LAMK
Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

HITSAUSKONEEN PANEELIN TAUSTAKUVAN LUONTIPALVELU

Case: Kemppi Oy

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Tieto- ja viestintätekniikka
Ohjelmistotekniikka
Opinnäytetyö
Syksy 2019
Tuomas Alhainen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Alhainen, Tuomas	Opinnäytetyö, AMK	Syksy 2019
	Sivumäärä	
	38	

Työn nimi

Hitsauskoneen paneelin taustakuvan luontipalvelu
Case: Kemppi Oy

Tutkinto

Insinööri (AMK), tieto- ja viestintätekniikka

Tiivistelmä

Opinnäytetyössä toteutettiin hitsauslaitteen paneelin taustakuvan luonti- ja muokkauspalvelu. Palvelun vaatimat järjestelmät hyödyntävät pilvilaskentaa, sekä niitä ylläpidetään Amazon Web Services -pilvipalvelussa. Toimeksiantajana toimi Kemppi Oy.

Työssä tutkittiin keinoja käyttöliittymän laajan selain- ja alustatuen saavuttamiseksi sekä käyttäjäkokemuksen edistämiseksi. Erityisesti painoitettiin yhtenäisyyden saavuttamista palvelun ulkonäössä sekä sen toiminnoissa.

Käyttöliittymä toteutettiin Angular-ohjelmistokehyksellä, ja sitä jaetaan hyödyntämällä AWS Cloudfront- ja S3-palveluita. Taustajärjestelmässä hyödynnettiin Serverless-arkkitehtuuria ajamalla NodeJS-ajoympäristöllä kirjoitettua sovellusta AWS Lambda -palvelussa. Molemmissa järjestelmissä käytettiin myös Typescript-kirjastoa tyypittämistä varten.

Lopputuloksena oli onnistunut projekti, joka julkaistiin ajallaan. Käyttöliittymässä loppukäyttäjän on mahdollista muokata ja ladata oma kuva hitsauslaitteen paneelia varten. Käyttöliittymä toimii uusimmilla selainversioilla ja on käytettävissä perinteisten tietokoneiden lisäksi myös mobiililaitteilla. Käyttöliittymässä on myös ohjeet kuvan asentamiseksi, joten loppukäyttäjä saa kaiken tarvitseman informaation käyttämällä palvelua.

Asiasanat

Amazon Web Services, Angular, Serverless, NodeJS, käyttäjäkokemus

Abstract

Author(s)	Type of publication	Published
Alhainen, Tuomas	Bachelor's thesis	Autumn 2019
	Number of pages	
	38	

Title of publication

Service for creating a background for welding machine panels
Case: Kemppi Oy

Name of Degree

Bachelor's Degree Programme in Information and Communications Technology

Abstract

The objective of this thesis was to produce a service for creating a background for welding machine panels. The systems required for the service will use cloud computing and they will be maintained in Amazon Web Service. This thesis was done for Kemppi Oy.

The theoretical section of the thesis dealt with ways to achieve broad browser and platform support and improved user experience in the user interface. These are examined from the perspective of uniformity in the appearance and functionality of the user interfaces.

The frontend was developed with the Angular-framework and it is distributed with the AWS CloudFront- and S3-services. The backend implements Serverless architecture by executing an application written with NodeJS runtime in the AWS Lambda-service. Both systems also use the Typescript-library for code typing.

The project succeeded and was published in time. The frontend provides ways for the end-user to modify and download a picture of their choice for their welding machine's panel. The user interface works with the latest browser versions and is usable with traditional computers and mobile devices. The user interface also includes a guide for the image installation process, so the end-user gets all the information they need by using the service.

Keywords

Amazon Web Services, Angular, Serverless, NodeJS, user experience

SISÄLLYS

LYHENTEET

AWS Amazon Web Services, eli Amazonin pilvipalvelualusta.

HTTP Hypertext Transfer Protocol on protokolla, jota käytetään tiedonsiirtoon selaimissa.

REST Representional State Transfer on arkkitehtuuri, joka määrittää ohjeita web-liikenteen toiminnallisuuteen ja rakenteeseen liittyen.

1 JOHDANTO

Opinnäytetyön toimeksiantajana toimi lahtelainen hitsausalalla toimiva yritys Kemppi Oy. Päätuotteinaan Kemppi tarjoaa hitsauslaitteita, niihin liittyviä asiantuntijapalveluita sekä ohjelmistoratkaisuja hitsaamisen kehittämiseksi ja hallitsemiseksi. Kempillä on yli 800 työntekijää 17:ssä eri maassa, ja yritys palvelee asiakkaita yli 60 maassa. Yrityksen liikevaihto vuonna 2018 oli noin 125 miljoonaa euroa. (Kemppi Oy 2019.)

Pilvipalvelut tai pilvilaskenta perustuu loppukäyttäjän laskuttamiseen palveluista, kuten tietokannoista tai palvelimista, jotka sijaitsevat pilvipalvelussa. Näin loppukäyttäjä välttyy kalliin laitteiston hankinnalta tai infrastruktuurin rakentamiselta ja suunnittelemiselta, sillä heillä on pääsy haluamiinsa toimintoihin internetin yli. Palveluista maksetaan usein käytön mukaan, sekä niitä voidaan laajentaa vaivattomasti omiin käyttötarkoituksiin sopivaksi. (Mathew 2019, 1.)

Pilvipalvelut myös mahdollistavat Serverless-arkkitehtuurin käyttämisen. Serverless eli toiselta nimeltään Backend as a service (BAAS) tai Function as a service (FAAS) mahdollistaa palveluiden ajamista ilman arkkitehtuurin tai palvelimien ylläpitämistä (Monroe & Lovaas 2018, 14; Amazon Web Services 2019c). Vaikka nimi Serverless viittaa palvelimien poissaoloon kyseisestä ympäristöstä, todellisuudessa asia ei ole näin. Taustalla palveluntarjoaja yhä ylläpitää taustajärjestelmän vaatimia palveluita, mutta kehittäjälle näkyy vain rajapinta. (Monroe & Lovaas 2018, 14.)

Opinnäytetyön tavoitteena on kehittää kuvanmuokkaus- ja latauspalvelu hitsauslaitteiden käyttäjille, jotta he voivat asentaa laitteensa paneeliin haluamansa taustakuvan. Lisäksi on tarkoitus saada käyttöliittymä toimimaan usealla eri alustalla ja näyttökoolla. Palvelun käyttöliittymä tullaan tekemään toimeksiantajan pyynnöstä Angular 2+ -ohjelmistokehystä käyttäen. Käyttöliittymässä luodaan kuvan muokkaamiseen vaadittavat asetukset ja kuva ladataan. Taustajärjestelmä tulee käyttämään NodeJS-ajoympäristöä ja olemaan vastuussa kuvan muokkauksesta ja konvertoinnista hitsauslaitteelle sopivaksi. Molemmat palvelun tulevat myös hyödyntämään Amanzon Web Services -pilvipalveluita.

2 AMAZON WEB SERVICES

AWS eli Amazon Web Services on globaali pilvipalvelualusta, jota voidaan käyttää noin 190 valtiossa (Mathew 2019, 1). Yrityksellä on yli miljoona asiakasta, jotka jakautuvat useille eri toimialoille ja yrityskoille. Alustassa on tarjolla yli 165 palvelua useaan eri käyttötarkoitukseen, kuten tallennukseen, analytiikkaan, koneoppimiseen, tietoturvaan ja tietoverkkoihin liittyen. (Amazon Web Services 2019b.)

AWS tarjoaa myös kehitystyökaluja useille ohjelmistokielille, joiden avulla voidaan kutsua palveluiden toimintoja koodista. Työkalut sisältävät rajapintoja useimmille AWS-palveluille, kuten S3:lle, EC2:lle ja DynamoDB:lle. Javascript-ohjelmointikielelle on tehty myös erikseen selain ja NodeJS-ajoympäristölle omat kehitystyökalut. (Amazon Web Services 2019s; Amazon Web Services 2019t.)

2.1 S3

Amazon S3 eli Simple Storage Service on skaalatuvua tietovarasto, jolla on objektimainen rakenne (Mathew 2019, 84). S3:sta voidaan käyttää miltei minkä tahansa tyyppisen datan tallentamiseen ja noutamiseen, jos sen sisältö ei riko soveltuvia lakeja tai ole Amazonin käytäntöjen vastaista (Amazon Web Services 2019w; Amazon Web Services 2019å). Tallennetulle datalle taataan 99%:n säilyvyys, sillä kaikki objektit kopioidaan useaan eri paikkaan (Amazon Web Services 2019f).

2.1.1 Bucket

Bucket on S3-hierarkiassa ylin komponentti ja sitä käytetään objektien säilömiseen, käyttötilin identifioimiseen tallennus- ja siirtokuluihin liittyen, käytön rajoittamiseen sekä S3-nimiavaruuden organisoimiseen (Amazon Web Services 2019e). Luotu bucketti voi sijaita vain yhdellä Amazonin käyttöalueella, mutta sen nimi on estetty myös muilla alueilla, pitäen nimiavaruuden organisoituna (Amazon Web Services 2019i).

Buckettia sekä sen sisältöä voidaan käyttää selaimella Amazonin konsolin kautta, julkisesta osoitteesta, tai ohjelmallisesti Amazonin kehitystyökaluilla tehdyillä sovelluksilla. Muokkaus- ja hallinta-ominaisuudet ovat käytettävissä vain konsolissa ja koodista. (Amazon Web Services 2019i.)

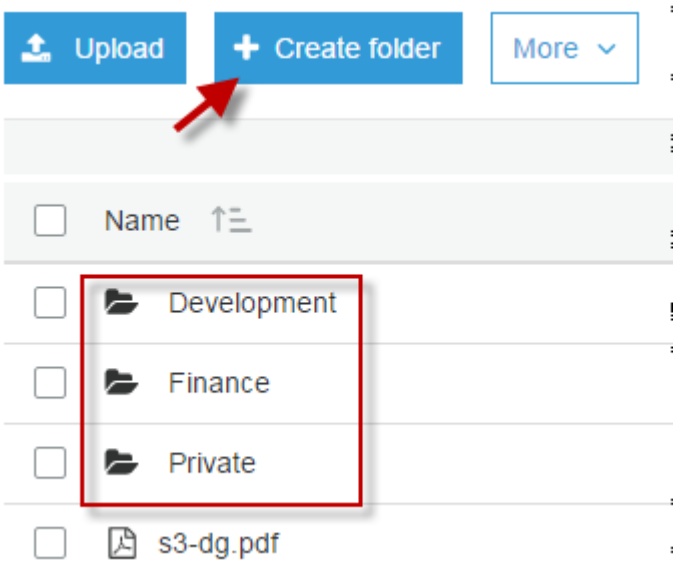
```
http://bucket.s3-aws-region.amazonaws.com

http://s3-aws-region.amazonaws.com/bucket
```

Kuva 1. S3 Bucketin virtuaali- ja polku-osoitetyypit (Amazon Web Services 2019i)

Bucketin julkinen osoite koostuu bucketin ja domainin nimestä sekä tallennetun objektin sijainnista. Osoitteen muoto voi olla virtuaali- tai polku-tyyppinen, joiden erona on bucketin nimen sijainti osoitteessa. Kuvassa (kuva 1) ylempi osoite on virtuaali-tyylinen eli bucketin nimi on osana palvelimen nimeä, ja alempi on polku-tyylinen, jossa bucketin nimi on URL-polku muuttujana, eli palvelimen nimen jälkeen. (Amazon Web Services 2019i.)

2.1.2 Objekti

S3-objekti on korkealla tasolla avain-arvo pari, joka voi sjaita useassa bucketissa samanaikaisesti. Datan ja avaimen lisäksi objekti sisältää myös käyttäjän ja järjestelmän asettamia metatietoja, sekä siihen voidaan liittää aliresursseja ja käytönhallintasääntöjä. Objektin koko voi olla maksimissaan 5 teratavua, mutta minimikokoa sillä ei ole. (Amazon Web Services 2019l.)



