Backend as a service in web development

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor thesis

Degree Programme in
Computer Applications
spring, 2023

Samu Uunonen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Computer Applications		Tiivistelmä
Tekijä	Samu Uunonen	Vuosi 2023
Työn nimi	Backend as a Service in web development	
Ohjaaja	Lasse Seppänen	

TIIVISTELMÄ

Opinnäytetyö esittelee Backend palveluiden mahdollisuuksia web kehityksessä. Tutkimuksen tavoitteena oli selvittää palveluntarjoajien eroavaisuuksia yrityksen ja ohjelmistokehittäjän näkökulmista. Opinnäytetyössä kehitettiin demo sovellusta missä backendinä käytettiin Firebase ja Amplify palveluita. Työ on tehty tekijän omasta kiinnostuksesta aiheeseen ja ulkoista toimeksiantajaa ei ole.

Teoreettisessa osuudessa määritellään työn kannalta keskeisiä käsitteitä. Tietopohja koostuu tietoturvasta, web kehityksestä ja pilvipalveluista. Tutkimusaineisto kerättiin kehittämällä demo sovellus ja tutkimalla palveluntarjoajien dokumentaatioita.

Tutkimuksessa havaittiin, että Backend palveluiden välillä on eroavaisuuksia. Erot ovat jopa niin suuria, että yrityksiä suositellaan tutkimaan tarkasti mikä palvelu soveltuu parhaiten omaan käyttötarkoitukseen.

Avainsanat	Backend, Pilvipalvelut, Serverless
Sivut	36 sivua ja liitteitä 1 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Computer Applications		Abstract
Author	Samu Uunonen	Year 2023
Subject	Backend as a Service in web development	
Supervisors	Lasse Seppänen	

ABSTRACT

The thesis presents the Backend service's possibilities to the web development. The goal of the research was to investigate differences between the service providers in business and developer's perspective. Demo application was developed where the Firebase and Amplify services were used as a backend. The thesis has been made by the author's personal interest on the topic, without external principle.

The theoretical part of the thesis describes the key terms that contain privacy, web development and cloud computing. Research material was collected by developing the demo application and by investigating the documentation of the services.

The research results showed that the services have big differences. It is recommended to investigate precisely what service is suitable for that specific purpose.

Keywords	Backend, Cloud, Serverless
Pages	36 pages and appendices 1 pages

Glossary

API	Application programming interface
AWS	Amazon Web Services
BaaS	Backend as a service
Backend	Part of the website or application that user does not see.
CLI	Command Line Interface
FaaS	Functions as a Service
Frontend	Part of the website or application that user interacts with.0
JSON	JavaScript Object Notation
PaaS	Platform as a Service
SDK	Software Development Kit
UI	User interface

Contents

Figures and Program codes

Annexes

1 Introduction

Developing a web application requires lots of time and work. Every entrepaneur know that these two things are not infinite. Web development project consists of backend and frontend development. The workload is not always divided half between the backend and frontend, the proportion of the work backend requires varies, but it is a big part of the development work. By outsourcing parts of the work, the amount of the developers in the team can be decreased or same team can get more features done with the same resources. Nowadays there are multiple ways to outsource the development work, from offshore developers to services that handles one or more tasks from application. Software developers are an expensive and difficult resource to find, because the world’s demand exceeds the supply. Every company in the world are trying to solve this problem.

Cloud computing has reduced an infrastructure management work from the development teams. Backend as a service is a cloud computing service. It is one way to outsource part of the backend development. Other company has built a universal backend product that can be integrated into any application. Backend services usually has managed database and authentication, which are essential part of almost every web application. The thesis will lead reader into world of serverless computing with Firebase and Amplify backend services. Both services are inspected through the demo application and their documentations.

- What are the things to consider when setting up a software business on a backend service?
- What are the differences between BaaS providers from a business and developer perspective?

2 Finnish ICT job market

Numbers differs based on the source, but it seems clear that there is a massive need for software developers in Finland. Software development requires specific skills and lack of developers is a dangerous situation. The country cannot digitalize as much it wants and cannot attract foreign investments to Finland because lack of the skillful workforce. According to Finnish Ministry of Economic Affairs and Employment’s Työvoimatiekartat report, software designers were fourth most wanted high paying employees in Finland. The estimate is that 426 software designer job openings are unfulfilled in 2022, and it does not include software architects or software developers. In Finnish Business and Policy Forum EVA’s analysis Ahopelto (2018) states that the software field has job openings immediately for 7000-9000 new employees in Finland, and by 2024 it could increase to 40,000. In 2021 software companies employed 56,000 people. One of the suggested solutions to fix the shortage is to increase the starting places in the universities for ICT sector, which has been decreasing in the 2010s. However, the analysis claims that even doubling the number of students in the universities will not be enough to fill the demand. Other solutions presented in EVA’s analysis are to increase amount of labor immigration and to persuade more women into the ICT sector. Attracting the coders from the other countries is not an easy job because many other countries are struggling with the same issue. Finnish companies lack the skills to find employees outside of Finland. (Ahopelto, 2018, pp. 1–10; Larja & Peltonen, 2023, pp. 16–17)

3 Privacy

Data privacy is a relatively new concept as it started to exist after the internet started to spread exponentially. Every action that is made with digital devices leaves a digital trail. These trails are used to profile users and convert it into a source of revenue. Outside of the internet, private information is always shared in specific contexts. On the internet people usually understand that information shared with someone else is out of their personal control. Users cannot know how their data will be used. (Sharma, 2020, pp. 1–4)

3.1 General Data Protection Regulation

General Data Protection Regulation or shorter GDPR sets requirements on how to collect, store and manage personal data. It applies to applications that process data in the European Union (EU), even when the actual processing takes place outside of the EU. Data is considered personal data when it includes name, address, income, ip address or any other data where a person can be identified with. The process contains two main profiles, the data controller and -processor. The controller decides why and how the data is processed, usually that is the owner of the product. The processor holds and processes the data on behalf of a controller. (Your Europe, n.d.)

3.2 Privacy frameworks

When reading the Privacy clauses from the cloud services, the reader often runs into the terms Privacy Shield Framework and Standard Contractual Clauses. The Privacy Shield Framework was a data privacy framework for transferring personal data from the European Union and Switzerland to the United States. On 16 July 2020, the Court of Justice of the European Union issued that the Privacy Shield Framework is not a valid mechanism to follow when transferring personal data from the European Union to the United States. The Standard Contractual Clauses are agreements between the EU and a party who wants to transfer data from the EU to non-EU countries. It replaced The Privacy Shield Framework. The level of personal data protection must be checked on a case-by-case basis to make sure it meets the EU requirements. (Privacy Shield, n.d; Tietosuojavaltuutetun toimisto, n.d.)

4 Web Development

Web development is a high-level term that includes many smaller parts that are introduced in this chapter.

4.1 Frontend development

HTML, CSS, and JavaScript are the foundation of the frontend development. HTML stands for Hypertext Markup Language and is the foundation of the website. HTML is the structure of the content and presents the content to the users. The content between the HTML tags has a specific meaning. CSS adds the style, and it stands for Cascading Style Sheets. It works by referring to the HTML tags of the site. (Wellens, 2015, pp. 3-10)

JavaScript is a programming language to add the functionalities to the site. Today it can be used to build both frontend and backend of the application. With the Webpack solution for managing dependencies in JavaScript applications, frameworks such as React have changed the way web applications are developed. (Brown, 2019, p. 2)

React is a JavaScript library designed to build user interfaces. It is based on reusable components that can present mutable data. Every element of the application is a component, even the application itself. React does not force developers to use any specific presentational pattern. In Model-View-Controller or Model-View-View Model pattern context, it is implementing the View part. (Chiarelli, 2018, pp. 6–7)

4.2 Backend development

Backend development or server-side development is the part of the application that user cannot see. It consists of everything that happens on the server, from API designing to data modeling. The main job of API is to communicate with the frontend and fetch the requested data, modify it, and send it to the frontend. Backend development can be done with multiple programming languages, such as Java, Python, PHP, and JavaScript. Database and server management are key parts of any application’s backend. Database is used to store data and it has two types, relational and non-relational. Non-relational database stores data in flexible formats, while in relational database data is stored in tables. Web servers store the code and process the application. Servers handle communication between the user and application with HTTP, which stands for HyperText Transform Protocol. (Gallardo, 2023)

4.3 Full stack development

Full stack refers to an end-to-end application that includes both frontend and backend. Today full stack web applications can be pure JavaScript applications. One example of a many full development stacks is MERN, that is a shortener from MongoDB, Express, React and Node. It is a combination of four JavaScript based technologies. MongoDB is a NoSQL database. Express and Node are backend technologies and frontend is created with React. (Komanduri & Hoque, 2018, pp. 6-9).

Communication between the different services can be done by utilizing SDKs and APIs. SDK stands for Software Development Kit. It is a set software tools for a specific platform. By using SDK developers can create fast integrations between the services. Usually, an SDK includes at least one API, that stands for Application Programming Interface. API facilitates communication between two platforms. By using APIs developers can easily connect the application with third-party applications. (IBM, 2021)

5 Cloud computing

Cloud computing is a cost effective and scalable way to use computing resources. Cloud computing enables users to use computing resources without owning physical hardware. Instead of building on-site data centers and servers, the cloud computing provider takes care of the hardware, and the user only buys the needed resources monthly or as pay-as-you-go pricing model. Cloud computing may improve security and reliability of the services as computing resources are outsourced to professionals who manage the cloud computing services. (Microsoft Azure, n.d.)

