developing the case hybrid mobile application for the following reasons. According to Malik (2018), the Hot Reload feature of React Native improves time efficiency during the hybrid application process. Chrzanowska (2019) also admits that thanks to React Native Hot Reload, the developers will not spare time on rebuilding the app. Gawron (2019) discusses the ease of performance in React Native by its native components, modules, and APIs.

Chrzanowska (2019) believes React-Native by its profound ready-to-apply components helps to build apps 30% faster than Swift (just for iOS). The other important feature for the author as the leading developer of the Jukola 2020 developing team was React Native open-source platform and its growing community. Rajput (2018) points out how React Native can cut the cost of the resources by the possibility of using the same code on both the Android and iOS platforms. Chrzanowska (2019) also admits React Native cost-efficiency by shrinking the size of the developing team for Android and iOS. Rajput (2018) counts React Native as a reliable and stable platform as already is utilized by Facebook and Skype. Furthermore, Malik (2018) mentions less coding is needed while applying React-Native. Accordingly, Figure 39 depicts how React Native merits outweigh its demerits.

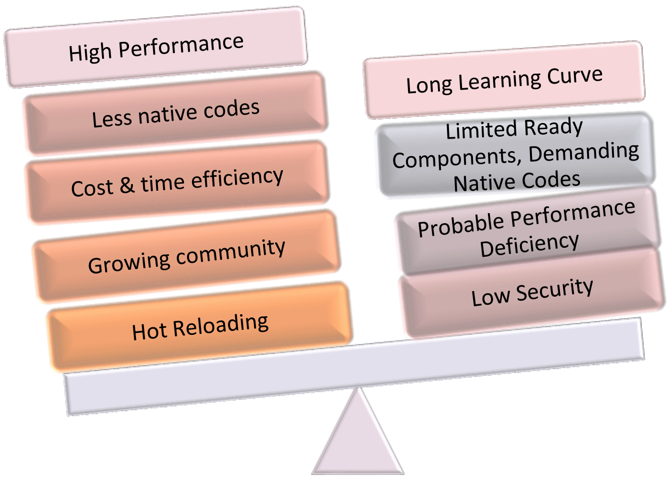


Figure 39 React Native Merits & Demerits (adapted from Malik 2018)

The author also decided about the selected platform after comparing React Native with Flutter and Xamarin as other cross-platform mobile application development tools regarding Hot Reloads, Application Performance, Libraries, tooling, third party support, UI and community. By considering many pros of React Native that shade over its cons as Table 3 shows, including limited collection of ready-made components, native code requirements, performance deficiency, compared to native apps, some components bugs, and the security ad memory management, regarding the needed features of Jukola 2020 project in defined time frame selected React Native as the most efficient platform.

Table 3 Xamarin, React Native and Flutter comparison (adapted from Hackernoon 2018)

	Xamarin	React Native	Flutter
Hot reload	No	Yes	Yes
Application performance	4/10	5/10	9/10
Libraries and tooling	6/10	7/10	5/10
Third-Party Support	6/10	9/10	5/10
UI	4/10	7/10	9/10
Community	7/10	8/10	8/10

4.1.3 Why the NoSQL database is used for storing data in Jukola 2020 application?

By considering the four types of NoSQL, including the document databases, graph stores, key-value stores, and wide column stores, NoSQL databases serve superior performance compared to relational databases. Since NoSQL databases provide a wide volume of rapidly changing structured, semi-structured, and unstructured data aligned with agile sprints, quick schema iteration, and recurring code push. NoSQL uses object-oriented programming by geographically distributed scale-out architecture. For further development of the Jukola 2020 application, the data structure should be modified by adding several new fields and tables, which is completely supported by the NoSQL database and needs less effort compared to the SQL database.

4.1.4 Why is Azure applied for hosting the web application and database?

Considering the literature review of the study, the Microsoft Azure key advantages comparing to AWS are strength Pass, flexibility, .NET compatibility, security and compliance, a variety of data centers, and cost-benefit analysis.

Strength PaaS

Many organizations, including Tietotalo, use Microsoft products, services, and frameworks, and Azure offers a platform for migrating the services to the cloud. Developers can apply the same tools and frameworks for deploying applications to the cloud. All leading development frameworks are supported by Azure PaaS, such as .NET, Java, Node.JS, and Python. It also supports several DevOps tools like Bitbucket and GitHub. Meanwhile, in 2018, Azure PaaS was the leader of Gartner Magic Quadrant, which assessed by versatility in data, application, API, and process integration for the hybrid cloud environment. Considering the lack of products and services with enterprise integration features, AWS was not involved in the Magic Quadrant (Chhibber 2018) that adds up to Microsoft Azure merits.