Kuva 2. Kansio rakenne S3 Bucketin sisällä (Amazon Web Services 2019m)

Tallennettu data identifioidaan käyttäen avainta, eli bucketin osoitetta, sijaintia bucketissa sekä objektin nimeä ja versio id:tä (Amazon Web Services 2019l). Todellisuudessa objekteilla ei ole erillistä sijaintia bucketin sisässä, vain uniikki avain. Antamalla avaimelle etuliitteitä voidaan luoda bucketin sisälle kuvitteellisia tasoja, eli kansioita (Amazon Web Services 2019m). Kuvan (kuva 2) S3 Bucketissa on kolme etuliitettä, Development,

Finance ja Private, jotka visualisoidaan kansioina, sekä objekti nimeltä s3-dg.pdf ilman etuliitettä.

Objektikohtaisesti määritettäviä metatietoja asetetaan, kun objekti lisätään Buckettiin, mutta kaikkia niistä ei käyttäjä voi sen jälkeen enää muokata. Alla olevassa taulukossa (taulukko 1) on kaikki järjestelmän asettamat metatiedo. Ensimmäisessä sarakkeessa on on metatiedon nimi, kuten päivämäärä. Seuraavassa sarakkeessa kuvataan metatiedon merkitystä ja viimeisessä sarakkeessa käyttäjän oikeudet muokata kyseistä tietoa. Näitä tietoja käytetään järjestelmän puolesta objektien hallintaan, esimerkiksi hinnoitteluun varastoluokkia asetettaessa. Käyttäjän on myös mahdollista asettaa kustomoituja avain-arvo pareja metatietoihin. (Amazon Web Services 2019m.)

Taulukko 1. S3 objektin järjestelmän asettamat metatiedot (Amazon Web Services 2019m)

Name	Description	Can User Modify the Value?
Date	Current date and time.	No
Content-Length	Object size in bytes.	No
Last-Modified	Object creation date or the last modified date, whichever is the latest.	No
Content-MD5	The base64-encoded 128-bit MD5 digest of the object.	No
x-amz-server-side-encryption	Indicates whether server-side encryption is enabled for the object, and whether that encryption is from the AWS Key Management Service (SSE-KMS) or from AWS managed encryption (SSE-S3). For more information, see Protecting Data Using Server-Side Encryption.	Yes
x-amz-version-id	Object version. When you enable versioning on a bucket, Amazon S3 assigns a version number to objects added to the bucket. For more information, see Using Versioning.	No
x-amz-delete-marker	In a bucket that has versioning enabled, this Boolean marker indicates whether the object is a delete marker.	No
x-amz-storage-class	Storage class used for storing the object. For more information, see Amazon S3 Storage Classes.	Yes
x-amz-website-redirect-location	Redirects requests for the associated object to another object in the same bucket or an external URL. For more information, see (Optional) Configuring a Webpage Redirect.	Yes
x-amz-server-side-encryption-aws-kms-key-id	If x-amz-server-side-encryption is present and has the value of `aws:kms`, this indicates the ID of the AWS Key Management Service (AWS KMS) master encryption key that was used for the object.	Yes
x-amz-server-side-encryption-customer-algorithm	Indicates whether server-side encryption with customer-provided encryption keys (SSE-C) is enabled. For more information, see Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C).	Yes

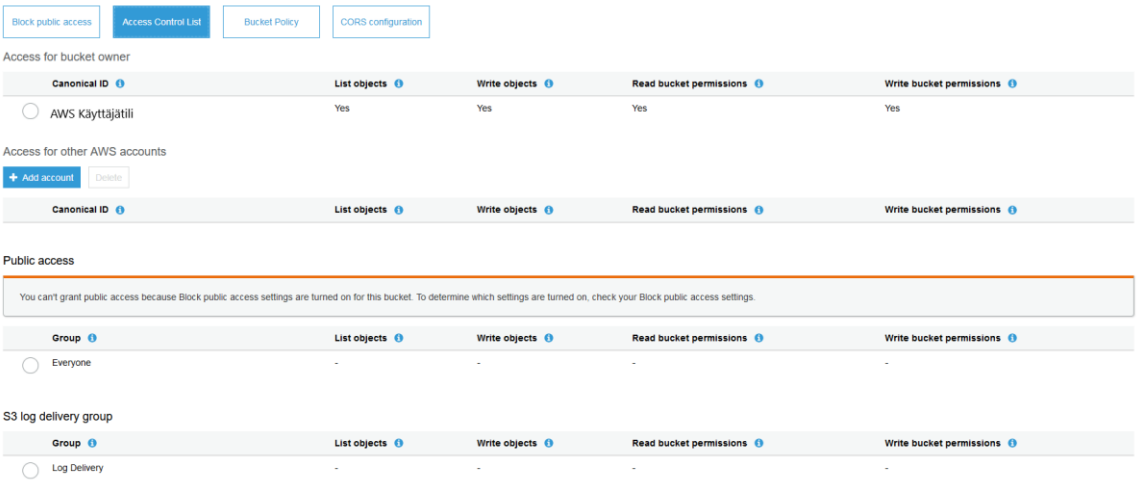
2.1.3 Pääsynhallinta

Bucketin ja sen objektien käyttöä voidaan hallita muokkamalla pääsynhallintalistoja (ACL) tai asettamalla JSON-muotoisia menettelysääntöjä (Bucket Policy) (Amazon Web Services 2019i). JSON eli Javascript Object Notation on ECMAscript-ohjelmointikielestä jalkautunut tekstipohjainen kieliriippumaton dataformaaatti (Ecma International 2017, 1). Menettelysäännoillä on mahdollista asettaa suppeampia oikeuksia kuin listoilla, mikä on nähtävissä seuraavassa taulukossa (taulukko 2). Taulukon ensimmäisessä sarakkeessa on pääsynhallintalistaan määritettävä käyttöoikeus, ja muissa sarakkeissa vastaavat oikeudet menettelysääntöinä.

Taulukko 2. S3 bucketin käyttöoikeuksien vertailu (Amazon Web services 2019k)

ACL permission	Corresponding access policy permissions when the ACL permission is granted on a bucket	Corresponding access policy permissions when the ACL permission is granted on an object
READ	`s3:ListBucket`, `s3:ListBucketVersions`, and `s3:ListBucketMultipartUploads`	`s3:GetObject`, `s3:GetObjectVersion`, and `s3:GetObjectTorrent`
WRITE	`s3:PutObject` and `s3:DeleteObject`. In addition, when the grantee is the bucket owner, granting `WRITE` permission in a bucket ACL allows the `s3:DeleteObjectVersion` action to be performed on any version in that bucket.	Not applicable
READ_ACP	`s3:GetBucketAcl`	`s3:GetObjectAcl` and `s3:GetObjectVersionAcl`
WRITE_ACP	`s3:PutBucketAcl`	`s3:PutObjectAcl` and `s3:PutObjectVersionAcl`
FULL_CONTROL	Equivalent to granting `READ`, `WRITE`, `READ_ACP`, and `WRITE_ACP` ACL permissions. Accordingly, this ACL permission maps to a combination of corresponding access policy permissions.	Equivalent to granting `READ`, `READ_ACP`, and `WRITE_ACP` ACL permissions. Accordingly, this ACL permission maps to a combination of corresponding access policy permissions.

Pääsynhallintalistoilla määritetään AWS käyttäjätilejä tai ryhmiä sekä käyttöoikeuden tyyppejä. Käyttöoikeuksiin kuuluu bucketin luku, kirjoitus, käyttöoikeuksien luku ja kirjoitus, sekä täysihallinta -oikeudet. Asetukset voidaan määrittää selainnäkymässä, tai ohjelmallisesti XML-formaatissa REST-rajapinnan kautta (kuva 3). (Amazon Web services 2019k.)



Kuva 3. S3 Bucket käytönhallinta listanäkymä konsolissa

2.2 Lambda

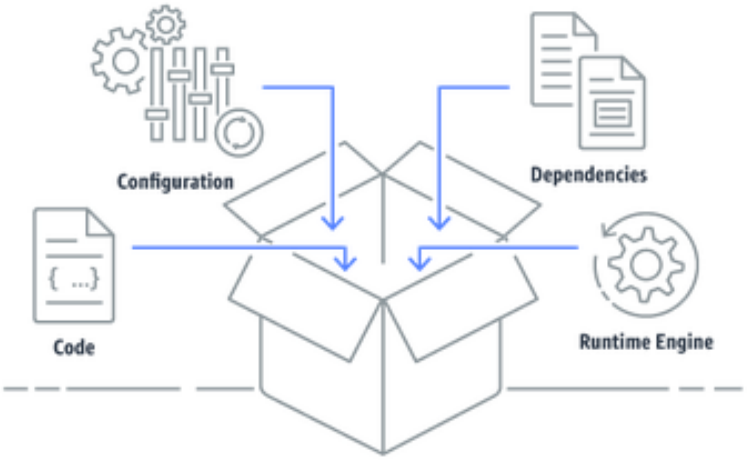
AWS Lambda on Serverless-arkkitehtuuriin perustuva palvelu, jossa ajetaan käyttäjän lataamaa koodia vain silloin kuin sitä tarvitaan (Amazon Web services 2019n). Käyttömaksut perustuvat funktion ajokertoihin ja suoritusten kestoon, tehden siitä kustannustehokkaan (Amazon Web Services 2019o, 2). Lambdan käyttö helpottaa kehittäjän työtä, koska palvelimen ja alustan ylläpito, sekä ajettavan koodin monitorointi tehdään automaattisesti (Amazon Web Services 2019o, 1).

2.2.1 Suoritusympäristö

Lambdan toiminnallisuus jaetaan hallinta- ja datatasoihin, joista käyttäjä on tekemisissä vain hallintatason kanssa. Hallintatasolla voidaan luoda, päivittää, sekä poistaa Lambda funktioita, asetuksia, sekä metatietoja käyttäen sen tarjoamaa rajapintaa (Amazon Web Services 2019p; Amazon Web Services 2019o, 5). Lambda funktion käynnistysrajapinta sijaitsee datatasolla, missä myös koodin vaatima ympäristö muodostetaan ja varataan kutsujalle. (Amazon Web Services 2019o, 5.)

Funktion ympäristö koostuu käyttäjän lataamasta koodista, minimalistisesta Linux-käyttöjärjestelmästä, koodin vaatimasta ohjelmointikielikohtaisesta alustasta sekä mahdollisista Lambda kerroksista (Amazon Web Services 2019o, 6). Ympäristöt ovat funktiokohtaisia ja suorittavat vain yhtä kutsua kerrallaan kunnes funktion elinkaari loppuu. Mikäli samaa funktiota kutsutaan useasti lyhyen aikavälin sisällä, voidaan ympäristöä uudelleenkäyttää, mikä nopeuttaa funktion suorittamista (Amazon Web Services 2019q; Amazon Web Services 2019o, 5).

Ympäristöjä ajetaan AWS käyttäjätilikohtaisien virtuaalikoneiden sisällä, missä voi sijaita useiden eri funktioiden ympäristöjä. Ajon aikana ympäristöt erotetaan toisistansa käyttäen Linuxin sisältämiä kontitus-teknologioita kuten cgroupsia (Amazon Web Services 2019o, 6). Ohjelmistokontti koostuu sovelluksen koodista ja sen vaatimista resursseista (kuva 4). Kontit jakavat myös käyttöjärjestelmän keskenään, tehostaen niiden ajamista ympäristöstä riippumatta (Amazon Web Services 2019r).



Kuva 4. Ohjelmistokontin sisältö (Amazon Web Services 2019r)

2.2.2 Serverless Framework

Serverless Framework on avoimen lähdekoodin ohjelmistokehys, joka mahdollistaa Serverless-ohjelmistojen rakentamisen usealle eri pilvialustalle. Se sisältää komentokehote-rajapinnan, jonka avulla Serverless-sovelluksia voidaan kehittää, testata ja ladata pilvipalveluun. Serverless tarjoaa myös web-käyttöliittymän, jota voidaan käyttää Serverless-ohjelmistojen ylläpitämistä ja tarkkailua varten. (Serverless 2019a.)