5.1 Cloud Service Models

Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are the major cloud service models. All cloud service models provide cloud computing services. IaaS contains only the hardware resources. IaaS is a cost-effective option for purchasing and maintaining hardware. Extended to IaaS, PaaS contains operating systems, databases, and development tools. It is possible to run and develop scalable applications in PaaS infrastructure. SaaS is a ready to use software product which has all features built. (IBM, n.d; Microsoft Azure, n.d-a)

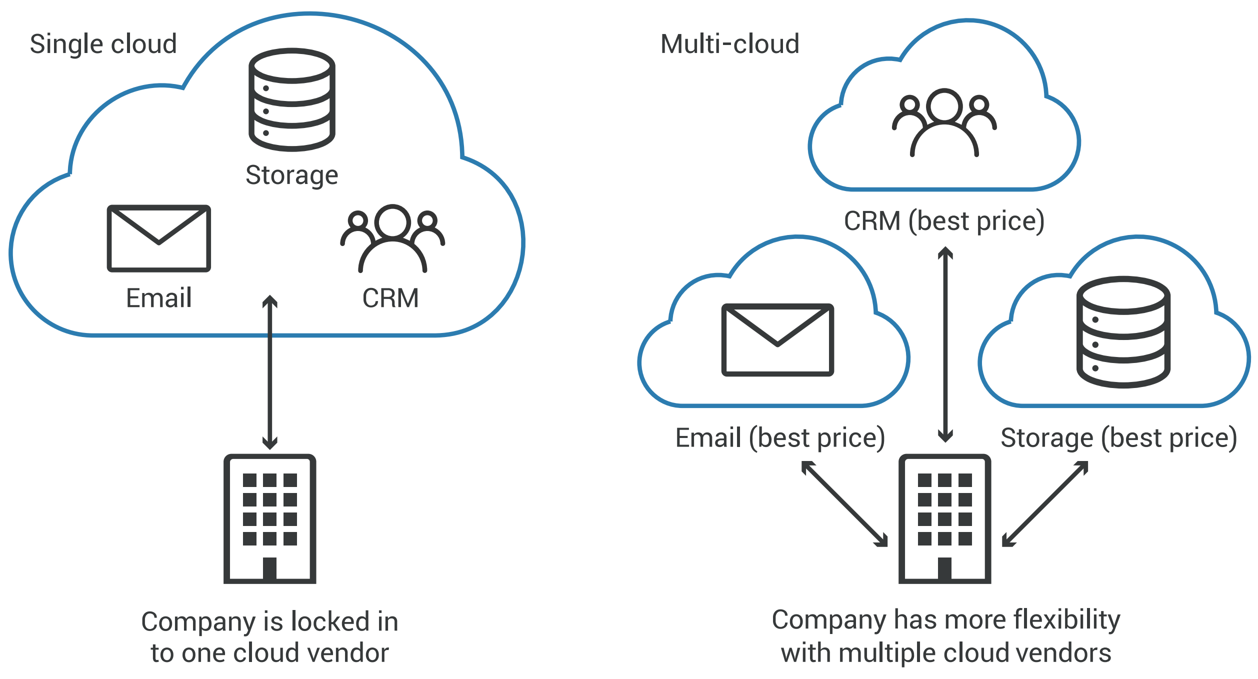
5.2 Serverless computing

The old applications are usually built in a monolithic way which means that the whole application is a single unit. Microservice applications are divided into smaller units, where each unit performs a single service. Serverless application is built into multiple units that performs only one specific task. According to Chowhan, the name Serverless is misleading because serverless applications still require a server to run. Cloud infrastructure is extended with the abstraction that gives developers capability to deploy and run applications without managing servers. Serverless infrastructure can be defined as an outsourced server management. Serverless applications usually consist of multiple functions that work together with other services. Architecture that is combined from multiple functions is called FaaS that is shortened from Functions as a Service. FaaS is an event driven architecture where functions are triggered by events. Trigger events can be for example API request, scheduled event, and database event. (Chowhan, 2018, p. 8-11)

The benefits of serverless computing are reduced costs, faster development, and scalability. According to Chowhan, serverless applications are cheaper to operate because of the shared infrastructure. Computing is shared with the cloud provider’s other customers. Every customer is billed by the running time, which can be charged in as low as millisecond intervals. Applications that utilize serverless architecture run in a fully managed infrastructure with automatic provisioning and scaling by the cloud provider. Developers focus only on writing the code. (Chowhan, 2018, pp-17-20)

The drawbacks of serverless computing are the lack of infrastructure control, cold start, and vendor lock-in. As mentioned, the serverless application runs in fully managed infrastructure where developers do not have access. Cold start means the function takes longer to respond to an event if it has been inactive for a while. Vendor lock-in means that changing to a different vendor is so expensive, that business is locked with a certain vendor. As the Chowhan (2018, p. 23) summarized: “One of the biggest fears with serverless applications concerns vendor lock-in.” Even though the Figure 1 is not specific to serverless but to the whole cloud computing, companies who use more than one cloud solution have more flexibility with the services. Businesses who rely on multiple vendors can transform their solutions if circumstances change. Businesses who are relying on one vendor are not as flexible. The risks of vendor lock-in are that quality-of-service declines, vendor changes the product, increases prices, or goes out of business. (Chowhan, 2018, pp. 22–25; Cloudflare, n.d-a.)

Figure 1. Vendor lock-in explained in cloud computing. (Cloudflare, n.d-b.)


Single cloud
Storage
Email
CRM
Company is locked in to one cloud vendor
Multi-cloud
CRM (best price)
Email (best price)
Storage (best price)
Company has more flexibility with multiple cloud vendors

5.3 Cloud security

Security in cloud computing consists of data integrity and privacy in an end-to-end perspective. Security containers are one approach to addressing security concerns in cloud environments. Security container is a virtual environment that is created through the segmentation of the network in a cloud computing environment. This environment allows elements with similar security characteristics to share a common boundary of trust. This means that the computing elements within the container are trusted to communicate with each other and share data, while outsiders are prevented from accessing this data. The use of security containers enables organizations to secure their data by limiting access to authorized users and restricting unauthorized access to sensitive information. (Ruparelia, 2016, pp. 99-107)

According to Ruparelia (2016), three security procedures are typically used to secure access into a security container: identification, authentication, and authorization. Identification is the process of identifying a user or computing element attempting to access the security container. Authentication is the process of verifying the identity of the user or computing element. This is typically done using a username and password, biometrics, or some other form of authentication factor. Authorization is the process of granting or denying access to the security container based on the identity and permissions of the user or computing element. (Ruparelia, 2016, pp. 105-107)

Network segmentation is the process of dividing a network into smaller segments. Each segment has own set of security policies and controls. Segmentation is generally used to create a security container, which contains the elements that share the same security attributes. The network layer is the most common layer where segmentation occurs because it is the mode of transport for information. Virtual networks are created within the physical network and each container uses the virtual networks inside it to prevent outsiders from accessing its internal data. Data integrity is protected by encrypting the network channel that is used to transport the data. Encryption can be done with the use of certificates that allow communication over a trusted channel. Encrypted channels are known as a Secure Socket Layer (SSL). SSL is using a handshake protocol that uses public and private keys to encrypt the channel. Receiving device decrypts the channel that was encrypted by the sending device with encryption keys. (Ruparelia, 2016, pp. 105–112)

6 Backend as a Service

BaaS stands for Backend as a Service, and it is an extended cloud service model from PaaS with pre-built backend features for mobile and web applications. Backend features are for example database, user management and file storage. BaaS is like a serverless architecture, where infrastructure management is taken care of by the service provider. When the development team decides to use BaaS in the project, they decide to outsource the backend development to the service provider. This increases the speed of the development work because developers can focus on building the front-end features. (Dudjak & Martinović, 2020, p. 208)

Backend service providers have created documented APIs and SDKs that enable fast implementation of ready-made code functions. APIs and SDKs are used to provide interaction between the backend and frontend. Essentially BaaS is a third-party software product that developers can buy to utilize it in their own application. Usually, the service has a free tier that makes it free to start, and payments start occurring when the application usage increases. Usually pricing is pay-as-you-go based on the features that are used. The application’s time to market can be reduced by using backend service. Time to market is reduced because most of the backend has already been coded with someone else and the infrastructure is taken care by the BaaS company. Especially when the SDKs have been built for the development platform that the team is using, for example React. Security and performance have been taken care of by the professionals because BaaS company sells reliability, performance, and security. (Arkhipov, 2023)

6.1 Authentication

In computer world authentication identifies user’s identity and control access to application or resource. Popular method to identify user is a combination of password and email, where both act as a credentials. This is called a single-factor authentication, and it can be extended into multi-factor authentication when an additional authentication factor is required. The additional factor for the authentication could be a pin code that is shared between the system and user. Passwordless authentication removes the need of passwords. It can work by sending a pin code via email or by utilising facial recognition. (Auth0, n.d-a.)