Flexibility

Azure also provides a wide range of services for easily scalable, cloud-native web, and mobile applications (Chhibber 2018). Flexibility is reported as the biggest highlight of Azure compared to AWS because of a strong hybrid cloud strategy.

.NET Compatibility

NET-based applications can be migrated easier by Azure Virtual Machines and Azure App Services offered by Microsoft Azure. Deploying the .NET to Azure Paas using Windows-native tools is much simpler than deploying to equivalent PaaS services in AWS (Chhibber 2018.)

Security and Compliance

A wide range of services for guaranteeing the security, privacy, compliance, and transparency is offered by Azure such as Azure Security Center, Azure Key Vault, and Azure Active Directory. Azure Security Center prevents, detects, and responds to security vulnerabilities. Comparing to Azure, AWS provides underlying security, and third-party products are needed for advanced protection capabilities and threat detection. Azure with 70 compliance offering has more compliance coverage comparing to AWS. Azure has 38 services with FedRAMP high authorization. AWS has only 21 services in the same category

(Chhibber 2018). In the Jukola 2020 application, the personal user data, including email, and phone number, is stored that needs to be kept is a secure environment.

Variety of Data Centers

Azure has more regions than other cloud service providers. There are 40 regions available currently, and ten new regions will be added to 140 countries. (Chhibber 2018.)

Cost-benefit analysis

Azure Hybrid Benefit package can help companies to save up to 40 percent of the cost of virtual machines to reuse the on-premises Windows operating system and software licenses for large-scale deployments. Azure Reserved Instances generates as much as 82 percent saving for the customers with one year or three years commitment. Azure Site Recovery makes cloud migration virtually much cheaper. Azure DevTest Labs offers quick deployment of development and test environments in a cost-effective manner. (Chhibber 2018.)

4.2 Thesis Assessment

4.2.1 Validity and Reliability

As defined by Shuttelworth (2019), validity refers to the accuracy of measurement done by a study; on the other hand, reliability represents the consistency and duplicability of the research. (Shuttleworth 2019.) Heale and Twycross (2015) define validity as the extent to which a concept is accurately measured in a quantitative study, while reliability indicates how consistent is the result of the research in the same situation on repeated occasions. They also define three types of validity, including content validity, which refers to measuring the related variable. Secondly, construct validity, which draws inferences about the test results to the study and thirdly, criterion validity, which refers to the instrument that measures the variable. (Heale & Twycross 2015.) Validity in quantitative research means appropriateness of the tools, processes, and data (Lawrence 2015).

This thesis aimed to provide a noble and model-based problem-solving approach in developing a hybrid mobile application based on the real-life project which has been created by the technical team of Tietotalo Infocenter Oy and tested by the customer thus as an evidence-based project this study considered valid. Regarding reliability, both primary and second resources have been used in the theoretical part of the to propose a practical approach based on pre studied and tested technologies. Considering the concept of time which lags behind the quick pace of technology improvements, the proposed method by this thesis might not be reliable during upcoming years. The study met its goals by answering the research questions accurately, following a constructive research methodology. Regarding reliability, both primary and second resources were applied in the theoretical part of the study to propose a practical approach based on pre-studied and tested technologies. Nevertheless, the study results might not be consistent over the time by bearing in mind the concept of time which can be overtaken by the quick pace of technology improvements, the proposed method by this thesis might not be reliable in future.

4.2.2 Scope and Limitation of the Study

The thesis presents the developing process of a hybrid mobile application using React Native platform, NoSQL DB, and Azure cloud computing service.

Regarding the time limitation, about three weeks for releasing Minimum Viable Product (MVP), the React Native platform was selected for developing the Jukola 2020 mobile application. Other hybrid platforms, such as Flutter, introduced by Google, can be used for doing similar projects.

Mongo database was chosen for two reasons, firstly for saving the developing time and secondly for decreasing the amount of effort for further development. Other NoSQL databases can be implemented to develop a similar app.

Azure is one of the most famous and significant providers of cloud computing services. Since the Tietotalo Infocenter Oy uses Azure, the developing team used this cloud for hosting. Amazon Web Services also can be used for similar hosting services.

4.3 Furthur Reseach Suggestions

The thesis presents the developing process of a hybrid mobile application, “Jukola 2020”, by applying React Native platform, NoSQL DB, and Azure cloud computing service.