Amazon Web Services Lambdoja käytettäessä Serverless-sovellukset koostuvat palveluista (service), jotka ovat yksittäisiä projekteja. Nämä projektit voivat koostua useasta Lambda-funktiosta mahdollistaen suurempienkin ohjelmistoprojektien moduloimisen pienempiin osiin. Jokaiselle palvelulla on oma serverless.yml-tiedosto, jossa määritetään kaikkien sen Lambdojen ympäristöasetukset. (Serverless 2019b.)

Serverless.yml kuvailee koko palvelun rakenteen ja sen tulee sisältää kaikki vaaditut resurssit. Tiedoston avulla komentorivityökalu luo kaikki AWS-resurssit itsenäisesti ja lataa sovelluksen pilvipalveluun. (Serverless 2019c.) Jokaiselle palvelulle määritetään sen ympäristön rakenne, kuten käytetty NodeJS versio, varatun muistinmäärä tai AWS-käyttöalue. Kaikille funktioille voidaan antaa myös yhteisiä asetuksia, kuten ympäristömuuttujia. (Serverless 2019c; Serverless 2019d.) Useimmilla asetuksista on oletusarvoja, joten kaikkia asetuksia ei tarvitse itse määrittää (Serverless 2019c). Muita AWS-resursseja asetettaessa tulee myös määrittää vaaditut käyttöoikeudet, jotta funktioden kommunikointia muihin resursseihin ei estetä (Serverless 2019d).

Asetustiedostossa määritetään myös käytössä olevat funktiot sekä niiden toimintaan vaikuttavat asetukset. Funktiolle voidaan asettaa laukaisutapa (trigger), esimerkiksi http-kutsu ja sen polku, sen sijainti projektitiedostoissa, sekä myös ympäristöön liittyviä vaatimuksia (Serverless 2019d). Ympäristön määrittäminen funktiokohtaisesti ohittaa palvelulle määritetyt parametrit, mikäli sama asetus on laitettu molempiin (Serverless 2019c.)

Komentokehote-rajapinta mahdollistaa toimintoja, joiden avulla voidaan hallita palveluita tai funktiota. Komennolle voidaan määrittää taso jolla se toimii, kuten kehitys tai tuotanto, sekä toiminta-alue Amazonissa (Serverless 2019e). Yleisimmin käytetään deploy- ja invoke-komentoja AWS Lambdaa kehitettäessä.

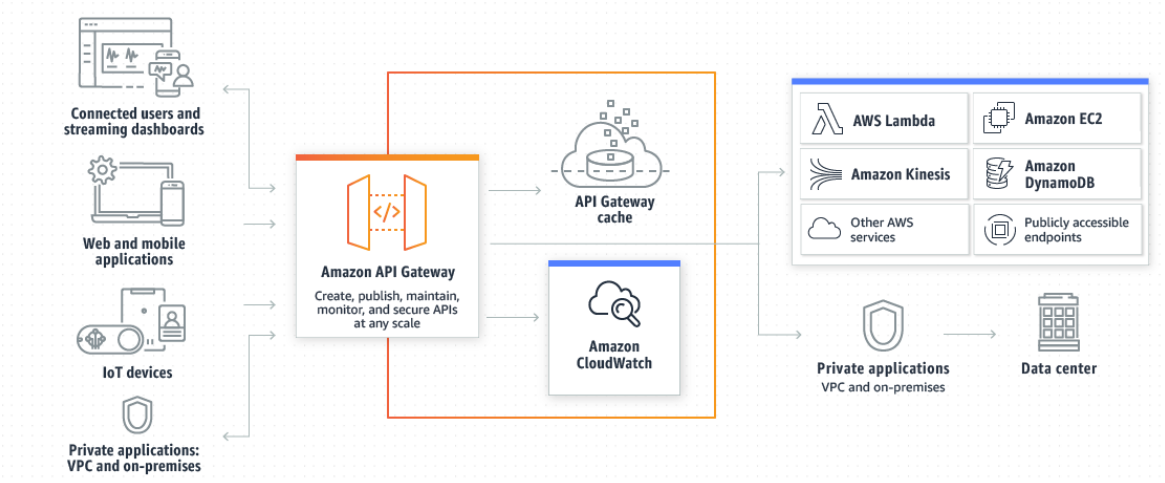
Valmis funktio tai palvelu on mahdollista ladata automaattiseti pilvipalveluun käyttäen deploy-komentoa. Deploy ensin paketoi koko palvelun koodin ja sen tarvitsemat kirjastot, ja sitten lataa sen AWS CloudFormation-palvelua käyttäen. Tarvittaessa on mahdollista ladata yksi funktio kerrallaan palveluun, mikä on nopeampaa.Tässä tapauksessa palvelun

rakenteeseen tehdyt muutokset eivät päivity serverless.yml tiedostosta. (Serverless 2019e.)

Funktiota voidaan myös ajaa komentokehotteella invoke-komentoa käyttäen. Näin funktiota voidaan testata itse määritetyillä syötteillä, tulostaen komentokehotteelle ajon tulokset. Komennolle voidaan myös määrittää paikallisuus-lippu, jolloin kutsu tapahtuu paikallisesti. (Serverless 2019f.)

2.2.3 API Gateway

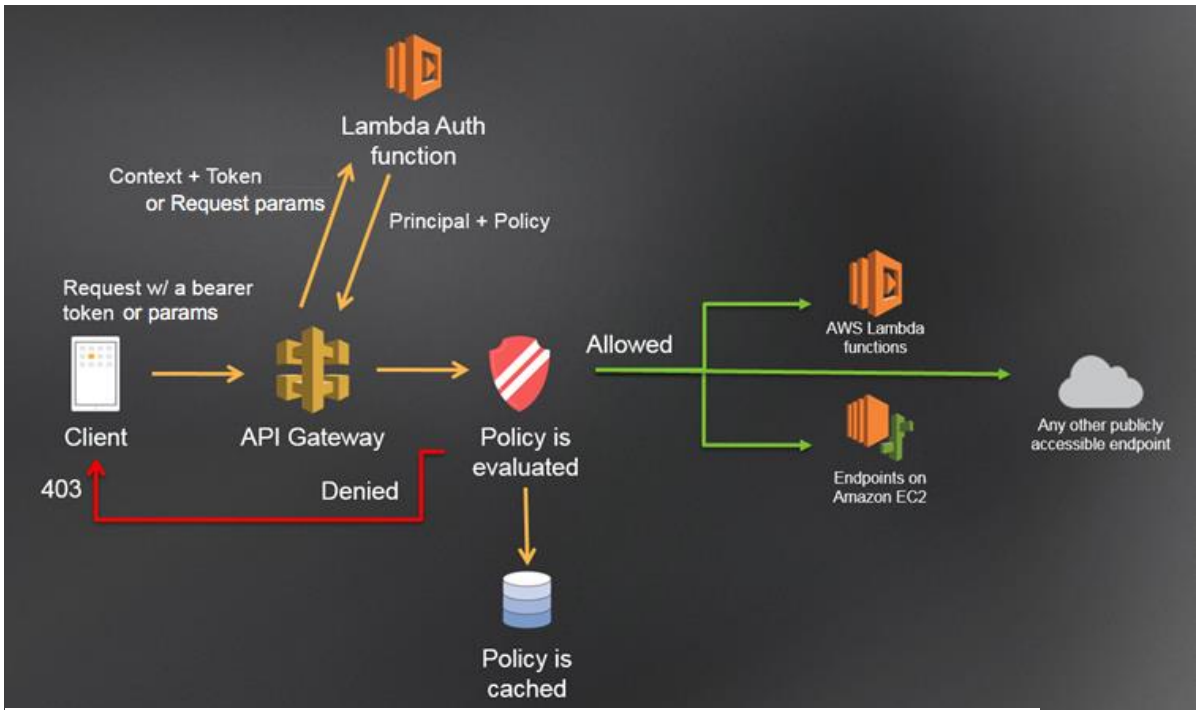
AWS API Gatewayn avulla on mahdollista luoda ja ylläpitää REST- sekä Websocket-rajapintoja AWS-pilvipalvelun infrastruktuurissa. Nämä rajapinnat pystyvät vastaanottamaan satoja tuhansia samanaikasia kutsuja sekä ohjaamaan liikennettä AWS-palveluihin, kuten Lambda funktioihin sekä AWS:n ulkopuolelle. (Amazon Web Services 2019u.) Alla oleva kuva (kuva 5) esittää API Gatewayn luoman rajapinnan rakenteen, sekä mahdollisuudet laajentaa sen toiminnallisuutta hyödyntämällä AWS CloudWatchia tai tallentamalla tietoja välimuistiin.



Kuva 5. AWS API Gateway arkkitehtuurin rakenne (Amazon Web Services 2019u)

API Gatewayn kutsurajapintaan saapuvien pyyntöjen käyttöoikeuksia voidaan hallita tunnistamalla käyttäjä useilla eri menetelmillä. Perinteisten alfanumeeristen API-avainten lisäksi voidaan määrittää IAM-, sekä resurssikohtaisia-sääntöjä ja Lambda-funktioita käyttäjän tunnistamiseen. On myös mahdollista määrittää miten rajapinta käyttäytyy erilaisten vastaanotettujen kutsujen kanssa. (Amazon Web Services 2019x.)

Lambdalla tunnistautuessa myöntäjäfunktiot (authorizer function) tukevat merkkijono (token)-tyyppisiä tunnisteita, kuten JWT- tai OAuth-teknologioita tai kutsu parametreissa sisällytettyjä tietoja. Kutsun saapuessa API Gatewaylle ajetaan määritetty Lambda-funktio käyttäjän antamilla syötteillä, mikä palauttaa kyseiselle käyttäjälle määritetyt käyttöoikeudet. Näitä oikeuksia verrataan resurssin vaatimiin oikeuksiin, jolloin kutsu hyväksytään ja käyttäjän tunnistautuminen tallennetaan välimuistiin, tai API Gatewayn kutsujalle palautetaan HTTP-tilakoodi 403. (Amazon Web Services 2019v; kuva 6.)



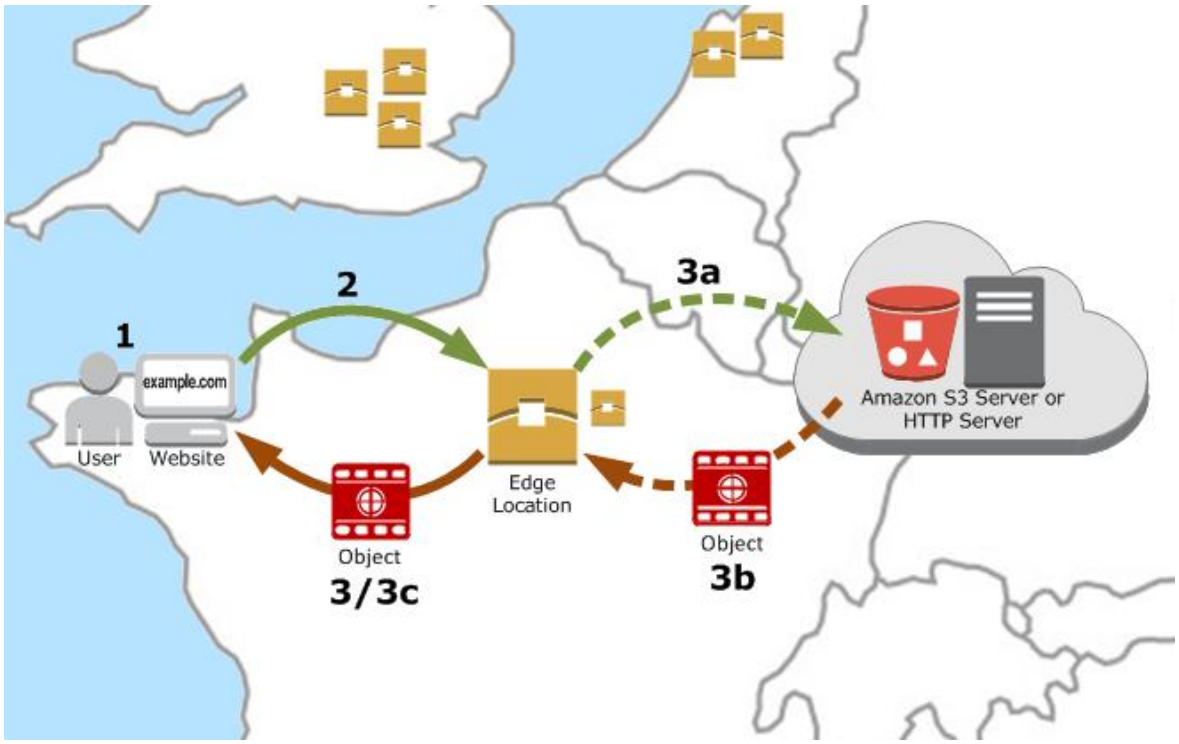
Kuva 6. API Gateway Lambda myöntäjä funktion tunnistautumisen vaiheet (Amazon Web Services 2019v)

2.3 CloudFront

CloudFront on sisällön jakeluverkko, jolla voidaan nopeuttaa dynaamisen ja staattisen Web-sisällön jakamista loppukäyttäjille. Se perustuu laajaan verkostoon datakeskuksia, joiden avulla resurssit voidaan noutaa nopeinta mahdollisinta reittiä pitkin. CloudFront sisältää myös oman osoitteensa jakamilleen resursseille, mikä voidaan halutessa vaihtaa myös DNS-nimeksi. (Amazon Web Services 2019y.) Tähän voidaan käyttää AWS Route53 DNS-palvelua tai muuta DNS-palvelinta.