OAuth 2.0 stands for open authorization and is one of the popular authorization protocols on the web. It is designed to grant access to a set of resources, for example APIs or user data. OAuth 2.0 is based on Access Tokens which data contains authorization to access resources. JSON stands for JavaScript Object Notation, and it is often used format in the tokens. JSON Web Token is a digitally signed JSON object (Jwt, n.d.). SAML (Security Assertion Markup Language) is designed to authenticate a user. It is popular among enterprises and governments, because of its cross-domain single sign-on compatibility. OpenID Connect (OIDC) is a protocol that utilizes mechanisms of OAuth 2.0. While OAuth 2.0 is an authorization protocol, OIDC is an identity authentication protocol. In addition to user's requests, contact information or other data can also be shared on requests. OIDC is based on JSON Web Tokens and can be used for single sign-on. It is a simpler alternative to SAML protocol and is suitable for all kinds of applications. (Auth0, n.d-b; Auth0, n.d-c; Auth0, n.d-d)

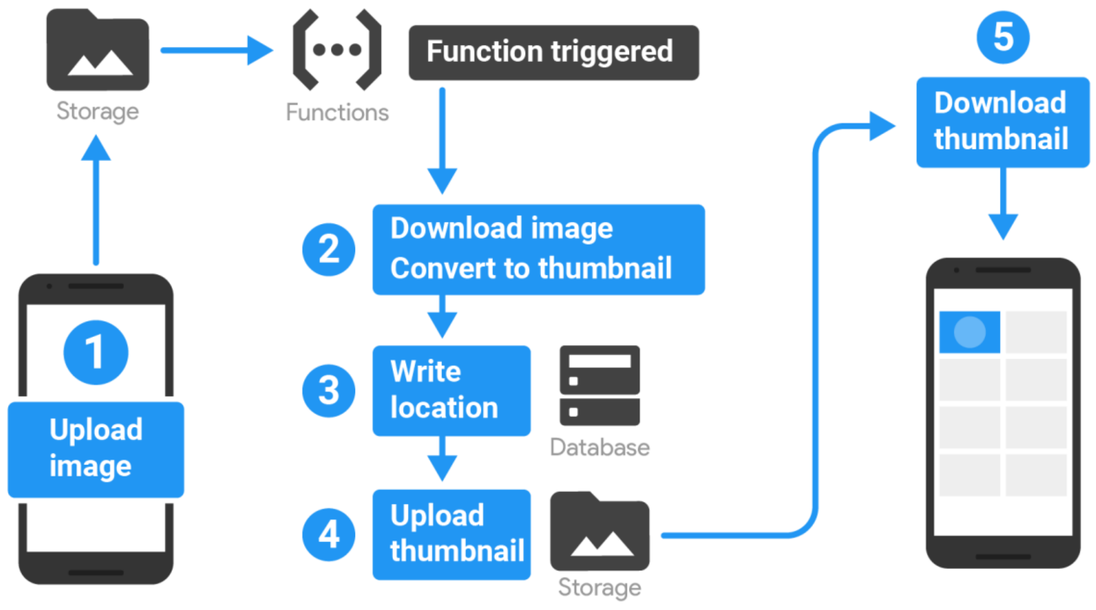
6.2 Database

Database is the part of an application where information is stored. In BaaS, the service provider takes care of the database management and developers can focus on stored data. Database can be Relational, Object-oriented, Distributed, Data warehouse, NoSQL, Graph, Open source, Cloud, Multimodel, Document/JSON or Self-driving. (Oracle, n.d.)

6.3 Serverless functions

Serverless functions allow developers to develop backend functionality without maintaining a server. It requires low maintenance and automatically scales up computing resources when needed. Functions can be triggered with HTTP requests or database updates. Custom backend logic may be required when the Backend Service's features are not enough for the use case, or the task is too intensive to be handled in the frontend. Performance is a key reason to handle features in server instead of the user's device. Figure 2 has an example of an intensive task, where an image file is resized in the cloud instead of a device. Another reason could be security, in some cases it is safer to run code on the server side instead of the client side. The application logic never leaves the server and cannot be accessed from the client side. (Firebase, n.d-f; Amazon Web Services, Inc, n.d-i)

Figure 2. Execute intensive tasks in the cloud instead of a user's device. (Firebase, n.d-f.)



6.4 Hosting

Web hosting is a service that essentially is a server where a website or application is stored. Server transfers all necessary files to the client's browser to make content visible for the user. Maintenance and configuration of the server is handled by the hosting service provider. Service can include additional features such as security and backup. Hosting should not be mixed up with domain that is a unique address for the web application. Custom domain can be set in hosting service. (Amazon Web Services, Inc., n.d-g.)

6.5 Storage

Storage is a part of computing that store data and files. Cloud storages has three main types: object stroage, file storage and block storage. Object storage is good for large amount of unstructured data. Data is stored to bucket instead of organized files and folders. Applications ofte use Object storage. File storage stores data in hierarchical folder and file format. It is common in Network File Systems. Block storage store data in blocks with unique identifier and it is used in databases. (Amazon Web Services, Inc., n.d-g.)

7 Development work

The purpose of the development work is to use backend services in practical and experimental ways. The goal is to find out what kind of differences platforms have in developers or business point of view. Part of the work is to get familiar with service by studying its documentation. The other part is to try key features of the service in the demo application. The key features are Authentication, Database and Storage. In the development work, following aspects are inspected:

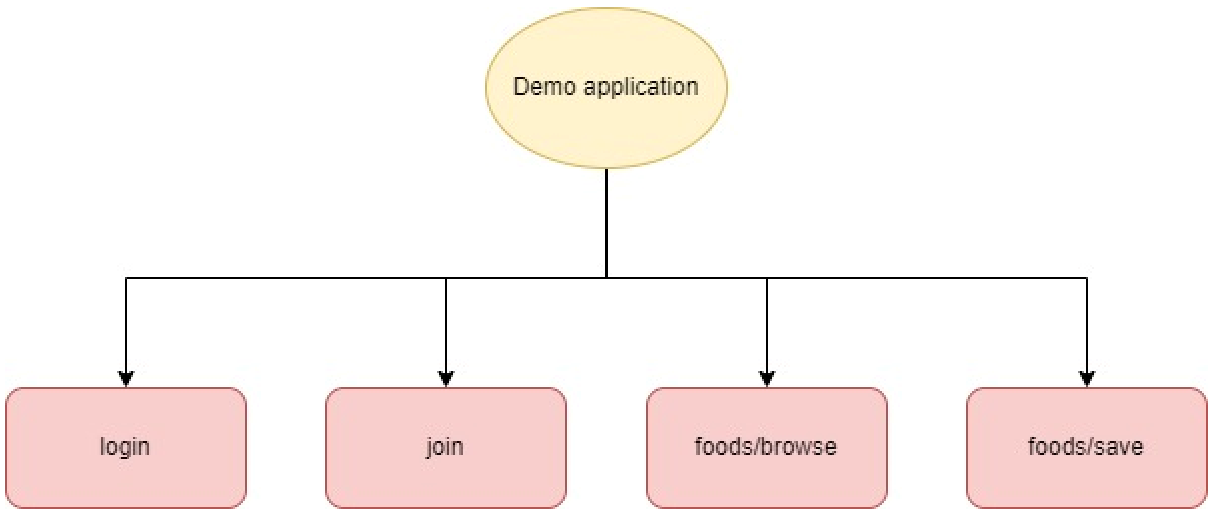
- What programming languages are supported by SDKs?
- Documentation quality, does it contain code examples?
- What kind of development environment is available?
- Learning curve and complexity of the service.
- Dashboard quality, is it possible to configure services without coding skills?
- State of privacy from EU user point of view.

According to Zhang (2023) Amazon Web Services and Google Cloud Platform are in the top 10 largest cloud service providers (Zhang, 2023). That is the reason why Google owned Firebase and AWS owned Amplify were selected for this thesis.

7.1 Demo application

Demo application is a small web application with user authentication. In the application users can save and view data and image files. Authentication, Database, Storage and Hosting are the features that are used from the backend service. The frontend part of the application is coded with React and Bootstrap UI components, but this work is not focused on the frontend. As can be seen from Figure 3, the application has 4 pages: login, join, browse, and save. Login page contains the login form, join page contains the create account form. Browse foods page contains all items saved by the user and is used to browse and access their saved items. Save food page contains a form to save new items, where user adds food information which are name, description, and image file.

Figure 3. Demo application page structure.



7.2 Version control

The source code is saved into GitHub repository. All code is in one repository, and each implementation is separated into its own branch. Master branch contains the base structure of the application where backend will be implemented.

8 Firebase

Firebase is a development platform where developers can build mobile and web applications. It is backed by Google Cloud. Firebase includes fully managed backend infrastructure, monitoring, and user engagement products. (Firebase, n.d-i.)

To create a Firebase project, Google account is required because Firebase is based on Google Cloud services. Project is created in the Firebase console. To be able to use Firebase features in the application, an app must be created within the project in the console for a specific platform. The project can contain multiple apps, which makes it possible to use the same backend with multiple platform applications. Available app platforms are iOS, Android, Web, Unity, and Flutter. All features are disabled by default and must be activated before access. Project is in free tier, until it is manually changed into pay as you go Spark plan.