Regarding the timeline of the Jukola 2020 mobile application project, which was merely three weeks for releasing MVP, the React Native platform was selected for developing the hybrid application. In a more profound study, other hybrid platforms such as Flutter can be investigated for implementing similar projects. In Jukola 2020, Mongo database, which was opted firstly to save the developing time and secondly to decrease the amount of effort for further development, is utilized. Other NoSQL databases can be implemented to develop similar applications.

Azure is one of the most famous and significant providers of cloud computing services. Since the Tietotalo Infocenter Oy uses Azure, the developing team used this cloud for hosting. Amazon Web Services also can be used for similar hosting services for related projects and applications

5 SUMMARY

Following a constructive research methodology, the study aimed to present the case hybrid mobile application “Napapiiri Jukola 2020 Teaser mobile application” developing procedure by analyzing the implemented technologies, including the React Native platform, Mongo database, and Azure cloud computing service. The thesis author, as the leading developer of the Jukola 2020 mobile application developing team of Tietotalo Infocenter Oy, found it inspiring to elaborate the integrate of technologies such as React Native, Mongo database, and Azure to develop a hybrid mobile application within a tight time framework of three weeks.

In the theoretical part of the study, both primary and secondary resources are used to review the implemented technology in the project, including React Native and its pros and cons in the first part of chapter 2. The NoSQL database including its types is analyzed in the second part of chapter 2 followed by cloud computing concept, features, and highlights in the third part of chapter 2.

In the imperial part of the study, chapter 3 is dedicated to the implantation process of the case mobile application project has been explained by taking into account the confidentiality of project implementation regarding the consulting company “Tietotalo Infocenter Oy” rights. Accordingly, the author elaborates Jukola 2020 hybrid application development process by explaining the logic behind the app technology, including React Native, Mongo DB, and Azure Cosmos DB emulator, as well as the implementation phases. The study successfully provided a concrete and noble problem-solving approach by answering the research questions, including:

1. How to develop a hybrid mobile application by React Native, NoSQL database on Azure?
2. Why selecting the React Native platform for Jukola 2020 Teaser mobile development?
3. Why is the NoSQL database used for storing data?
4. Why is Azure applied for hosting the web application and database?

The study results indicate that the React Native platform was selected for developing Jukola 2020 teaser mobile application for its high performance, less needed native code, cost and time efficiency, outstanding growing community, and hot reloading feature that outweighs Xamarin and Flutter platforms.

In the other perspective, the study presents NoSQL database superior performance, including a vast volume of structured, semi-structured, and unstructured data, aligned with agile sprints, quick schema, and recurring code push. Furthermore, Azure is applied for hosting the Web application and database for its strength Paas, flexibility and scalability, .NET compatibility, security and compliance, variety of data centers as well as cost-benefit package.

REFERENCES

Altarad, M. 2019. NoSQL Databases: The Definitive Guide. Community NoSQL Databases: The Definitive Guide. Blog. [accessed 12 August 2019]. Available at: https://pandorafms.com/blog/nosql-databases-the-definitive-guide/

AltexSoft. 2019. The Good and The Bad of Xamarin Mobile Development. Blog. [accessed 30 July 2019]. Available at: https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/

Amazon. 2019. Cloud Computing With AWS. The Leading Cloud Platform. [accessed 24 July 2019]. Available at: https://aws.amazon.com/what-is-aws/

Bala, R., Gill, B., Smith, D., Wright, D. 2019. Magic Quadrant for Cloud Infrastructure as a Service, Worldwide [accessed 22 July 2019]. Available at: https://www.gartner.com/doc/reprints?id=1-2G2O5FC&ct=150519

Bigdata Madesimple. 2014. Top five advantages and challenges of NoSQL. . [accessed 24 July 2019]. Available at: https://bigdata-madesimple.com/top-five-advantages-and-disadvantages-of-nosql/

Blair, I. 2017. How to Choose Between Native, Hybrid or Web App for Your Business. Blog. [accessed 15 July 2019]. Available at: https://buildfire.com/choose-native-hybrid-web-mobile-app/

Couchbase. 2019. Downloadable Couchbase NoSQL Server for Free. [accessed 29 September 2019]. Available at: https://info.couchbase.com/nosql_database.html?utm_source=google&utm_medium=search&utm_campaign=Nonbrand+-+Others+-+Desktop+-+GGL+-+Exact&utm_keyword=nosql%20database%20types&kpid=go_cmp-1070197905_adg-54111584284_ad-295916740722_kwd-355555881279_dev-c_ext-_prd-&gclid=CjwKCAjw0vTtBRBREiwA3URt7mNoKV880RCr5Ni1QKMkRcOgJJUpqOGUEktlVIukPj4pCmY8-8mQUhoC2pgQAvD_BwE