Sisältö jaetaan käyttäen siihen tarkoitettuja datakeskuksia eri raja-alueilla (edge-locations). Resursseja noudettaessa käytetään lähintä pistettä, josta haluttu sisältö palautetaan, jos se on välimuistissa. Kutsun saapuessa CloudFrontin pisteeseen se ohjataan pisteeltä toiselle. Jos resurssia ei löydy, noudetaan data CloudFrontin

määrittämästä sijainnista, tallennetaan välimuistiin tulevaisuutta varten ja palautetaan (kuva 7). (Amazon Web Services 2019z.) Tätä jakeluverkkoa käyttämällä loppukäyttäjän ei tarvitse tietää kuin osoite, josta sisältöä voidaan pyytää ja data saapuu nopeinta reittiä pitkin. Verkoston avulla data on tallessa useassa eri paikassa, tarjoten nopean käyttökokemuksen myös tuleviin kutsuihin paikasta riippumatta. Sisällön jakamisen luotettavuus ja saatavuus myös tehostuu, kun sama data on kopioitu useaan eri paikkaan (Amazon Web Services 2019y).



Kuva 7. CloudFront jakeluverkoston toimintakuvaus (Amazon Web Services 2019y)

3 ANGULAR 2+

Angular 2 on uudelleenkirjoitettu versio alkuperäisestä Angular 1.x-ohjelmointikehyksestä, minkä tavoitteena nopeuttaa sillä tehtyjä sovelluksia ja parantaa niiden responsiivisuutta. Se tukee ES2015 Javascript-standardia sekä Typescript-laajennusta, mutta toimii myös vanhemmilla standardeilla sekä Dart- ja CoffeeScript-skriptikielillä. (Deeleman 2016, ix.) Angular 2 hyödyntää myös uusimpia HTML5-standardin ominaisuuksia, kuten Shadow DOM-tekniikkaa (Deeleman 2016, 1).

3.1 Modulaarinen Rakenne

Angular mahdollistaa sovelluksen moduloinnin pienempiin kokonaisuuksiin moduulien, palveluiden ja komponenttien avulla. Nämä ovat Javascript luokkia, joille on määritetty funktio, joka muokkaa sen toimintaa annettujen parametrien perusteella. Esimerkiksi komponenteille ennen luokkaa asetetaan @Component-funktio. Näitä luokkia tulee olla juuritasolla vähintää yksi, jotta sovellus voidaan käynnistää. (Angular 2019a.)

3.1.1 Moduulit

Moduulit jakavat applikaation toiminnallisuutta korkealla tasolla säilömällä komponentteja, palveluita tai kolmannen osapuolen kirjastoja. Moduulitiedostossa määritetään sen sisällä käytetyt sekä jaetut resurssit ja riippuvuudet. Nämä metatiedot ovat moduulikohtaisia, joten niiden asettaminen ei vaikuta muihin projektin osiin. Applikaatiot voivat sisältää useita ominaisuusmoduuleja, jotka sidotaan yhteen juurimoduulilla. (Angular 2019g.)

Ominaisuusmoduuleja suositellaan käyttämään, jotta sovelluksen juurimoduuli ei kasva liian isoksi (Angular 2019h). Näin vältytään epäselvyyksiltä koodin kehityksessä sekä ylläpidossa, sillä tietty moduuli sisältää vain tietyn ominaisuuden. Mahdollisia virheitä on myös helpompi selvittää, jos tietty osa koodia on vastuussa vain tietystä tehtävästä, kuten datan hausta, mutta ei sen manipulaatiosta. Ylimääräinen logiikka voidaan hakea toisesta moduulista luoden selkeät rajat moduulin tehtäville. Kaikki ominaisuusmoduulit tulee määrittää juurimoduulissa, jotta ne ovat käytettävissä sovelluksessa. (Angular 2019h.)

Moduloinnin ansiosta on myös mahdollista nopeuttaa applikaation toimintaa. Oletusarvoisesti Angular-sovelluksissa kaikki moduulit ja niiden sisällöt ladataan heti kun käyttäjä saapuu sivulle. (Angular 2019i.) Tässä tapauksessa käyttäjä joutuu lataamaan palvelun osia, joita ei välttämättä tarvitse, pidentäen ensimmäistä latauskertaa. Reititysmoodulien avulla on mahdollista toteuttaa viivästetyn lataamisen (lazy loading)-suunnittelumallia, jolloin ominaisuuksia ladataan vain silloin kuin niitä tarvitaan.

Ominaisuusmoduulin määrittämisen sijaan juurimoduulille annetaan reitti, jota kutsumalla ominaisuusmoduuli ladataan. (Angular 2019i.) Näin vähennetään ensimmäisten ladattavien tiedostojen kokoa, mikä nopeuttaa sovelluksen käynnistymistä. Viivästettyä lataamista voidaan käyttää myös ominaisuusmoduulin sisällä, sillä moduulien rakenne ei eroa metatietoja lukuunottamatta, nopeuttaen tuleviakin latauksia (Angular 2019h).

3.1.2 Komponentit

Angular 2 komponentit ovat uudelleenkäytettäviä osia sovelluksessa, jotka sisältävät oman graafisen ilmeen sekä logiikan (Kaufman 2016, 4). Yhdistämällä komponentteja muodostetaan suurempia kokonaisuuksia sovelluksesta. Angular 2-sovellus on korkealla tasolla vain komponenttipuu, jonka tulee vähintään koostua yhdestä komponentista. (Kaufman 2016, 5; Angular 2019a.)

Komponentit noudattavat yleistä MVC-arkkitehtuuria datan esittämiseen. MVC eli Model View Controller jakaantuu kolmeen osaan, jotka keskustelevat keskenään ja hoitavat omia tehtäviään. Model eli malli sisältää itse datan, jota halutaan näyttää. View eli näkymä on vastuussa datan esittämisestä ja Controller eli ohjain toimii välikappaleena näkymän ja mallin välillä. (Kaufman 2016, 2.) Angularin tapauksessa ohjain ja malli osuus on usein sulautettu samaan luokkaan, mutta malli voi myös sijaita omassa palvelussaan (service). (Angular 2019a.)

Dataa on mahdollista syöttää komponenttiin asettamalla sen HTML-elementtiin attribuutti (Angular 2019b). Jotta dataa voidaan käyttää, tulee komponentin luokassa olla määritelty samalla nimellä attribuutti käyttäen datatyyppiä @Input. Syötettäessä monimutkaisempaa dataa kuten objekteja, tulee näkymässä oleva attribuutti sijoittaa hakasulkujen sisälle. (Angular 2019c; Kuva 8.)

```
@Component({
    selector: 'app-example',
})
export class ExampleComponent {
    @Input() property: number | string;
    @Output() propertyChange = new EventEmitter<number>();

}
@Component({
    selector: 'app-parent',
    template: `
        <app-example [(property)]="parentProperty"
            (propertyChange)="updateProperty($event)" >
        </app-example >
    `
})
export class ParentComponent {
    parentProperty: number;
    @Output() updateSomeProperty = new EventEmitter<number>();
    updateProperty(event) {
        this.updateSomeProperty.emit(event);
    }
}
```

Kuva 8. Angular 2 komponenttien välinen datansiirto

Syötettyyn attribuuttiin on myös mahdollista liittää muutos tapahtumankuuntelija sijoittamalla attribuutti sulkujen sisälle. Tapahtuma-attribuutin nimi tulee koostua syötetyn attribuutin nimestä sekä sanasta Change eli muutos. (Angular 2019c.) Yllä olevassa kuvassa (kuva 8) ExampleComponent-komponentin property-arvon muuttuessa, kutsutaan automaattisesti propertyChange tapahtuma-attribuuttiin liitettyä updateProperty-funktiota.

Hakasuluilla ja normaaleilla suluilla [()] merkitty kaksisuuntainen datasidos samanaikaisesti päivittää myös parentProperty-muuttujan (kuva 8). Kaksisuuntaisen datasidoksen avulla on mahdollista uudelleenkäyttää ylemmän tason komponentin hakemaa dataa. Komponentteja voidaan tehdä yleispätevimmiksi, sillä niiden ei tarvitse kuin vastaanottaa ja käsitellä dataa. Lopulta muutokset syötetään tapahtuma-attribuutilla määritetylle funktiolle tai muuttujalle, välittämättä mistä data on peräisin.

3.2 Tyylittely

Angular sovellusten ulkonäön tyylittelyä on mahdollista tehdä käyttäen perinteistä CSS-tyylisivukieltä (Angular 2019f). CSS eli Cascading Style Sheets -kieltä käytetään HTML tai XML-dokumenttien ulkonäön muokkaamiseen kuvaamalla miten elementit piirretään

ruudulle, paperille tai muulle medialle (Mozilla 2019h). CSS-kielen lisäksi Angular tukee LESS, SASS ja Stylus -tyylikehyksiä (Angular 2019f).

Oletusarvoisesti sovelluksessa on styles-tiedosto, jonka sisältämät tyylit ovat käytettävissä koko sovelluksessa. Tyylittely on myös mahdollista pilkkoa pienempiin osiin sitomalle ne tiettyihin komponentteihini. Näin kirjoitetut tyylit liitetään vain tietyn komponentin elementteihin, eivätkä ne periydy muille komponenteille. Tämän ansiosta valitsijat (selectors) voidaan nimetä komponentin kontekstin mukaisesti, aiheuttamatta ristiriitoja muiden tyylien välillä. (Angular 2019f.) Kehittäjän ei tarvitse myös huolehtia muiden kokonaisuuksien tyylien tuhoamisesta tai riippuvuuksista, vaan muutokset voi kohdistaa yhteen alueeseen.

Angularille on myös tehty oma tyylikirjasto Angular Material, joka sisältää valmiiksi tehtyjä komponentteja, ikoneja, rajapintoja, sekä työkaluja. Se noudattaa Googlen kehittämän Material Design-tyylikielen ohjeistuksia, jonka tavoitteena on luoda yhtenäinen käyttökokemus eri alustoilla ja laitteilla. (Deeleman & Noring 2017, 276-278.)

Angularin suositeltujen tapojen mukaisesti, Angular Materialin tuomat ominaisuudet on sisällytetty omiin moduuleihinsa, jotka tulee lisätä applikaation riippuvuuksiin, jotta niitä voidaan käyttää. Lisäämällä HTML-elementteihin kirjaston attribuutteja, liitetään valmiiksi tehtyjä tyylejä automaattisesti elementtiin. (Deeleman & Noring 2017, 282.) Näitä tyylejä käyttämällä applikaation ulkonäkö saadaan yhtenäiseksi.