8.1 Configuring the application and development environment

First the Firebase SDK needs to be installed into the app. Most of the SDKs are available for iOS+, Android, Flutter, C++, Unity, and JavaScript. Firebase features are initialized in the firebase-config file, which is a normal JavaScript file. The configuration details can be copied from the Firebase console after the web application is created, or later from the project settings. In the Program code 1, configuration details are imported as environment variables from the .env file. In the code the following features are initialized: firebase app, authentication, database, storage, and local emulators. At the end of the code, the emulators are initialized only in the development environment. When deployed to the production environment, emulator code will not run. Authentication, Firestore database and Storage are initialized in the firebase-config file. Every Firebase service that will be used in the application has to be initialized in this file. (Program code 1)

Program code 1. Firebase-config.js

```
import { initializeApp } from "firebase/app";
import { getAuth, connectAuthEmulator } from "firebase/auth";
import { connectFirestoreEmulator, getFirestore } from
"firebase/firestore";
import { connectStorageEmulator, getStorage } from "firebase/storage";

const firebaseConfig = {
  apiKey: import.meta.env.VITE_FIREBASE_API_KEY,
  authDomain: import.meta.env.VITE_FIREBASE_AUTH_DOMAIN,
  projectId: import.meta.env.VITE_FIREBASE_PROJECT_ID,
  storageBucket: import.meta.env.VITE_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: import.meta.env.VITE_FIREBASE_MESSAGE_SENDER_ID,
  appId: import.meta.env.VITE_FIREBASE_APP_ID,
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const db = getFirestore(app);
export const storage = getStorage(app);

// Connect to emulators
if (import.meta.env.DEV) {
  connectFirestoreEmulator(db, "localhost", 8080);
  connectStorageEmulator(storage, "localhost", 9199);
  connectAuthEmulator(auth, "http://localhost:9099");
}
```

8.2 Authentication

Firebase Authentication utilizes standard protocols as OAuth 2.0 and OpenID Connect. It provides SDKs with methods to create and manage users with email and password. It has built-in features to send reset and verify emails. Authentication with social sign-in is provided with Google, Apple, Facebook, Twitter and GitHub accounts. Phone number authentications allows users to identify without password by sending a one time code via SMS message. Anonymous auth allows users to sign-in via temporary anonymous account, and if the user later wants to sign up, temporary account can be upgraded to regular account. Firebase Authentication can be upgraded with Identity Platform that allows multi-factor authentication and support for SAML. FirebaseUI Auth is a drop-in authentication solution with a full pre-built authentication flow. It includes all sign-in methods mentioned above. Pre-built user interface can be customized to match application’s style. (Firebase, n.d-a)

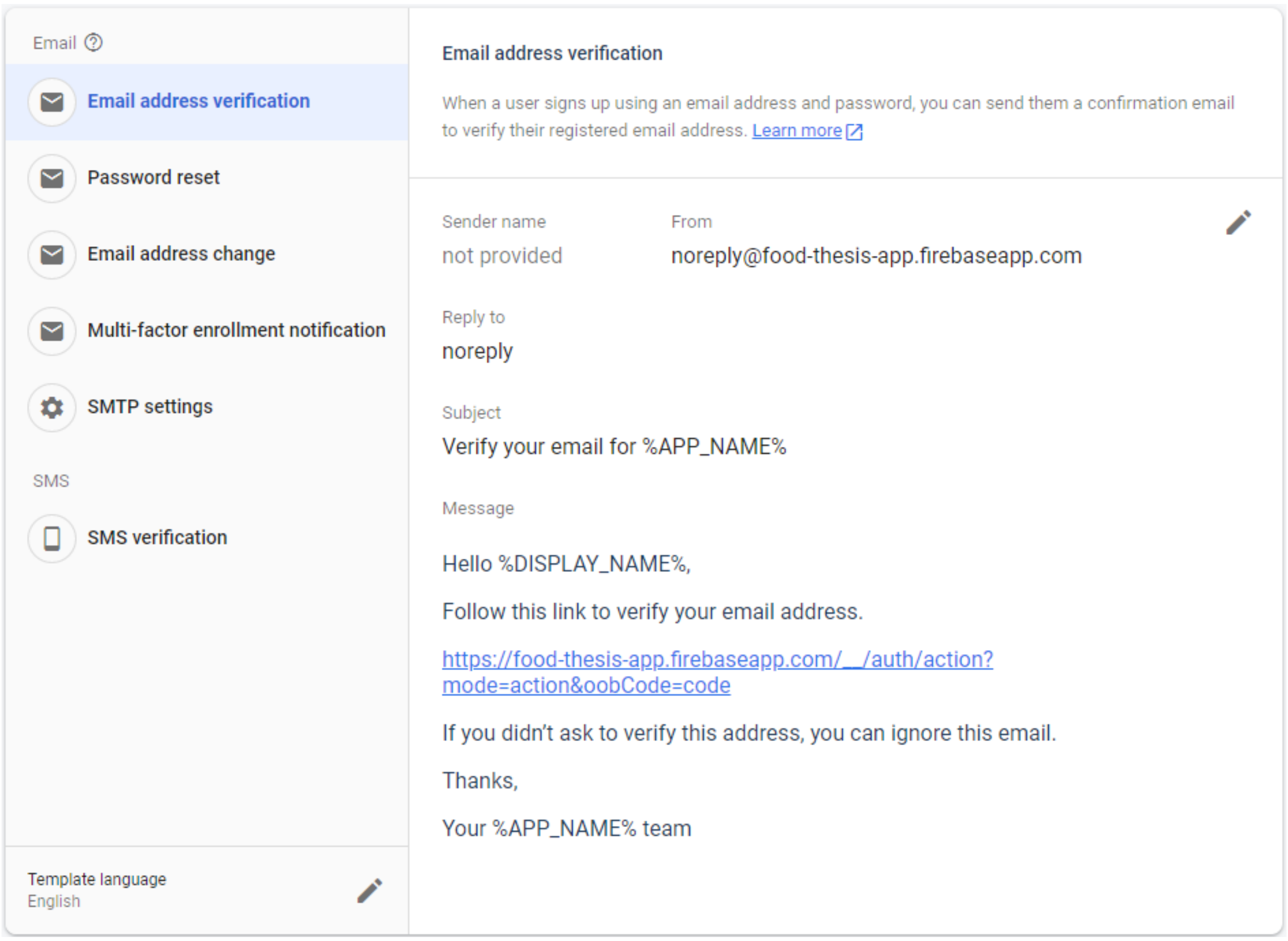
Sign in providers are selected in the Firebase console by enabling them from the list of providers. After the methods are selected, the code snippets for user management can be copied and pasted from the Firebase documents. Documents offers ready to use code snippets for all authentication functionalities. All that remains to do is to insert the code into the application and define what to do with the response. (Program code 2)

Program code 2. Create user function code for Firebase.

```
const createUser = async (credentials) => {
  createUserWithEmailAndPassword(
    auth,
    credentials.email,
    credentials.password
  )
    .then((userCredential) => {
      setUser(userCredential.user.uid);
      localStorage.setItem("user", userCredential.user.uid);
    })
    .catch((error) => {
      console.error(error.message);
    });
};
```

Email templates can be set from the console. Firebase Authentication has templates for email verification, password reset, email change etc. Content of templates cannot be modified, but language can be selected from multiple langauges including Finnish. (Figure 4)

Figure 4. Email templates settings in Firebase console.



Email address verification

When a user signs up using an email address and password, you can send them a confirmation email to verify their registered email address. Learn more

Sender name: not provided

From: noreply@food-thesis-app.firebaseapp.com

Reply to: noreply

Subject: Verify your email for %APP_NAME%

Message

Hello %DISPLAY_NAME%,

Follow this link to verify your email address.

https://food-thesis-app.firebaseapp.com/__/auth/action?mode=action&oobCode=code

If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your %APP_NAME% team

8.3 Database

Firebase offers two NoSQL databases, Realtime Database and Cloud Firestore. Realtime Database stores data in a JSON tree format. Firestore stores data as collections of documents which offers a possibility to create more complex queries. Support for offline persistence is a built-in feature in both databases. Offline support caches the data from application if the device is offline and when the device is online, data will be synchronized to database. Databases are secured with the Security Rules, which manage access to data. Security Rules are integrated with Firebase Authentication. Rules can be set from the project's firestore.rules file or from the Firebase console. (Firebase, n.d-d; Firebase n.d-e.)

Firestore does not require a data modelling, but it can be done with the security rules. The rules file has a specific language that resembles JavaScript. The Program code 3 contains a function that validates the data types before it is inserted to the database. Function is used within the write operations. Both read and write operations in foods collection require that user is authorized.

Program code 3. Firestore rules file.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    function isAuthorized(request, resource) {
       return request.auth.uid == resource.data.uid ||
       request.auth.uid == request.resource.data.uid;
    }

    match /foods/{document} {
      function validateFoods(data) {
        return data.name is string &&
        data.description is string &&
        data.photo is string &&
        data.uid is string;
      }

      allow read: if isAuthorized(request, resource);
      allow write: if isAuthorized(request, resource) &&
      validateFoods(request.resource.data)
    }
  }
}
```

When initializing the database in the Firebase console, the region must be selected. The region selection is permanent and cannot be changed. The Demo application uses a Firestore database, and the region is Europe-west, that is a multi-region option. The region will be the same for both Database and Storage and the selection is permanent for the project. Using the Firestore database is straightforward. In Program code 4, data is written into the foods collection of the database. If the collection does not exist, it is created automatically. Response from the successful write operation contains the document identifier that is automatically created if not defined.

Program code 4. Insert data into Firestore database.

```
const saveFood = async (food) => {
  const docRef = await addDoc(collection(db, "foods"), food);
  food.id = docRef.id;
  setFoods([...foods, food]);
};
```

Response from a database read is a snapshot of the results. Snapshot is an array that contains documents from the query. It is possible to chain multiple filters in one query. Filter can be based on property value, like in the Program code 5or query based on array value. Addition to filters, result can be sorted by a property value. Results can be paginated with the query cursors. Firebase has made it easy for developers by providing code snippets for all Firestore features in documentation.

Program code 5. Read data from Firestore database.