Chhibber, U. 2018. Why Are Organizations Choosing Azure Over AWS? [accessed 19 August 2019]. Available at: https://evessio.s3.amazonaws.com/customer/8c4659ee-526a-4e9c-89dc-f6f4c3c1a789/event/9003422d-6d7c-4754-92f3-a95c386f392d/media/media/fffeef11-profile_Navisite_Why_Orgs_choose_Azure.pdf

Chrzanowska, N. 2019. React Native Pros and Cons – Facebook’s Framework in 2019 (update) [accessed 10 September 2019]. Available at: https://www.netguru.com/blog/react-native-pros-and-cons

Churylov, M. 2019. Pros and Cons of React Native for cross-platform app development. blog. [accessed 11 October2019]. Available at: https://www.mindk.com/blog/react-native-pros-and-cons/

Czerniewski, P. 2019. Pros and cons of React Native and Native. Blog. [accessed 9 November 2019]. Available at: https://itcraftapps.com/blog/pros-and-cons-of-react-native-and-native-development/

Dyvliash, V. 2018. React Native Advantages. Blog. [accessed 20 July 2019]. Available at: https://belitsoft.com/blog/react-native-advantages

Eisenman, B. 2019. What Is React Native? [accessed 20 July 2019]. Available at: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html

Flutter.2019. Fast Development. [accessed 29 July 2019]. Available at: https://flutter.dev/

Gartner IT Glossary. 2019. Mobile Web Application. [accessed 20 July 2019]. Available at: https://www.gartner.com/it-glossary/mobile-web-applications

Gawron, K. 2019. Pros and Cons of React Native Development According to Programmers and Business Owners. [accessed 20 September 2019]. Available at: https://www.monterail.com/blog/react-native-development-pros-cons

Hackernoon. 2018. Flutter vs. React Native vs. Xamarin for Cross-Platform Development. Blog. [accessed 31 July 2019]. Available at: https://hackernoon.com/flutter-vs-react-native-vs-xamarin-for-cross-platform-development-5f92cfb178ff

Harrison, G. 2010. Full Lifecycle API Management Vendors. Blog. [accessed 21 August 2019]. Available at: https://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases/

Harrison, G. 2010. 10 Things You Should Know About NoSQL Databases. [accessed 7 October 2019]. Available at: https://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases/

Heale, R., Twycross, A. 2015. Validity and Reliability in Qualitative Studies. [accessed 10 July 2019]. Available at: https://ebn.bmj.com/content/18/3/66

Jackson, K.R., Ramakrishnan, L. , Muriki, K., Canon, S. Cholia, S., Shalf, J., et al. 2010. Performance Analysis of High-Performance Computing Applications on the Amazon Web Services Cloud. IEEE second international conference on cloud computing technology and science (CloudCom). IEEE; 2010.p. 159-68.

Jamsheer, K. 2019. SQL and NoSQL: An Overview with Advantages and Disadvantages. [accessed 7 October 2019]. Available at: https://acodez.in/sql-and-nosql-an-overview/

Jukola. 2019. General *Information. [accessed 25 July 2019]. Available at: https://www.jukola.com/en/tietoja-tapahtumasta/

Kasanen, E., Lukka, K., Siitonen, A.1993. The Constructive Approach in Management Accounting Research. Journal of Management Accounting Research, 5 (1), pp. 243-263

Lawrence, L. 2015. Validity, Reliability, and Generalizability in Qualitative Research. [accessed 9 July 2019]. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4535087/

Malik, N. 2018. React Native: Pros and Cons. [accessed 19 July 2019]. Available at: https://citrusbits.com/react-native-pros-and-cons/

Microsoft Azure. 2019. Get to Know Azure. [accessed 19 July 2019]. Available at: https://azure.microsoft.com/en-us/overview/

Microsoft Azure. 2019. Welcome to Azure Cosmos DB.[accessed 10 August.2019]. Available at: https://docs.microsoft.com/en-us/azure/cosmos-db/introduction

MongoDB. 2019. NoSQL Databases Explained. [accessed 19 July 2019]. Available at: https://www.mongodb.com/nosql-explained