Teemat koostuvat useista väripaleteista, jotka asetaan Angular Materialin komponenteille. Paletit on jaettu värin tarkoituksen perusteella kuten virhe, tausta tai pää -väreihin. (Angular 2019j.) Näin tietyn tyyppisissä taustoissa tai sivun elementin osissa käytetään samaa väriä. Tällä tavalla luodaan selkeä rakenne sivuston tyyleihin. Teemojen määrittämisessä hyödynnetään SASS-tyylikehyksen mixineitä, joilla voidaan sitoa useita eri tyylejä yksittäiseen muuttujaan (Angular 2019j).

3.3 Eri alustojen tukeminen

Angular -ohjelmistokehys on tarkoitettu toimimaan uusimmilla selaimilla, käyttäen uusimpia web-teknologioita. Seuraavassa taulukossa (taulukko 3) on listattu Angularin tukemat selaimet sekä alustat, joilla sovelluksien tulisi toimia. Uusimpien standardien kehityksen mukana kulkeminen vaikeuttaa laajan selaintuen ylläpitämistä, sillä kaikki selaimet eivät näitä ominaisuuksia tue. Tästä syystä Angularissa on tuki useille työkaluille, joilla muokataan koodia eri selaimille, ja selainversioille sopivaksi. Angularin kehittäjät myös hyödyntävät selainkohtaisia yksikkötestejä versionhallinnassaan löytääkseen yhteensopivuus poikkeuksia. (Angular 2019d.)

Taulukko 3. Angular -ohjelmistokehyksen tuetut selaimet ja alustat (Angular 2019d)

Browser	Supported versions
Chrome	latest
Firefox	latest
Edge	2 most recent major versions
IE	11, 10, 9 ("compatibility view" mode not supported)
IE Mobile	11
Safari	2 most recent major versions
iOS	2 most recent major versions
Android	Nougat (7.0), Marshmallow (6.0), Lollipop (5.0, 5.1), KitKat (4.4)

3.3.1 Selaimet

Vanhempien selaimien tuki on mahdollista saavuttaa polyfill-tiedostoilla, jotka liitetään projektin tiedostojen mukaan. Polyfillit ovat ylimääräistä koodia, joilla luodaan moderneja ominaisuuksia vanhempiin selaimiin, jotka eivät niitä vielä tue. Useimmiten ne ovat Javascriptillä tehtyjä. Polyfillit ovat viimeinen keino saavuttaa haluttu ominaisuus, sillä niiden käyttö ei ole yhtä nopeaa tai monipuolista kuin selainrajapinnan toteuttamana (Mozilla 2019c).

Angularissa polyfillejä tuetaan natiivisti käyttäen polyfills.ts-tiedostoa, jos projekti on luotu käyttämällä komentokehotetyökalua. Tässä tiedostossa määritetään kaikki tarvittavat skriptit, joilla haluttu selaintuki saavutetaan. Mikäli työkalua ei ole käytetty, voidaan halutut tiedostot määrittää index.html-tiedostoon, mitä voidaan käyttää myös projekteissa, joissa Angularia ei ole käytetty. Loppukäyttäjän navigoidessa web-sivulle määritetyt skriptit ladataan, oli selain vanha tai ei. (Angular 2019d.)

Laajempaa selantukea tavoiteltaessa saattaa ladattavien tiedostojen määrä kasvaa suureksi, jolloin palvelun käyttö hidastuu. Angular versio 8 mukana on tullut ominaisuus, joka luo projektista erillisen version vanhemmille selaimille sopivaksi. Tämä mahdollistaa

pienemmän koon projektin tiedostoille uusimmilla selaimilla, mikä nopeuttaa sivuston lataamista. (Angular 2019e.)

Sovelluksen tiedostojen rakennusvaiheessa komentokehotetyökalu lukee selainlista-tiedostosta tuetut selaimet ja niiden versiot, ja tämän perusteella lisää index.html tiedostoon ladattavat skriptit. Skripteille on määritelty tyyppi-attribuutti ja sille arvo moduuli tai nomodule-tagi. Uudemmat selaimet, jotka tukevat EcmaScript-moduuleja tunnistavat tyyppi-attribuutin, ja lataavat tiedostot ilman polyfillejä. Vanhemmat selaimet lataavat nomodule-tagilla varustetut skriptit, eli selainversiosta riippumatta ladataan vain vaaditut tiedostot, eikä yhtään enempää. (Angular 2019e.)

3.3.2 Laitteet

Teknologian kehittyessä useammat laitteet ovat saaneet tuen selaimille, mutta sivustojen käyttö ei näillä täysin samalla tavalla toimi. Esimerkiksi useimmilla mobiililaitteilla ei ole mahdollista käyttää hiirtä navigoimiseen, vaan tämä on korvattu kosketuksella. Näitä tilanteita varten on kehitetty erillisiä rajapintoja, jotka mahdollistavat eri laite- ja näyttötyypeille keinoja olla vuorovaikutuksessa selaimen kanssa.

Web-sovelluksia käytettäessä useimmiten odotetaan hiiren painalluksia, jotka laukaisevat hiiritapahtuman. Muiden osoittimien yleistymisen myötä on luotu yhtenäien osoitin-rajapinta näiden kaikkien tukemiseksi. Se pohjautuu hiiritapahtumaan, tarjoten sovellukselle pääsyn kosketuksen sijaintiin, kohde elementin ominaisuuksiin ja muihin vastaaviin tietoihin. Rajapinta myös mahdollistaa osoittimen tyypin tunnistamisen, sekä useamman eri osoittimen samanaikaisen käytön. (Mozilla 2019d.)

Toiminnallisuuksien jäljentämisen lisäksi selaimet tukevat myös useita alustakohtaisia ominaisuuksia. Nämä mahdollistavat muunmuassa akullisten laitteiden varaustietoihin pääsyn, peliohjainten signaalien vastaanoton tai mobiililaitteiden vibraation aktivoimisen (Mozilla 2019e; Mozilla 2019f; Mozilla 2019g). Hyödyntämällä näitä rajapintoja voidaan antaa käyttäjälle interaktiivisempi käyttökokemus. Kaikki rajapinnat eivät kuitenkaan ole vielä valmiita, joten niitä ei suositella käytettäväksi tuotantoympäristössä (Mozilla 2019g).

4 TAUSTAJÄRJESTELMÄ

Palvelun taustajärjestelmän teknologioissa päädyttiin käyttämään NodeJS-ajoympäristöä, sekä Amazon Web Services -pilvipalvelua tukemaan tätä. Taustajärjestelmä oli vastuussa raskaamman data manipulaation ja laskennan suorittamisesta, jotta käyttöliittymä tulisi toimimaan sulavasti vanhemmillakin laitteilla ja alustoilla. Järjestelmä toteutti myös REST-arkkitehtuuria, jotta sitä voidaan tarvittaessa laajentaa.

4.1 Kuvan manipulointi

Kuvanmuokkaus ja konvertointi totetutettiin NodeJS-ajoympäristössä, jota laajennettiin Typescript-kirjastolla. Tämä mahdollisti koodin tyypittämisen, mikä helpotti kehitysvaihetta vähentämällä ohjelmointivirheiden määrää. Koodista tuli myös luettavampaa, sillä käytetty tekstieditori ilmoitti, minkä tyyppistä dataa valittu funktio palauttaa.

4.1.1 Asettelu ja rajaus

Totetutuksessa käytettiin sisäänrakennettujen NodeJS-moduulien lisäksi julkisia Javascript-kirjastoja tarvittavien ominaisuuksien luomiseksi. Kuvanmuokkaus kirjastoa valittaessa testattiin useita eri ratkaisuja, joista nopein ja riippumattomin oli Sharp-kirjasto. Sharp-kirjaston avulla kuvan kokoa, sijaintia sekä asentoa muokataan käyttäjän antamien metatietojen mukaisesti. Kuvaa käsitellään Buffer-formaatissa, jotta väliaikaiseen konvertointiin ei kulutettaisi aikaa.

Kuvan rajauksessa käytetään sen vasemman ja yläreunan sijaintia paneelin keskipisteestä, sekä sen hetkisiä mittoja. Jos reuna on paneelin ulkopuolella, kuvasta leikataan ylimenevän osan verran pois. Kuvan lopulliset mitat saadaan lisäämällä kuvan sen hetkiset mitat reunojen sijainteihin. Jos summasta saatu arvo on enemmän kuin paneelin leveys tai korkeus, rajoitetaan korkeus niihin mittoihin. Muussa tapauksessa käytetään käyttäjän antamia mittoja.

Kuvalle lisätään myös hitsauskoneen paneelin kokoinen musta tausta, jotta lopullisen kuvan suhteet pysyisivät samana kuvan koosta ja asennosta riippumatta. Lopulta kuva formatoidaan RGBA888-pikselidataksi. Jokaisella pikselillä on siis 4 tavua jotka kuvaavat sen punaisen (R), vihreän (G), sekä sinisen (B) värin määriä ja läpinäkyvyyttä (A).

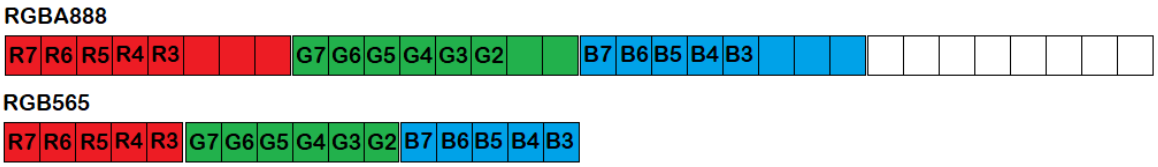
4.1.2 Konvertointi hitsauslaitetta varten

Visuaalisten muokkausten lisäksi kuvadatan rakennetta tuli muuttaa loppulaitteelle sopivaksi. Sharp-kirjastolla muokatut kuvat ovat RGBA888-formaatissa, mikä tarkoittaa että jokainen kuvan pikseli varaa 32-bittiä värin määrittämiseen, eli 8 jokaista väriä ja pikselin läpinäkyvyyttä varten. Tämä on käyttötarkoitukseen nähden tarpeettoman paljon, joten värimaailmaa päädyttiin vähentämään 16-bittiin, eli käyttämään RGB565-formaattia.

```
data.forEach((val, i, arr) => {
    switch (colors[color]) {
        case 'red':
            RGB565[index] = ((val & 0b11111000) << 8);
            color++;
        break;
        case 'green':
            color++;
            RGB565[index] += ((val & 0b11111100 ) << 3);
        break;
        case 'blue':
            color++;
            RGB565[index++] += (val >> 3);
        break;
        case 'alpha':
            color = 0;
        break;
    }
});
```

Kuva 9. RGBA888 kuvan pikselien värien bittien maskaus ja siirtäminen

Formaatin muutos tapahtuu maskaamalla sekä siirtämällä värien bittejä, eli tiputtamalla vähiten merkitsevien bittien määrää. Muokatun kuvan pikseleitä käydään läpi väri kerrallaan ja sen hetkisestä väristä riippuen bittejä siirretään ja maskataan (kuva 9). 8-bitin väreistä poistetaan maskeilla vähiten merkitsevät bitit, kuten punaisesta viimeiset 3-bittiä maskilla 0b11111000. Lopulta bittejä siirretään omiin ’osioihinsa’ 16-bitin lohkossa. Punainen siirretään 8-bittiä vasemmalle lohkon alkuun, vihreä sen perään ja sininen lohkon loppuun. Siirtämällä sekä maskaamalla täytetään koko 16-bitin väriavaruus, joten RGBA888-formaatin alpha, eli kuvan värien läpinäkyvyys jätetään kokonaan pois (kuva 10).