```
const loadFoods = useCallback(async () => {
  if (user) {
    const q = query(collection(db, "foods"), where("uid", "==", user));
    const querySnapshot = await getDocs(q);
    const loadedFoods = [];
    querySnapshot.forEach((doc) => {
      loadedFoods.push({ ...doc.data(), id: doc.id });
    });
    setFoods(loadedFoods);
  }
}, [user]);
```

8.4 Storage

Firebase Storage is object storage and can be used to store photos, videos, and other files. Files can be uploaded, downloaded, and removed with the SDK. Firebase Authentication is integrated with the Storage. Similar security rules that are used for the Firestore are used to manage access to resources. (Firebase, n.d-g.)

The uploading process can be followed in the application, it allows a functionality to display a progress bar while the uploading process is in progress. When the uploading task is completed, the download URL can be saved into the database. With the download URL, the file can be accessed without first downloading it programmatically. The download URL can be set to the html image tag as a source, and the image will be loaded. (Program code 6)

Program code 6. Save file to Firebase Storage.

```
const handleSave = async (e) => {
  e.preventDefault();
  const name = e.target.name.value;
  const description = e.target.description.value;
```

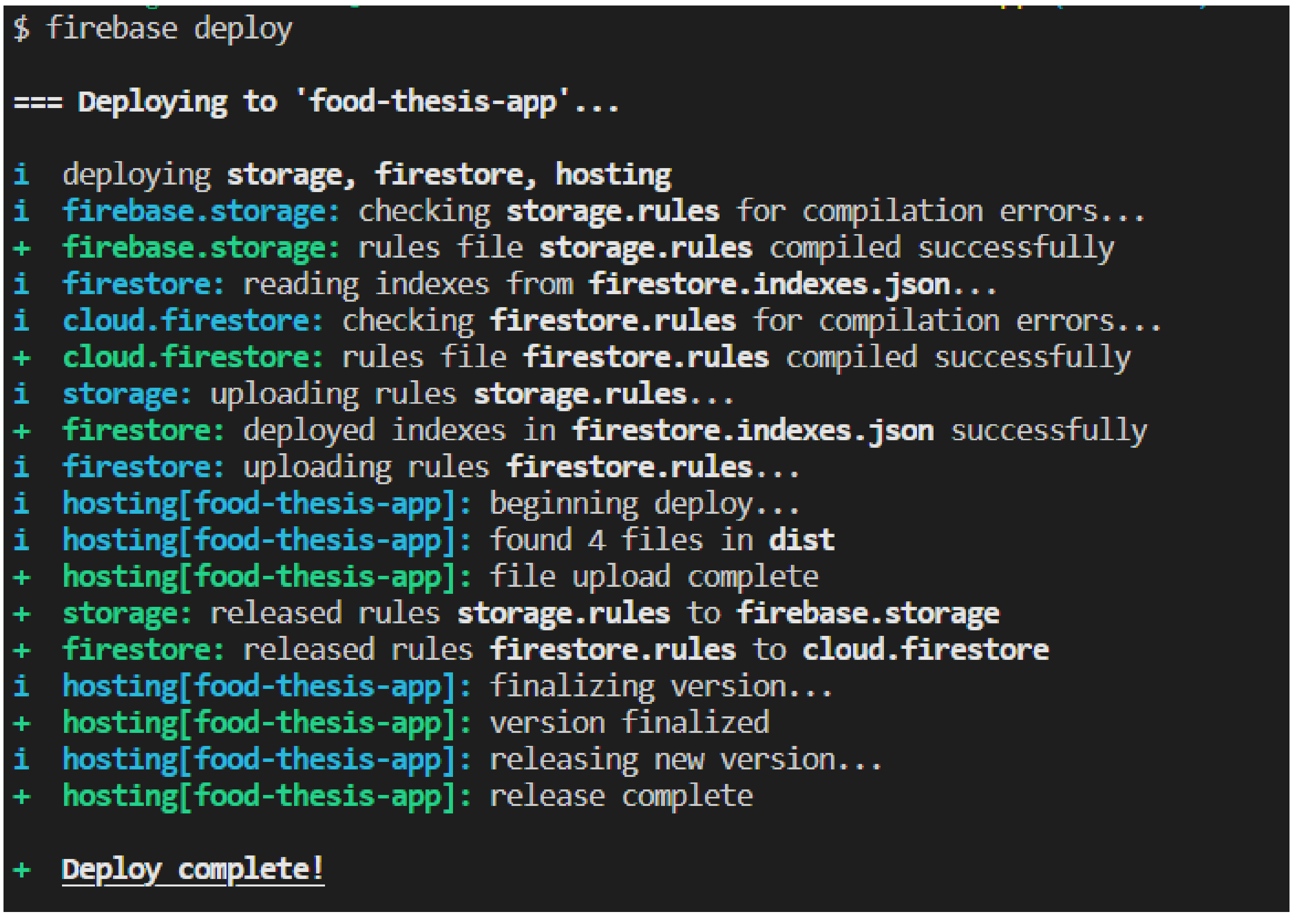
```
    const photo = e.target.photo.files[0];

    // Save image file to storage
    const fileRef = ref(storage, name);
    const uploadTask = uploadBytesResumable(fileRef, photo);

    uploadTask.on(
      "state_changed",
      (snapshot) => {
        const progress =
          (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
        console.log("Upload is " + progress + "% done");
      },
      (error) => {
        console.error(error);
      },
      () => {
        getDownloadURL(uploadTask.snapshot.ref).then(async (downloadURL) =>
{
          console.log("File available at", downloadURL);
          // Save to database
          saveFood({ name, description, photo: downloadURL, uid: user })
            .then(() => {
              navigate("/foods/browse");
            })
            .catch((error) => console.error(error));
        });
      }
    );
  };
```

8.5 Hosting

With Firebase Hosting developers can server static and dynamic content or microservices. All connections are secured with SSL. It has a feature to preview the content in temporary URL and use GitHub integration for automated deployments. Firebase has a default domain generated, but custom domains can be set. (Firebase, n.d-c.)

Deploying to Hosting is done with the automated deployments from GitHub repository, or with the CLI. With the firebase deploy command, CLI let the developer choose what services are deployed. It automatically detects what services have been updated from the current deployment. As can be seen in Figure 5. Deployment process in Firebase CLI., firebase deployment runs multiple checks during the process. It is checking the security rules, indexes and all changes that have been made to the files that are deployed.

Figure 5. Deployment process in Firebase CLI.



```
$ firebase deploy

=== Deploying to 'food-thesis-app'...

i  deploying storage, firestore, hosting
i  firebase.storage: checking storage.rules for compilation errors...
+  firebase.storage: rules file storage.rules compiled successfully
i  firestore: reading indexes from firestore.indexes.json...
i  cloud.firestore: checking firestore.rules for compilation errors...
+  cloud.firestore: rules file firestore.rules compiled successfully
i  storage: uploading rules storage.rules...
+  firestore: deployed indexes in firestore.indexes.json successfully
i  firestore: uploading rules firestore.rules...
i  hosting[food-thesis-app]: beginning deploy...
i  hosting[food-thesis-app]: found 4 files in dist
+  hosting[food-thesis-app]: file upload complete
+  storage: released rules storage.rules to firebase.storage
+  firestore: released rules firestore.rules to cloud.firestore
i  hosting[food-thesis-app]: finalizing version...
+  hosting[food-thesis-app]: version finalized
i  hosting[food-thesis-app]: releasing new version...
+  hosting[food-thesis-app]: release complete