MongoDB. 2019. The Database for Modern Applications. [accessed 28 July 2019]. Available at: https://www.mongodb.cohttps://www.mongodb.com/m/

NoSQL. 2019. List of NoSQL databases. [accessed 12 July 2019]. Available at: http://nosql-database.org/

Polidea. 2018. React Native vs. Native App Development- Pros and Cons for Business. [accessed 28 July 2019]. Available at: https://medium.com/@polidea/react-native-vs-native-app-development-pros-and-cons-for-business-438e09b7e205

Penchal Reddy, M. 2019. Overview of Azure Cosmos DB. [accessed 12 July 2019]. Available at: https://www.red-gate.com/simple-talk/cloud/cloud-data/overview-of-azure-cosmos-db/

Pruulmann, S. 2019. Native vs. React Native: which is right for you? Blog. [accessed 22 July 2019]. Available at: https://blog.mooncascade.com/native-vs-react-native-which-is-right-for-you/?gclid=Cj0KCQjw9fntBRCGARIsAGjFq5E-At2bU2itXyVIQS1qkVyUKGQHiQ35KvsfJHFRfshwk6Ja4hgiGoAaAj4zEALw_wcB

Rajput, M. 2018. The Advantages and Disadvantages of Using React Native as Cross-Platform App Development [accessed 22 July 2019]. Available at: https://www.netguru.com/blog/react-native-pros-and-cons

Ratner, J. 2019. The Pros and Cons of React Native App Development. Blog. [accessed 6 October 2019]. Available at: https://www.verypossible.com/blog/pros-cons-react-native-app-development

Ren, K., Wang, C., Wang, Q., 2012. Security Challenges for the Public Cloud. IEEE Internet Computing. Pp.69-73, IEEE Computer Society.

Regoli, N. 2017. 7 Pros and Cons of NoSQL. [accessed 6 October 2019]. Available at: https://greengarageblog.org/7-pros-and-cons-of-nosql

Rouse, M. 2011. Hybrid Application. [accessed 7 July 2019]. Available at: https://searchsoftwarequality.techtarget.com/definition/hybrid-application-hybrid-app

Rouse, M. 2018. Microsoft Azure (Windows Azure). [accessed 21 July 2019]. Available at: https://searchcloudcomputing.techtarget.com/definition/Windows-Azure

Shanahan, H.P., Owen, A.M. and Harrison, A.P. 2014. Bioinformatics on the Cloud Computing Platform Azure. PLoS ONE 9 (7): e102642. [accessed 21 Oct 2019]. Available at: https://doi.org/10.1371/journal.pone.0102642

Stonebranch. 2019. Cloud Orchstration. What are Typical Use for Cloud Workload Automation? [accessed 21 Oct 2019]. Available at: https://www.stonebranch.com/solutions/cloud-automation/?gclid=CjwKCAiA5JnuBRA-EiwA-0ggPeyDgrVQq0u9uGd2jcQ5OikkJEZAZ7wqkYM89dT0upufqfKCSbWcrRoC9DQQAvD_BwE

Techopedia. 2019. Native Mobile Application. [accessed 9 July 2019]. Available at: https://www.techopedia.com/definition/27568/native-mobile-app

Tietotalo. 2019. [accessed 9 July 2019]. Available at: https://www.tietotalo.fi/en

Tozzi, C.2016. The Limitations of NoSQL Database Storage: Why NoSQL's Not Perfect. [accessed 6 October 2019]. Available at: https://www.channelfutures.com/cloud-2/the-limitations-of-nosql-database-storage-why-nosqls-not-perfect

Visual Studio Code. 2019. Code 'Editing Redefined. [accessed 2 August.2019]. Available at: https://code.visualstudio.com/

Vries, RD. 2019. Getting Started with Cross Application Development in 2019. [accessed 30 July 2019]. Available at: https://hackernoon.com/getting-started-with-cross-platform-app-development-in-2019-dd2bf7f6161b

Wadhwa, D. 2019. Should I Go with React Native? Blog. [accessed 10 July 2019]. Available at: https://www.ryadel.com/en/react-native-pro-cons-review-guide-tutorial/

Wodehouse, C. 2019. 7 Things to Know Before Choosing an App Maker. [accessed 10 July 2019]. Available at: https://www.upwork.com/hiring/mobile/7-things-to-know-before-choosing-an-app-maker/

Yadav, K. 2018. What is AWS, and What Can You Do With It? Blog. [accessed 17 August 2019]. Available at: https://blog.usejournal.com/what-is-aws-and-what-can-you-do-with-it-395b585b03c