Kuva 10. Pikselin väribitit RGBA888 ja RGBA565-formaateissa

Hitsauslaitteen paneeli vaatii myös oman formaattinsa, jotta kuva voidaan tulkita oikein. Formaatti on Kempin oma kehittämä, joten sen rakennetta ei tässä opinnäytetyössä voida tarkemmin kuvailla. Lopulta formatoitu kuva palautetaan käyttöliittymälle.

4.2 Amazon Web Services -pilvipalvelun käyttöönotto

Taustajärjestelmän koodia päädyttiin ajamaan Amazon Web Services Lambda -palvelua käyttäen. Se mahdollistaa palvelun helpon laajentamisen sekä vianhaun, sillä jokainen toiminnallinen kokonaisuus, tässä tapauksessa funktio, ajetaan omassa ympäristössään. Funktion sisällä koodia jaettiin omiin kokonaisuuksiinsa, kuten kuvan visuaaliseen manipulointiin, konverointiin sekä värimaailman vähentämiseen. Näin koodi oli luettavampaa, mikä samalla helpotti mahdollista vianhakua.

4.2.1 Lambdan konfigurointi

Koodin ajaminen AWS Lambdassa edellyttää useiden konfiguraatioiden luomista, mikä automatisoitiin käyttäen Serverless Framework -ohjelmistokehystä. Tämä myös mahdollistaa pienillä muutoksilla koodin ajamisen muissakin pilvipalveluissa, kuten Google tai Azure -pilvipalveluissa. Näin pystytään välttämään palvelun sitomista tiettyyn teknologiaan tai palveluun.

Lambdan asetukset kirjoitettiin serverless.yml-tiedostoon, jota lukemalla ohjelmistokehys tunnistaa käytetyn pilvipalvelun ja luo funktion. Funktioille määritettiin myös HTTP-laukaisimet, jotta käyttöliittymä pystyy taustajärjestelmän kanssa viestimään. Hyvien käytäntöjen mukaisesti funktion osoite rakennettiin api-etuliitteestä, moduulin nimestä sekä toiminnosta.

4.2.2 Funktion rakenne

Lambda myös edellyttää käsittelijä-rakennetta koodissa, eli funktiota, joka ottaa vähintää tapahtuma-parametrin vastaan ja palauttaa vastauksen kutsuun. Koodin selkeyttämiseksi kyseinen käsittelijä sijoitettiin omaan tiedostoonsa ja se myös määritettiin ottamaan tapahtuma-parametrin lisäksi konteksti sekä takaisinkutsu (callback) -parametrit. Käsittelijä kutsuu logikkaa suorittavaa funktiota, joka palauttaa arvon, minkä perusteella

loppukäyttäjälle vastataan. Erillisen käsittelijän ansiosta voitiin testata erikseen tiettyjä pieniä osa-alueita yhden ison koodi lohkareen sijasta.

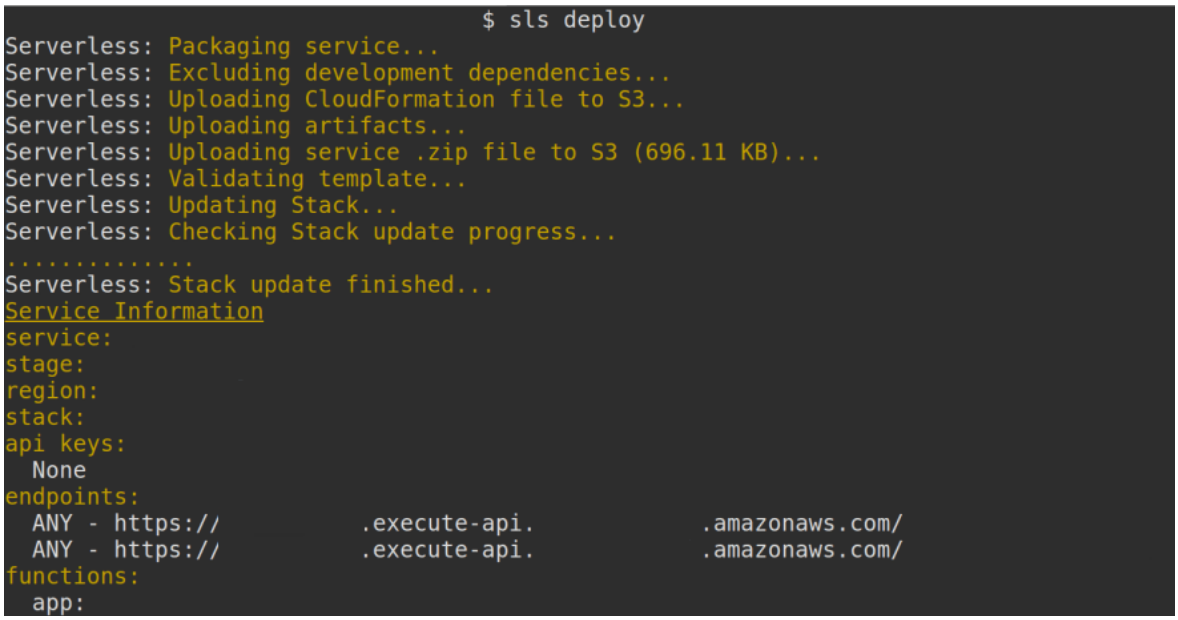
Konteksti-parametrin avulla saatiin tietoon AWS-palvelun luoma uniikki tunnus, jolla tietty kutsu voitiin indetifioida. Tämä liitettiin ongelmatilanteissa tehtyihin lokeihin virhetiedon kanssa. Vikatilanteessa tunnuksen avulla voitiin selvittää ongelman laajuus, mikä nopeutti viankorjaamista. Sovelluksen laajetessa tunnusta voitiin käyttää myös kutsujen etenemisen seurantaan taustajärjestelmässä, jos toimintaa haluttiin tarkemmin tutkia.

Takaisinkutsu-parametria käytetään itse määritetyn vastauksen palauttamiseksi kutsujalle. Virhetilanteissa palautetaan virhettä kuvaava HTTP-tilakoodi sekä viesti, jotta käyttöliittymä voi kutsun epäonnistuessa informoida loppukäyttäjää. Onnistuneissa kutsuissa käytetään yleistä HTTP-tilakoodia 200 ja palautetaan käyttäjälle muokattu kuva.

4.2.3 Lambdan ajaminen ja ylläpito

Komentokehotetta käyttäen funktio ladattiin pilvipalveluun. Funktiota varten luotiin S3-bucketti koodi tiedostoille, CloudFormation-pino resurssien sitomiseen, API Gateway kutsurajapinnaksi sekä itse Lambda funktio koodin ajamiseen. Tässä rakenteessa API Gateway toimii funktion laukaisimena, kun se vastaanottaa HTTP-pyynnön. Jos kutsussa on virheellinen osoite, ei Lambdaa ajeta vaan API Gateway palauttaa oletusarvoisesti kutsujalle HTTP-tilakoodin 403.

Lambdaa on mahdollista ajaa myös paikallisesti komentokehotetta käyttäen. Komennossa määritetään funktion nimi serverless.yml-tiedostossa sekä mahdollinen tapahtuma JSON-formaatissa. Tätä käytettiin testausmielessä ennen pilvipalveluun lataamista, sillä paikallisesti ajaminen on nopeampaa kuin koodin lataaminen ja ajaminen. Paikallisesti ajettaessa tuli myös ottaa huomioon komentokehotteen käyttämä NodeJS-versio, sillä serverless.yml -tiedostossa määritettyä versiota ei käytetä. Tästä ilmeni ongelmia tilanteissa, joissa koodi sisälsi ominasuuksia, joita ei vielä Lambdassa käytetty versio tukenut.



Kuva 11. Lambdan päivittäminen komentokehotteella

Lambdan päivittäminen tapahtuu Serverless Frameworkin ansiosta samalla tavalla kuin lataaminen. Taustalla Serverless Framework luo CloudFormation-pinoon tarvittavat muutos sarjat, ja päivittää funktion resurssit. Päivityksen aikana komentokehotteella on nähtävissä päivityksen tila, sekä lopulta ladattujen funktioiden tiedot (kuva 11). Tarvittaessa Cloudformation muutokset voidaan perua Amazon Web Servicen web-sivuilta.

4.3 Testaus

Sovellusta haluttiin testata, jotta julkaisun jälkeen ei palvelussa ilmenisi käyttöä estäviä virheitä. Testauksen tarkoituksena oli käydä läpi ohjelmiston toimintoja, mahdollisten virheiden löytämiseksi. Testausta suoritettiin kehityksen aikana sekä jälkeen, jotta sovellus voitiin todeta toimivaksi.

4.3.1 Sovelluksen toiminnan varmentaminen

Chai-kirjastoa käyttäen määritettiin useita testitapauksia, joita ajettiin läpi kun tietty toiminnallinen kokonaisuus oli valmis. Näissä tapauksissa sovellukselle syötetiin tietyt arvot ja määritetiin oletettu tulos. Tuloksen poiketessa oletetusta pystyttiin toteamaan tietyn toiminnallisuuden olevan puuttellinen. Projektin vaatimusten muuttuessa myös testejä päivitettiin, jotta koodiin tehtyjen muutosten paikkaansa pitävyys oli mahdollista todeta.

Testit jaettiin kahteen luokkaan, epäonnistuviin ja onnistuviin. Epäonnistuvissa testeissä sovellukselle syötettiin virheellistä dataa, jolloin oletettiin testin palauttavan virheviestin.

Näin oli mahdollista löytää virheet koodissa olevista suodattimista, joilla pyrittiin estämään virheellisen kuvan luonti ja sovelluksen kaatuminen

Onnistuvissa testeissä käytettiin hyväksi esimerkki kuvatiedostoja ja kuvan asetteluun liittyviä asetuksia. Sovellukselle annettiin ehjä ja tuettu kuvatiedosto sekä hyväksyttävät asetukset, jolloin oletettiin tuloksena olevan käytettävä kuva oikeassa formaatissa. Muilla työkaluilla oli mahdollista luoda hitsauskoneen paneelille sopiva kuvatiedosto, johon sovellukselta saatua tulosta verrattiin. Tämän ansiosta sovelluksen toiminnallisuutta kyettiin todentamaan bittien tarkkuudella.

4.3.2 Yhteensopivuus hitsauskoneen paneelin kanssa

Vaikka koodissa käytetyt, tai käyttöliittymän testit olivat onnistuneet, testattiin tulosta myös hitsaskoneen paneelin kopiolla. Näin oli helppo nähdä oliko sovellus ajantasalla hitsauslaiteen toiminnallisten vaatimusten kanssa, sekä oliko palvelusta saatu kuva oikeassa formaatissa. Koodikohtaisissa testeissä saattaa olla virheitä, mutta mikäli kuvaa ei paneelille saatu näkyväksi, nähtiin heti toiminnalliset puutteet.

Kuvaa asetettaessa paneelille virheiden sijaintia ei voitu yhtä tarkasti havaita kuin yksikkötesteissä. Mikäli paneeli ei hyväksynyt kuvaa, voitiin päätellä kuvaformaatin olevan virheellinen, mutta itse kuvan rakennetta ei voitu todentaa. Tästä syystä paneelilla testaamisella edettiin vain vaihe kerrallaan, kuvan asentamisesta sen asettamiseen, mikä oli hitaampaa kuin yksikkötestaaminen. Paneelilla testatessa löydettiin loppujen lopuksi useita vikoja, joita ei muuten olisi voitu löytää.