+  Deploy complete!
```

8.6 Cloud Functions

Firebase Cloud Functions is a serverless framework that allows to extend the backend with custom functions. Functions can be fired by event trigger, call from the app, scheduled and HTTPS requests. Event can be a database, storage, or authentication event. It is possible to modify documents from other collections based on recently saved documents in database. Moreover, it is possible to modify the user data after the user account is created. (Firebase, n.d-f.)

8.7 Analytics

Firebase has built-in Google Analytics that can be utilized to track app usage and user engagement. When analytics is set up in the application, it automatically captures certain default events and user properties. It is also possible to create custom events to get insight from the application. Analytics SDK automatically captures events and captured data is available in the Firebase console. (Firebase, n.d.)

8.8 Privacy

Firebase acts as a data processor, and the main company Google operates as a data controller. Developers can select a geographical region where most of the services are running, and data is stored. Firebase Authentication service is processing data in the United States, but the data handling is GDPR compliant via the Standard Contractual Clauses agreed with the EU. Storage, databases, and functions are geographical region-specific services. (Firebase, n.d-h)

8.9 Pricing

Firebase has two pricing tiers. The free tier is called a Spark plan, that has limited features and usage. Spark plan does not include Cloud Functions. If a user wants to use the Cloud Functions, the payment plan must be upgraded to a Blaze plan which is a pay as you go solution. Blaze plan includes all firebase features, and some of the features are completely free. The key features mentioned above in the Firebase chapter have free quota from the Spark Plan and above that cost is per usage.

The free quota is generous with the free 50,000 monthly active users for the Authentication service. The Firestore database have 50,000 reads, 20,000 writes and deletes per day for free. (Firebase, n.d-b)

8.10 Evaluate of the Firebase

Firebase has a user-friendly dashboard that non-programmers should be able to use. Firebase is a well-productized service that suits well to web development. The development environment helps the development work as the modifications can be tested locally with the Firebase. All the main services can be used with the development environment. It is much safer to develop new features while not connected to the cloud service. A mistake in the code can cause expenses in the cloud environment, but locally it does not matter. Firebase is a GDPR compliant service, and it is an option for the EU companies. If it is crucial for the use case to handle all data within the EU region, Firebase is not the appropriate option. This is because the most sensitive data, which is user data, is processed in the United States.

For a junior level developer, the service is easy to use, and the documentation is at first class level. Documentation has detailed instructions for every feature of every service. Documentations contain a wide range of code snippets and explanations of how to use services. The learning curve for the basic usage is low, but for advanced usage it might take some time. For experienced developers the service should be easy to use.

9 AWS Amplify

AWS Amplify is a full stack platform to develop web and mobile applications. It is based on Amazon Web Services and has access to more than 175 AWS services. Amplify Studio is a visual interface where developers can build backend features and UI components. The UI components still require a developer to add them into the application. (Amazon Web Services, Inc., n.d-e.)

Amplify project requires Amazon Web Services account. Amplify Studio is used to build the backend of the application. The features can be activated from the studio one-by-one. The initial phase was quick since there are not many configurations to do.

9.1 Configuring the application and development environment

Amplify creates the amplify folder into project that contains the configuration files. Everything is set up automatically by pulling the latest version of the app from the cloud. As can be seen from Program code 6, Amplify does not require much code to begin. Only one line of code in addition to import statements in the root route of the application is enough to start (Program code 7). Some of the Amplify features can be used locally without an AWS account, however it cannot be counted as a development environment, because it is an offline mode. Application is always connected to cloud if all services are in use, even though it could use some features in an offline mode.

Program code 7. Initialize Amplify in the root of the application.

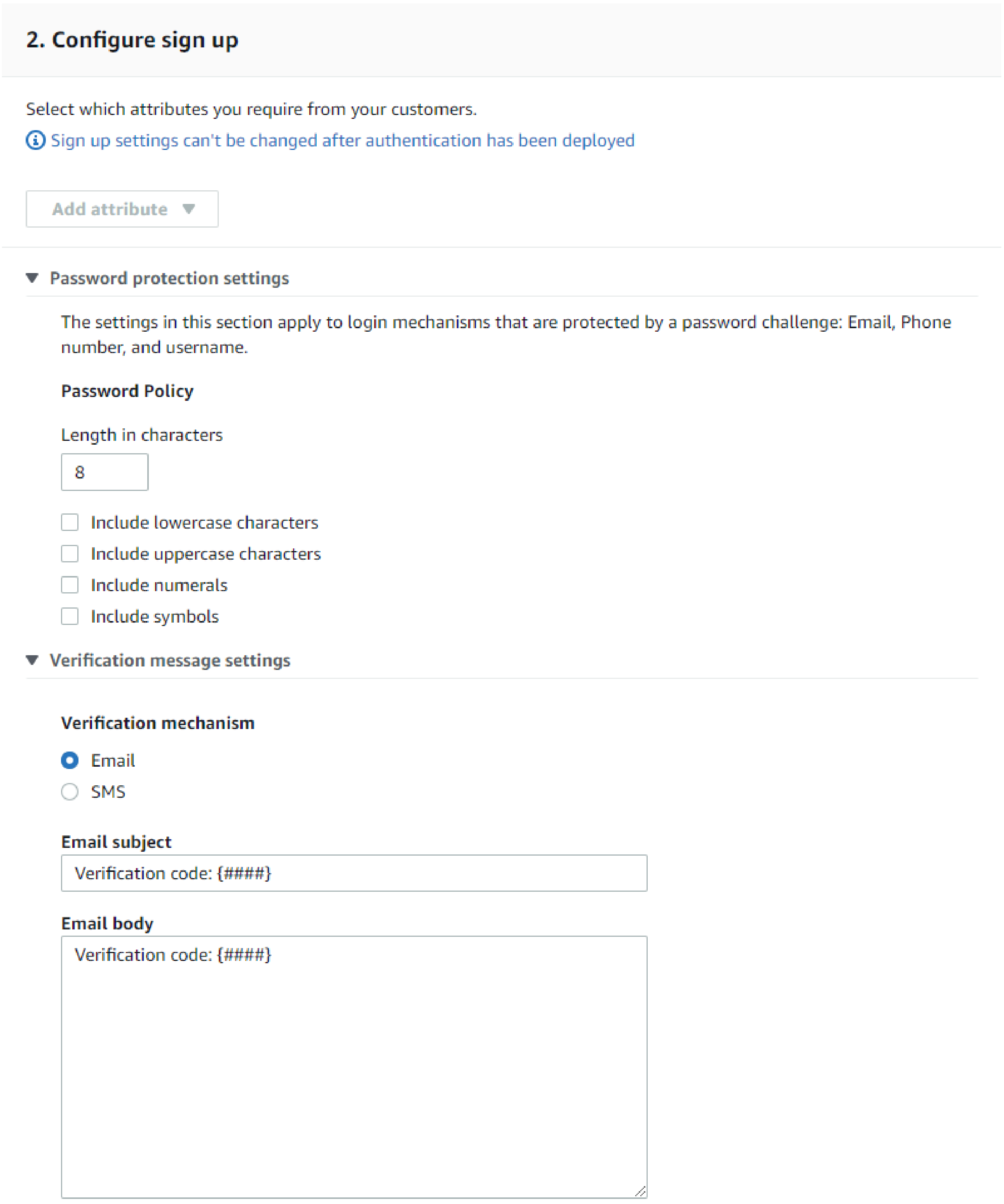
```
import { Amplify } from 'aws-amplify';
import awsconfig from './aws-exports';
Amplify.configure(awsconfig);
```

9.2 Authentication

Amplify authentication is based on Amazon Cognito service. Cognito supports OAuth 2.0, SAML and OpenID Connect protocols. Pre-built and customizable user interface for sign-up and sign-in processes. Email and password sign-in method is extended with Facebook, Google, Amazon, and Apple sign-in. Cognito comes with security services such as bot detection that protects applications from automated accounts. (Amazon Web Services, Inc., n.d-h; Amazon Web Services, Inc, n.d-a)

Amplify application has email login as a default login method. As can be seen in the Figure 6, additional attributes can be set for the user. Additional attributes are for example: birthdate, gender, picture, and nickname. After the authentication is deployed, attributes cannot be changed. Password policy can be set from the studio, settings include length, and forcing users to use lowercase and uppercase character, numbers, and symbols. Account verification mechanism can be set to use email or SMS. The verification messages are fully customizable.

Figure 6. Configure sign up in Amplify Studio.



2. Configure sign up

Select which attributes you require from your customers.

ⓘ Sign up settings can't be changed after authentication has been deployed

▼ Password protection settings

The settings in this section apply to login mechanisms that are protected by a password challenge: Email, Phone number, and username.

Password Policy

Length in characters: 8

- [ ] Include lowercase characters
- [ ] Include uppercase characters
- [ ] Include numerals
- [ ] Include symbols

▼ Verification message settings

Verification mechanism

[✓ Email] SMS

Email subject

Verification code: {####}

Email body

Verification code: {####}

In demo application users are authenticated with email and password. When the user is successfully authenticated, the user identifier is saved into the browser's local storage. Saving the user into local storage is not necessary with Amplify, because the user with attributes is automatically saved into local storage by the SDK. (Program code 8)

Program code 8. Create user function with Amplify.

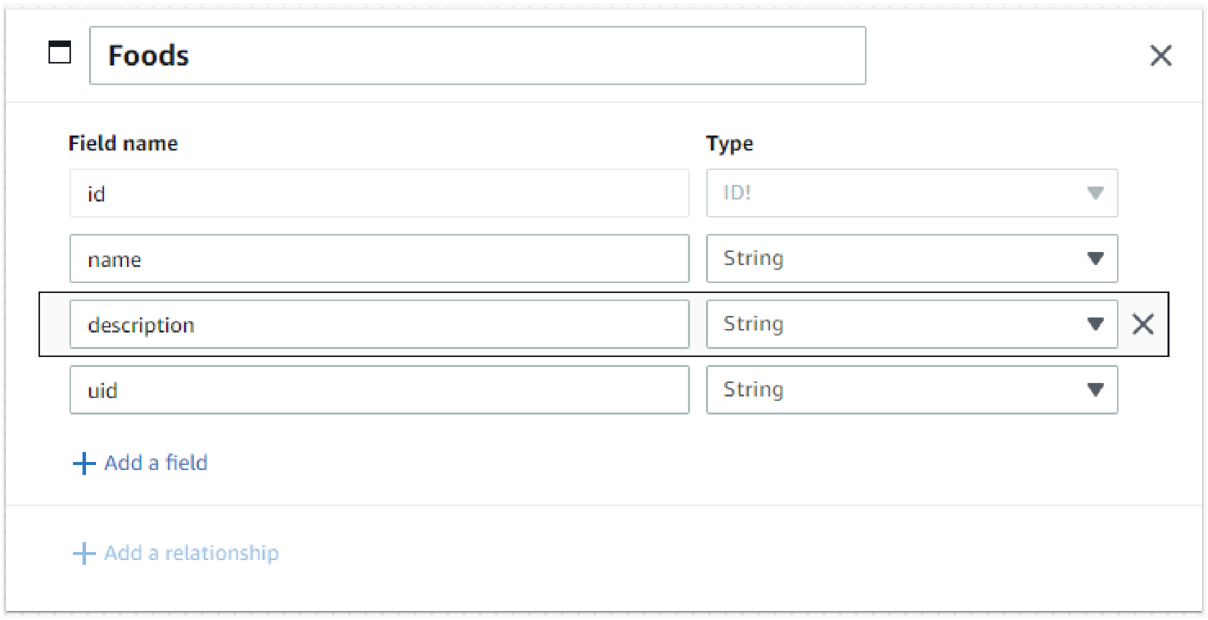
```
const createUser = async (credentials) => {
    try {
      const response = await Auth.signUp(credentials);
      if (response?.userSub) {
        setUser(response.username);
        localStorage.setItem("user", response.username);
      }
    } catch (error) {
      console.error(error);
    }
  };
```

9.3 Database

Amplify Datastore is a NoSQL database based on Amazon DynamoDB. It is an on-device storage engine that automatically synchronizes data between applications and database in the cloud. DynamoDB supports both key-value and document data models with flexible schemas. Developers can use visual data modeling tools to define data models. (Amazon Web Services, Inc., n.d-b.)

Database requires a data model before inserting data. It can be written as a GraphQL schema, or by using the Visual editor. GraphQL is a query language with capability to query data from many resources in one request (GraphQL, n.d). Figure 7 presents the Foods data model. It is simple to set the properties and types with the Visual editor. The types are validated before inserting the data, and invalid values are not accepted. When the data model is saved, Amplify automatically builds a UI form component for the data model, which can speed up the development process if used correctly. The demo application does not utilize this feature due to the focus on the backend integrations.

Figure 7. Visual editor for data modeling in Amplify.

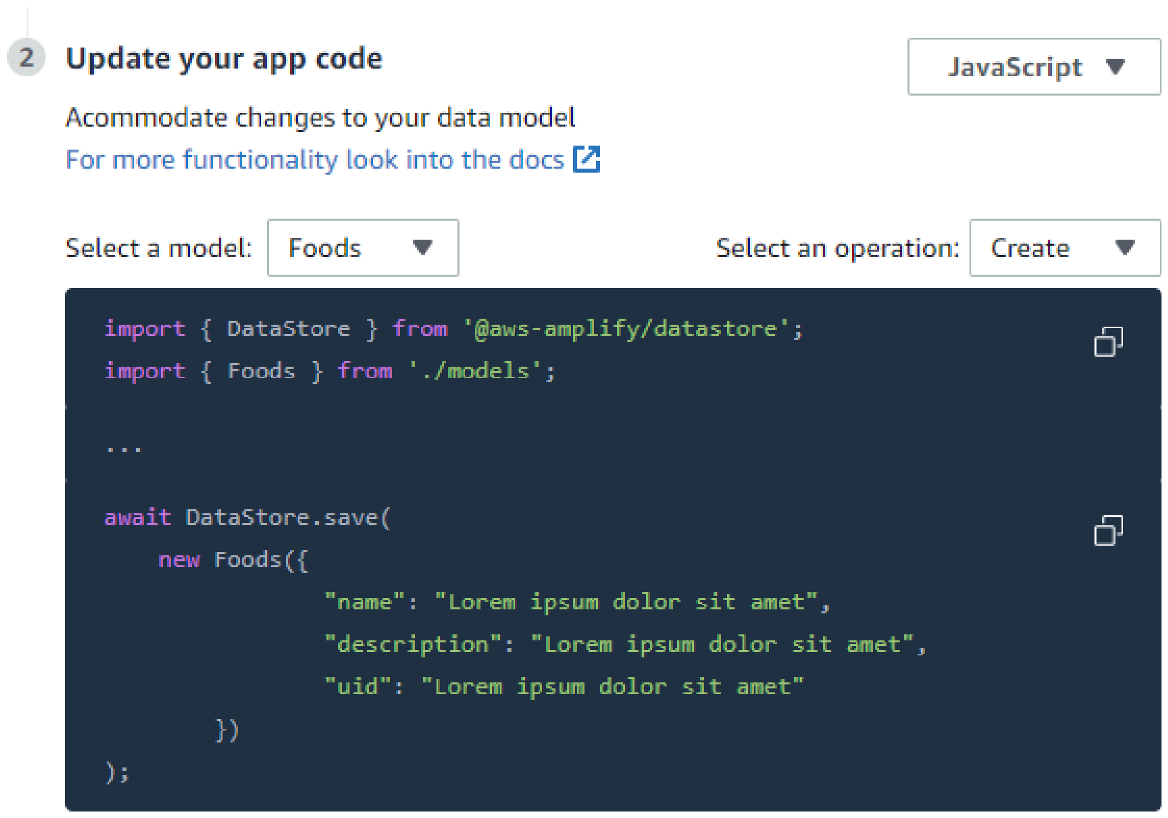


Foods

Field name	Type
id	ID!
name	String
description	String
uid	String

After deploying the data model, Amplify offers a code snippet for operations to the data model. Code snippets are available as JavaScript, Java, Kotlin, Swift and Flutter. It is easy to copy and paste the code to the application. Figure 8 shows a create operation, which can be changed to also update, delete, and read operations. Also, the data model can be changed to any model data model in the project. As can be seen from Figure 8, database operation uses the Foods model. The model validates the data before inserting it into the database, and prevents insertion if data is not valid. By default, the database is in offline mode, but with a single line of code it is turned into online mode.

Figure 8. Amplify code snippet to insert data.


2
Update your app code
JavaScript
Acommodate changes to your data model
For more functionality look into the docs
Select a model: Foods
Select an operation: Create