5 KÄYTTÖLIITTYMÄ

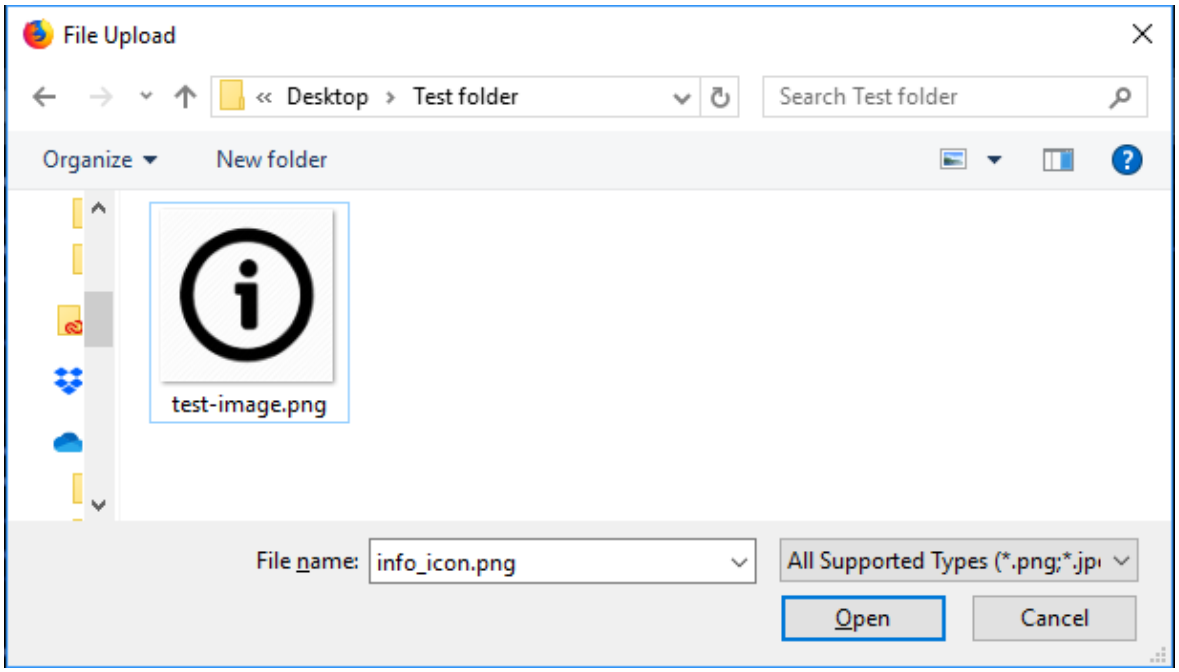
Palvelun käyttöliittymä tehtiin Angular 2+ -ohjelmistokehyksellä, sekä tyylittelyssä hyödynnettiin myös Angular Material-tyylikirjastoa. Koodissa hyödynnettiin modulointia, joten tarvittaessa sitä voidaan laajentaa näyttämään muitakin näkymiä vaatimatta suuria muutoksia. Käyttöliittymä sisältää itse kuvanmuokkaus alustan sekä ohjeet kuvan asentamiseksi hitsauskoneelle.

5.1 Kuvanmuokkaus

Käyttöliittymässä kuvalle on mahdollista suorittaa useita kuvanmuokkaukseen liittyviä operaatioita. Itse kuvan manipulointi tapahtuu taustajärjestelmässä, mitä varten käyttöliittymä lähettää kuvadatan sekä sen asetukset. Käyttöliittymän ja taustajärjestelmän välinen kommunikaatio tapahtuu HTTP-kutsuilla.

5.1.1 Kuvatiedoston lataaminen

Sivuston auetessa käyttäjä näkee kuvan hitsauslaitteen paneelista, jossa on nappi latausikonilla. Painamalla kyseistä nappia aukeaa käyttöjärjestelmäkohtainen ikkuna, jossa käyttäjä voi valita järjestelmästään tiedoston. Alla olevassa kuvassa (kuva 12) on esillä Windows 10-käyttöjärjestelmän versio tiedoston lataamisesta, jossa on myös mahdollista hakea tiedosto esimerkiksi Onedrive-pilvitallennustilasta.



Kuva 12. Windows 10-käyttöjärjestelmän tiedostonlataus-ikkuna

Palvelussa haluttiin tukea yleisimpiä kuvaformaatteja, jotta loppukäyttäjän ei tarvitse useimmissa tilanteissa itse konvertoida kuvaa. Rajaamalla tuettujen formaattien määrää, voidaan myös yksinkertaistaa taustajärjestelmää, mikä vähentää mahdollisten virheiden määrää. Lopulta päädyttiin tukemaan .jpg- ja .png -formaatteja.

Tiedostonlataus-ikkunalle on mahdollista myös määrittää tuetut tiedostoformaatit, mikä on nähtävissä edellisen kuvan (kuva 12) oikeassa alareunassa. Näin voidaan helpottaa käyttäjää oikeiden tiedostojen löytämisessä, sekä vähentää virheellisten syötteiden määrää. Tämä tiedostojen suodatin on mahdollista ohittaa valitsemalla kaikkien tiedostojen näyttämisen samasta valikosta. Ongelmien välttämiseksi oli määritettävä käyttöliittymän koodiin valitun tiedoston formaatin tarkistus, jotta käyttäjä ei voi vahingossa aiheuttaa vahinkoa. Lisäksi ladatun kuvan kokoa tuli rajoittaa, sillä taustajärjestelmälle ei voida lähettää yli 6 megatavun viestejä.

5.1.2 Kuvan asettelu

Kuvaa asetellaan hitsauspaneelista tehdyn kuvan sisällä, jotta käyttäjä näkee miltä lopullinen tulos tulisi näyttämään hitsauslaitteellaan. Näytön koosta riippumatta suhde paneelin ja ladatun kuvan välillä pidetään samana koko käytön ajan prosentuaalisten CSS-yksiköiden avulla. Kuvan todellinen pikselikoko lasketaan vasta latausvaiheessa sen elementin koosta ja tämä kerrotaan suhdeluvulla, mikä saadaan hitsauspaneeli kuvan mittojen suhteesta sen todelliseen kokoon.

Jotta kuvan siirtely ei vaatisi liikaa resursseja laitteelta, animaatioita eli tässä tapauksessa liikettä, suoritetaan silloin kun selain on valmis. Tämä saavutetaan käyttämällä selaimien rajapinnan tarjoamaa requestAnimationFrame -funktiota, joka pyytää selainta kutsumaan sille annettua funktiota ennen sivun seuraavaa uudelleenpiirtämistä (Mozilla 2019a). Itse kuvan siirto tapahtuu käyttäen CSS transform -ominaisuuden arvojen muuttamista, mikä hyödyntää näytönohjainta prosessorin sijaan, ollen kevyempi suorittaa (Mozilla 2019b). Käytön helpottamiseksi kuvan siirtäminen kokonaan paneelin ulkopuolelle on estetty tarkkailemalla hiiren ja kosketuksen sijaintia näytöllä suhteessa paneelin reunoihin.

Kuvan kokoa voidaan myös muokata käyttämällä paneelin yläpuolella olevaa työkalupalkkia. Käyttäjä voi suurentaa tai pienentää kuvaa plus- ja minus-painikkeilla. Fit-painikkeilla on myös mahdollista sovittaa kuva pysty- tai vaakasuunnassa paneelin rajojen sisäpuolelle, mikäli kuvan kyseinen mitta on suurempi kuin paneelin.

5.2 Käyttäjäkokemuksen edistäminen

Käyttöliittymää tehtäessä otettiin huomioon käyttäjäkokemusta edistäviä ominaisuuksia. Tavoitteena oli saavuttaa mahdollisimman laajalti loppukäyttäjiä eri alustoilla ja laitteilla. Palvelun tuli myös olla helposti käytettävä sekä jäsennelty, jotta loppukäyttäjä kokee jokaisen vaiheen etenevän luonnollisesti.

5.2.1 Responsiivisuus

Jotta palvelu olisi käytettävissä usealla eri alustalla sekä näyttökoolla, täytyi sivusto toteuttaa responsiiviseksi. Eri näytöillä palvelun tuli skaalata samassa suhteessa muuttamatta sen rakennetta liikaa. Tästä syystä koko sovelluksen käyttöliittymä oli suunniteltu mobiililaitteelle sopivaksi. Eteneminen palvelussa tapahtui ylhäältä alaspäin, eikä suurimpia elementtejä asetella vierekkäin.

Visuaalisten elementtien skaalatuvuuus saavutettiin käyttämällä CSS:n prosentti sekä em ja rem -yksiköitä kokoja määriteltäessä. Näin eri elementit saatiin suhteuteuttua toisiinsa, jolloin tuli vain asettaa muutamia mittoja käyttäen pikseleitä. Kyseiset tarkat pikselimitat asetettiin media query -ominaisuutta hyödyntäen yleisimmille näytöille sopivaksi. Näin eri kokoisilla näytöillä mittasuhteet asettuvat automaattisesti ja käyttäjäkokemus oli miltei identtinen.

Asetetut tyylit testattiin käyttämällä Chrome-selaimen kehitystyökalua. Työkalulla simuloitiin eri näyttökokoja, mobiililaitteista suuriin näyttöihin (kuva 13). Kehitystyökalulla oli myös mahdollista kokeilla kosketus rajapintaa, minkä ansiosta kuvan asettelemiseen liittyviä ongelmia saatiin korjattua.

Kuva 13. Chrome-selaimen laite kehitystyökalu

5.2.2 Palaute

Käyttöliiittymän informatiivisuus oli myös suuressa osassa toiminnallisuutta tehtäessä. Käyttäjän painaessa nappeja tai odottaessa latausta sivuston tilan tulee muuttua, jotta käyttäjä tietää palvelun toimivan. Kuvaa muokatessa itse kuva muuttuu ja uuden kuvan lataus-nappia painatessa aukeaa selaimen sisäänrakennettu tiedoston valintanäkymä, mitkä ovat palautteena itsestäänselviä.

Kuva 14. Latausnapin ulkonäkö eri latauksen tiloissa

Kuvaa ladattessa toiminnallisuus siirtyy taustajärjestelmälle, joten tätä päädyttiin informoimaan muokkaamalla latausnapin tekstiä sekä väriä sen hetkisestä tilasta riippuen. Edellisessä kuvassa (kuva 14) latausnapin teksti muuttuu painamisen jälkeen ylimmästä keskimmäiseen. Latauksen valmistuessa nappi siirtyy kuvan alimpaan vaiheeseen jolloin sen taustaväri poistuu ja tekstiksi tulee sana ready eli valmis. Samalla myös selaimen tyypistä riippuen aukeaa tiedostontallennus -ikkuna tai vaihtoehtoisesti ilmoitus, mikä kertoo tiedoston latauksesta valmistumisesta.

Mikäli palvelun aikana tapahtuu virhe, tämä myös kerrotaan käyttäjälle. Virhetilanteessa aukeaa pieni suljettava ikkuna missä on virhettä kuvaava teksti. Virheikkunan otsikko tulostuu punaisella värillä, mikä on yleisesti käytetty keino kuvata virhetilannetta. Vaikka palveluita ei suunnitella tuottamaan virheitä, oli tämä erikoistilanne hyvä ottaa huomioon, jotta käyttäjälle tulee selväksi mitä on tapahtunut.

5.2.3 Ohjeistus

Palvelun toimintojen tuli myös olla mahdollisimman selviä, jotta käyttäjä ei joudu pohtimaan seuraavaa vaihettaan liian kauaa tai yllättymään valintojensa vaikutusta. Edellisessä kuvassa (kuva 14) ylimpänä näkyy latausnappi ennen latauksen alkua, mihin viitataan tekstillä download eli lataa. Ohjaavien tekstien ohella käytetään myös yleisesti käytössä olevia ikoneja, joihin suurinosa käyttäjistä on todennäköisesti tutustunut aiemmin.

Kun käyttäjä on ladannut sivulle valitsemansa kuvan, tulee työkalupalkki näkyviin (kuva 15). Palkin toiminnoissa käytetyt ikonit ovat yksinkertaisia ja ne kuvaavat toimintoansa. Ikonien ohella silti päädyttiin lisäämään toimintoryhmille omat otsikot, jotta ryhmän toimintoja ei ymmärretä väärin. Erityisesti Fit-ryhmän napit saattavat antaa käyttäjälle kuvan, että nappeja painamalla kuva liikkuu pysty- ja vaakasuunnassa, vaikka toiminto liittyy kuvan sovittamiseen. Reset-napin tapauksessa myös ikonia muokattiin lyhentämällä puoliympyrän muodossa olevaa viivaa, jotta se ei sekaantuisi yleisesti käytetyn uudelleenlataus-ikonin kanssa.