```
import { DataStore } from '@aws-amplify/datastore';
import { Foods } from './models';

...

await DataStore.save(
    new Foods({
                "name": "Lorem ipsum dolor sit amet",
                "description": "Lorem ipsum dolor sit amet",
                "uid": "Lorem ipsum dolor sit amet"
        })
);
```

9.4 Storage

Amplify has built-in support for Amazon S3 object storage. S3 storage stores data as objects that consists of a file and optional metadata. Permissions can be set to the individual file. (Amplify Dev Center, n.d-a.)

Access rules for storage can be set from the Studio UI. In the demo application rule is set to the mode where only authenticated users can upload and view content. Amplify does not provide direct URL to the file that is saved into storage. Files must be downloaded programmatically when needed.

9.5 Lambda Functions

Amplify uses AWS Lambda serverless computer service to run functions. Lambda function can be triggered by HTTP requests, updates on objects in S3 storage or table updates in database. With Lambda functions developers can extend pre-built backend functionalities from Amplify or create completely custom backend services (Amazon Web Services, Inc., n.d-i.)

9.6 Hosting

Amplify Hosting service is designed for modern web applications. It has support for modern web frameworks such as React, Angular, Vue and more. With CI/CD workflows developers can build automated workflows to deploy code from git repository. Free SSL certificates are granted for custom domains. (Amazon Web Services, n.d-d.)

Deploying the demo application into the Hosting service cannot be done in the CLI. The choices are to deploy from the code repository or upload the project's build files to the AWS console. The second approach was used to deploy the demo application. While running the hosted application, an access denied error occurs from Amplify when the page is refreshed. The origin of the error has not been identified by the author. By default, the hosting service uses a default domain name generated by Amplify, which can be upgraded to a custom domain.

9.7 Analytics

Amplify has built-in support for Amazon Pinpoint service. Amazon Pinpoint is an AWS service that can be used to engage with customers across multiple messaging channels (Amazon Pinpoint, n.d.). Usage of the application can be tracked with the analytics by saving the events. Automatic tracking is an option that can be used for session, page view or page event tracking. Analytic features provide a wide range of tools to track users in the application. All features of Analytics are not compatible with the JavaScript SDK, and some are compatible only with the mobile environments. (Amplify Dev Center, n.d-b)

9.8 Privacy

Amplify acts as a data processor, and the main company Amazon Web Services operates as a data controller. Developers can select a geographical region where the services are running, and data is stored. Amazon claims that all their services are GDPR compliant via the Standard Contractual Clauses agreed with the EU. (Amazon Web Services, Inc., n.d-f.)

9.9 Pricing

Amplify has a pay as you go pricing model. Prices are the same as when bought separately from AWS without using Amplify. The main services, like authentication, database and storage have a free tier. Authentication has 50,000 monthly users included in the free tier and Database offers resources that are enough to handle 200M requests per month for free. (Amazon Web Service, Inc, n.d-k; Amazon Web Service, Inc, n.d-l)

9.10 Evaluate of the Amplify

Amplify has a simple dashboard but overall, the service is not well productized. It seems like independent services have been glued together. While this study does not review the UI builder, it has a potential to be a useful feature that increases the speed of development. The entire backend can be built with the Studio by a non-programmer person. Pulling the code into the codebase was easy, as was the configuration of the application, but from this stage upon programming skills are required. Amplify advertises that applications can be built in a few hours, which is a believable claim. But for more than basic usage, the learning curve is quite steep at least for developers who are new to AWS. The documentation is decent, but information is scattered around the AWS. The information can be found from that specific service's documentation from AWS, Amplify's documentation or from the Studio. Documentation contains code snippets of how to use the services, but only for the basic scenarios. The code snippets in the Amplify Studio are helpful in the development process because they are available when needed. The lack of the development environment is a disadvantage for Amplify service. If the application is developed in the cloud environment, it can cause unexpected expenses. Database has the offline mode that should be used during the development, but otherwise application is connected to the cloud.

10 Results

The research shows that Backend services have all the core parts of the modern web application. The demo application is built successfully with both backend services introduced in this thesis. A few key differences have been identified between the services. The largest difference between Amplify and Firebase is that Firebase is better productized. The impression of Amplify is that independent services have been glued together, which can be seen especially in the documentation where information is scattered. Firebase gives the impression that it is designed for the backend service. Some users may find an advantage that all AWS services are in the reach of Amplify. Another advantage for Amplify is that it is a Full stack platform, that includes the UI builder which can be used to build Form components that match to the application's data models. Another difference that should be noticed is that Firebase has the development environment and Amplify does not. Both services are GDPR compliant, but Firebase is not an option if it is important that data is handled in the EU region. Firebase handles the Authentication data in the United States.

Based to the development experience, backend services are a solid option to build web applications. The development is faster because a large part of the application is outsourced to backend service. However, the savings come with the drawbacks that are deep vendor lock-in and that application cannot be very complicated because of the universal nature of the services causes limited backend features. Backend features can be extended with the serverless functions, but it is probably better to build a custom backend if the application needs complicated backend features. Both reviewed services in the thesis have non-relational databases which may not be ideal for every use case. The demo application project shows that using backend services requires very little configuration on infrastructure and for that reason is a very fast way to build applications.

11 Summary

The goal of this thesis was to find out if backend services are a relevant choice for web applications. Services were tested with the small demo application, which does not utilize every feature that services offers. However, it was enough to test the core features which were Authentication, Database and Storage. The development work showed that backend services are a fast way to develop a web application. Firebase and Amplify have a few differences therefore it is recommended to investigate multiple services before selecting one. Due to the vendor lock it is difficult to change the service later. Backend services are especially suitable for small companies who do not have the resources to hire large development teams. Due to the free tiers of the services, it is possible to test the application on the market without running costs. Backend services can be utilized with a decent learning curve even by an inexperienced developer. It is a good solution for a small software company who lacks the resources to hire multiple developers.

During the process of writing the thesis my knowledge was expanded on serverless computing and backend services. Backend services will be in the author’s toolbox for the upcoming projects. The demo applications built during the thesis are deleted from the hosting services. Applications served their purpose and won’t be developed further.

References

Ahopelto, T. (2018). Nollaksi vai ykköseksi (No. 62; EVA Analyysi). Elinkeinoelämän valtuuskunta.

Amazon Pinpoint. (n.d.). What Is Amazon Pinpoint? Retrieved 16 April 2023, from https://docs.aws.amazon.com/pinpoint/latest/userguide/welcome.html

Amazon Web Services, Inc. (n.d-a.). AWS Amplify Dev Center Docs Authentication Social Sign-In. Retrieved 2 February 2023, from https://docs.amplify.aws/lib/auth/social/q/platform/js/

Amazon Web Services, Inc. (n.d-b.). App Data Modeling - Aws Amplify Datastore - Aws. Retrieved 6 February 2023, from https://aws.amazon.com/amplify/datastore/?nc=sn&loc=3&dn=2

Amazon Web Services, Inc. (n.d-c.). AWS Amplify Features. Retrieved 1 February 2023, from https://aws.amazon.com/amplify/features/

Amazon Web Services, Inc. (n.d-d.). Aws Amplify Hosting. Retrieved 6 February 2023, from https://aws.amazon.com/amplify/hosting/

Amazon Web Services, Inc. (n.d-e.). Aws Amplify. Retrieved 12 February 2023, from https://aws.amazon.com/amplify/

Amazon Web Services, Inc. (n.d-f.). Gdpr - Amazon Web Services(Aws). Retrieved 20 February 2023, from https://aws.amazon.com/compliance/gdpr-center/

Amazon Web Services, Inc. (n.d-g.). What Is Cloud Storage? Retrieved 23 February 2023, from https://aws.amazon.com/what-is/cloud-storage/

Amazon Web Services, Inc. (n.d-h.). What Is Web Hosting? - Web Hosting Service Explained. Retrieved 4 March 2023, from https://aws.amazon.com/what-is/web-hosting/

Amazon Web Services, Inc. (n.d-i.). Serverless Computing - Aws Lambda Features. Retrieved 4 March 2023, from https://aws.amazon.com/lambda/features/

Amazon Web Services, Inc. (n.d-k.). Amazon Dynamodb Pricing | Nosql Key-Value Database | Amazon Web Services. Retrieved 23 April 2023, from https://aws.amazon.com/dynamodb/pricing/

Amazon Web Services, Inc. (n.d-l.). Pricing | Amazon Cognito | Amazon Web Services(Aws). Retrieved 23 April 2023, from https://aws.amazon.com/cognito/pricing/

Amplify Dev Center. (n.d-a.). Storage Concepts. Retrieved 3 March 2023, from https://docs.amplify.aws/lib/storage/overview/q/platform/js/

Amplify Dev Center. (n.d-b.). Analytics. Retrieved 23 April 2023, from https://docs.amplify.aws/lib/analytics/personalize/q/platform/js/

Arkhipov, A. (2023, March 17). TechMagic. Top Benefits of Backend as a Service. https://www.techmagic.co/blog/backend-as-a-service/

Auth0. (n.d-a.). What Is Authentication? Definition and Uses. Retrieved 4 March 2023, from https://auth0.com/intro-to-iam/what-is-authentication

Auth0. (n.d-b.). What Is OAuth 2.0 and What Does It Do for You? Retrieved 4 March 2023, from https://auth0.com/intro-to-iam/what-is-oauth-2

Auth0. (n.d-c.). What Is SAML 2.0 and How Does It Work for You? Retrieved 4 March 2023, from https://auth0.com/intro-to-iam/what-is-saml

Auth0. (n.d-d.). What Is OpenID Connect and What Do You Use It For? Retrieved 4 March 2023, from https://auth0.com/intro-to-iam/what-is-openid-connect-oidc

Cloudflare. (n.d-a.). What Is Vendor Lock-in? | Vendor Lock-in and Cloud Computing. Retrieved 10 March 2023, from https://www.cloudflare.com/en-gb/learning/cloud/what-is-vendor-lock-in/

Cloudflare. (n.d-b.). Vendor lock-in explained in cloud computing. [Figure] Retrieved 10 March 2023, from https://www.cloudflare.com/en-gb/learning/cloud/what-is-vendor-lock-in/

Chiarelli, A. (2018). Beginning react: Simplify your frontend development workflow and enhance the user experience of your applications with react. Packt Publishing Ltd.

Firebase. (n.d-a.). Firebase Authentication. Retrieved 2 February 2023, from https://firebase.google.com/docs/auth

Firebase. (n.d-b.). Firebase Pricing. Retrieved 2 February 2023, from https://firebase.google.com/pricing

Firebase. (n.d-c.). Firebase Hosting. Retrieved 3 February 2023, from https://firebase.google.com/docs/hosting

Firebase. (n.d-d.). Firebase Realtime Database. Retrieved 6 February 2023, from https://firebase.google.com/docs/database

Firebase. (n.d-e.). Firestore. Retrieved 6 February 2023, from https://firebase.google.com/docs/firestore

Firebase. (n.d-f.). Cloud Functions for Firebase. Retrieved 6 February 2023, from https://firebase.google.com/docs/functions

Firebase. (n.d-f.). Execute intensive tasks in the cloud instead of a user's device. [Figure] Retrieved 6 February 2023, from https://firebase.google.com/docs/functions

Firebase. (n.d-g.). Cloud Storage for Firebase. Retrieved 6 February 2023, from https://firebase.google.com/docs/storage

Firebase. (n.d-h.). Privacy and Security in Firebase. Retrieved 6 February 2023, from https://firebase.google.com/support/privacy

Firebase. (n.d-i.). Firebase. Retrieved 12 February 2023, from https://firebase.google.com/

Firebase. (n.d-j.). Google Analytics for Firebase. Retrieved 17 April 2023, from https://firebase.google.com/docs/analytics

Gallardo, E. (2023, January 6). What Is Back-End Development? https://builtin.com/software-engineering-perspectives/back-end-development

GraphQL. (n.d.). A Query Language for Your API. Retrieved 6 February 2023, from https://graphql.org/

IBM. (n.d.). IaaS vs. PaaS vs. SaaS. Retrieved February 1, 2023, from https://www.ibm.com/topics/iaas-paas-saas

IBM. (2021, July 13). Sdk vs. Api: What's the Difference? https://www.ibm.com/cloud/blog/sdk-vs-api

Jwt. (n.d.). Json Web Tokens Introduction. Retrieved 4 March 2023, from http://jwt.io/

Komanduri, S. K., & Hoque, S. (2018). Full-stack react projects: Modern web development using react 16, node, express, and mongodb. Packt Publishing.

Larja, L., & Peltonen, J. (2023). Työvoiman saatavuus, työvoimapula ja kohtaanto-ongelmat vuonna 2022 (No. 113/2023).

Metwalli, S. (2022, December 22). What Is Front-End Development? https://builtin.com/software-engineering-perspectives/front-end-development

Microsoft Azure. (n.d.). What Is Cloud Computing? A Beginner's Guide. Retrieved February 1, 2023, from https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing

Oracle. (n.d.). What Is a Database? Retrieved 6 February 2023, from https://www.oracle.com/database/what-is-database/

Privacy Shield. (n.d.). Privacy Shield Program Overview. Retrieved 12 February 2023, from https://www.privacyshield.gov/Program-Overview

React – A JavaScript library for building user interfaces. (n.d.). Retrieved 3 February 2023, from https://reactjs.org/

Ruparelia, N. (2016). Cloud computing. MIT Press.

Sharma, S. (2020). Data privacy and GDPR handbook. John Wiley & Sons, Inc.

Tietosuojavaltuutetun toimisto. (n.d.). Tietosuojavaltuutetun toimisto. Retrieved 12 February 2023, from https://tietosuoja.fi/komission-hyvaksymat-vakiolausekkeet

Wellens, P. (2015). Practical web development: Learn CSS, JavaScript, PHP, and more with this vital guide to modern web development. Packt Publishing.

Your Europe. (n.d.). Data Protection under GDPR. Retrieved 6 February 2023, from https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_en.htm

Zhang, M. (2023, January 1). Dgtl Infra. Top 10 Cloud Service Providers Globally in 2023. https://dgtlinfra.com/top-10-cloud-service-providers-2022/

Annex 1: Material management plan

Code from the development work will be saved into the public GitHub repository. The development diary is stored on the author’s personal Google Drive. The diary is kept at Google Drive for at least one year after the completion of the thesis. Projects created in Firebase and Amplify are terminated.