Kuva 15. Kuvan muokkauksessa käytetyt painikkeet

Käyttäjää pyritään ohjaamaan sivulla jokaisen toiminnallisen vaiheen jälkeen. Vaiheita kuvataan ympyröidyillä järjestysluvuilla, jotka ovat aina näkyvissä. Tärkeänä pidettiin myös seuraavan vaiheen näkymistä, jotta käyttäjän ei tarvitse etsiä sivulla seuraavaa toimintoa. Aloitusnäkymässä (kuva 16) käyttäjän tulee ladata oma kuvansa palveluun. Jo tässä vaiheessa käyttäjälle näytetään latausvaihe, vaikka kuvaa ei ole vielä valittu. Kun kuvan latauspainiketta on painettu ja kuva ladattu, siirrytään automaattisesti sivulla alaspäin, jolloin näytetään seuraava vaihe. Mikäli käyttäjän selain ei tue automaattista siirtymistä, on seuraava vaihe helppo löytää, sillä sivustolla ei voi liikkua kuin ylös ja alas.

Kuva 16. Vaiheiden numerointi

5.3 Toiminnallisuuden toteaminen

Palvelu on tarkoitettu käytettäväksi yleisimmillä selaimilla, ja selainversioilla. Tässä tapauksessa toimivuus Mozilla Firefox, Google Chrome, Microsoft Edge ja Internet Explorer versio 11 -selaimilla määritettiin vähimmäisvaatimukseksi. Selainten välillä on toiminnallisia eroja, jotka tuli ottaa palvelua kehittäessä huomioon. Erojen löytämiseen käytettiin Mozillan ylläpitämää MDN Web Docs -sivustoa sekä Can I Use -palvelua. Näissä internet resursseissa on mahdollista hakea tiettyä ominaisuutta, ja nähdä millä selaimilla ja selainversioilla se on tuettuna.

Alusta riippumattomuutta todettiin myös testaamalla palvelua eri selaimissa jo kehityksen aikana. Testauksen aikana sivustolla tehtiin useita eri toimintoja eri järjestyksessä mahdollisten ongelmien löytämiseksi. Testausta tehtiin useaan otteeseen, jotta suurimmat

virheet löydettäisiin, sekä jo korjatut virheet voitiin todeta korjatuksi. Alustojen eroista ei toiminnallisia virheitä löytynyt, mutta visuaalisesti Internet Explorerin tapauksessa tuli tehdä poikkeuksia.

Kuva 17. Esimerkki kuva alkuperäisenä, asettelussa käyttöliittymässä sekä taustakuvana hitsauskoneen paneelissa

Lopulta koko järjestelmän toiminnallisuutta testattiin esimerkki kuvilla. Kuvaa ensin muokattiin palvelussa, jossa otettiin kuvan laskettu koko ylös. Myös kuvan asentoa ja kuvan koosta riippuen, sen ympärille jäävää tilaa verrattiin hitauspaneelille asennettuun kuvaan. Kuva asennettiin latauksen jälkeen näytölle ilmestyvien ohjeiden mukaisesti. Sen ilmestyessä paneelille oikean näköisenä ja kokoisena, voitiin todeta koko prosessin toimivan oikein (kuva 17).

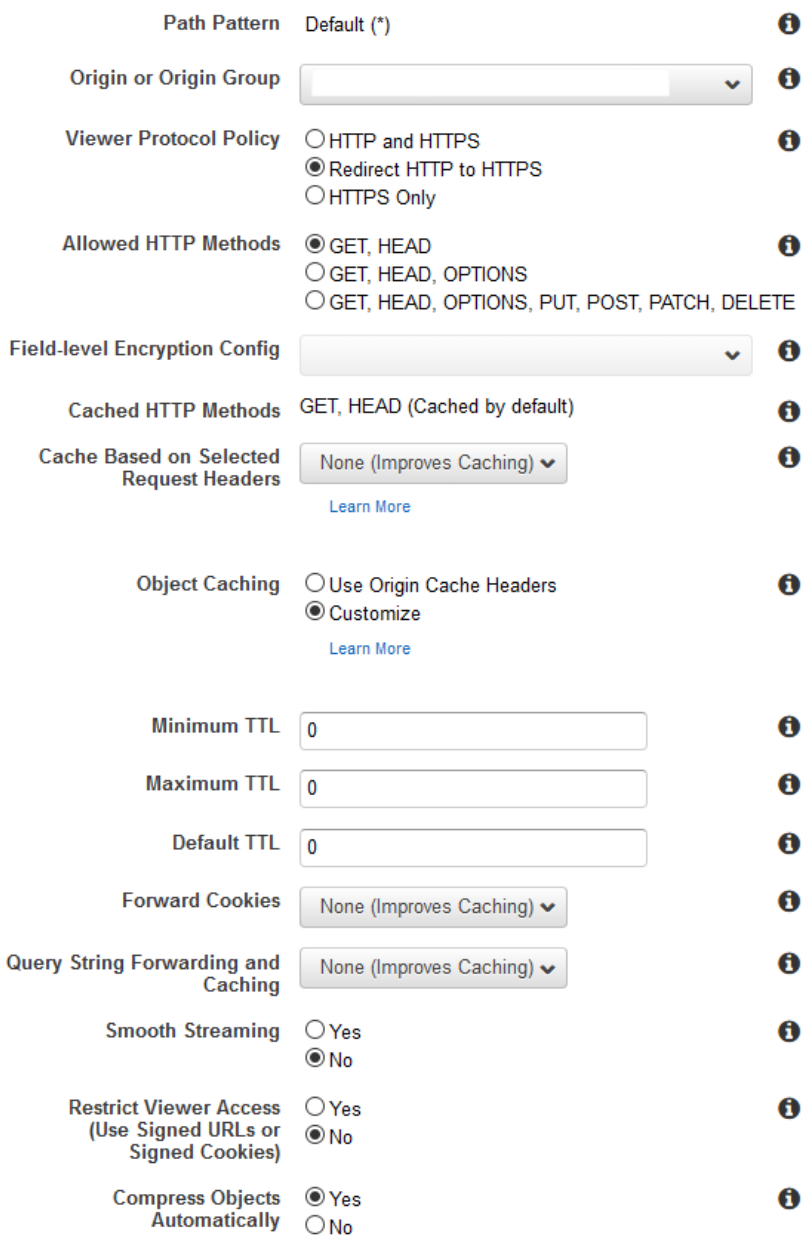
5.4 Ylläpito pilvipalvelussa

Käyttöliittymää jaetaan ja säilytetään Amazon Web Services -pilvipalvelussa, käyttäen S3-, CloudFront- ja Route53 -palveluita. Nämä pilvipalvelut mahdollistavat sivuston saavutettavuuden ympäri maailmaa, jolloin loppukäyttäjän tarvitsee tietää vain sivuston

osoite. Palveluiden käyttöönotossa otettiin huomioon tietoturvallisuus, mahdollisimman laaja loppukäyttäjien saavutettavuus sekä itse palvelun nopeus.

Itse käyttöliittymän tiedostot säilytetään S3-bucketissa, jossa on käytössä staattisen web-sisällön isännöinti (hosting). Buckettiin yhdistettäessä palautetaan automaattisesti bucketin ylimmällä tasolla oleva index.html-tiedosto. Se sisältää pohjasivun sekä viitteet muihin sivuston vaatimiin tiedostoihin. Bucketista myös estetään sen julkinen käyttö, jotta pääsy sivulle tapahtuu muiden palveluiden kautta.

Sivuston jakaminen tapahtuu CloudFront-palvelun kautta, jossa on määritetty kaikki sivuston jaettavat resurssit. Tietoturvaa edistämiseksi CloudFront-jakeluverkolle asetettiin SSL-certifikaatti, jotta sisältö voidaan turvallisesti jakaa HTTPS-protokollaa käyttäen. Sisällön nopeampi jakaminen myös mahdollistettiin tukemalla kaikkia HTTP-versioita, sekä käyttämällä kaikkia toiminnassa olevia jakelupisteitä ympäri maailmaa.

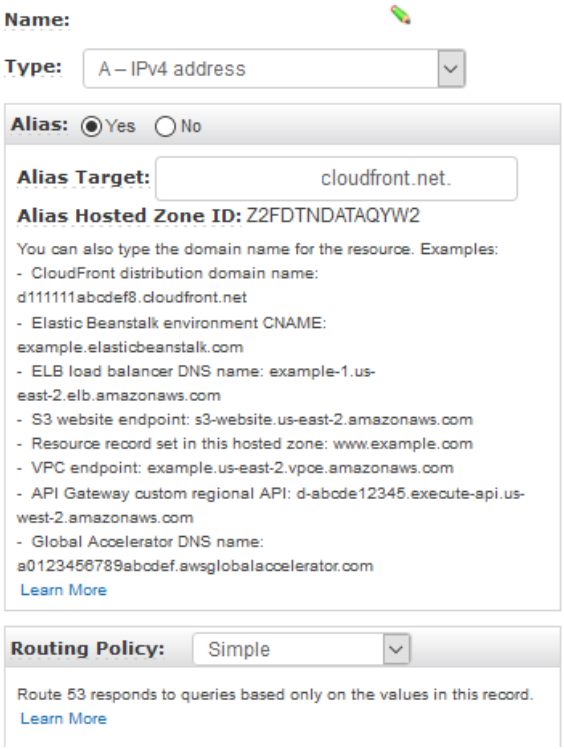


Kuva 18. Oletuspolun käyttäytymisasetukset Cloudfront jakeluverkossa

Cloudfront jakeluverkostolle asetettiin käytetyn S3-bucketin osoite, jolloin se luo linkin näiden palveluiden välille. Sivusto on jakeluverkon pääresurssi, joten sitä jaetaan oletusarvoisesti, mikäli muut kutsut Cloudfronttiin eivät toteudu. Ylläolevassa kuvassa (kuva 18) on Bucketin käyttäytymiseen liittyvät asetukset, joilla ohjataan kutsut tarvittaessa käyttämään turvallisempaa HTTPS -protokollaa, sekä estetään tarpeettomat HTTP-kutsutyypit. Myös sivuston tiedostojen ja sen HTTP-kutsujen lisäkenttien tallennus välimuistiin on otettu pois päältä, jotta jaettu sisältö on aina uusin versio palvelusta.

Lopulta palvelulle asetettiin oma DNS-osoite Route53-palvelua käyttäen. Osoite luodaan aliaksena, jolloin se voidaan linkittää suoraan Cloudfrontin-resurssiin, ohjaten kaikki kutsut

määritettyyn DNS-osoitteeseen automaattisesti. Osoitteen luonti myös mahdollistaa SSL-sertifikaatin käytön, sillä sertifikaatit luodaan verkkotunnus kohtaisesti. Route53 ohjauksen luomisnäkymässä määritetään IPv4-protokollaa käyttävien HTTP-kutsujen ohjauksesta nimi-kenttään määritetystä DNS-osoitteesta Cloudfront-palveluun (kuva 19).



Kuva 19. Route 53-palvelussa luodun DNS-osoitteen asetukset

6 YHTEENVETO

Työn tavoitteena oli toteuttaa sovellus hitsauskoneen paneelin taustakuvan luomista varten. Sovelluksen avulla tuli pystyä asettamaan, muokkaamaan sekä lataamaan loppukäyttäjän palveluun antama kuva. Sovelluksen käytön tuli myös olla helppoa eri laitteilla ja alustoilla.

Koko sovelluksen vaatima järjestelmä rakennettiin Amazon Web Services-pilvipalvelua käyttäen. Käytetyt palvelut valittiin järjestelmän vaatimusten sekä käyttökokemuksen perusteella. Käyttöliittymä sekä taustajärjestelmä toteutettiin erillään käyttäen parhaiten soveltuvia palveluita.

Valmiin järjestelmän avulla loppukäyttäjän on mahdollista luoda ja ladata taustakuva MasterTig-hitsauskonettaan varten. Käyttäjälle näytetään myös ohjeet kuvan asentamiseksi. Käyttöliittymä toimii yleisimmissä selaimissa sekä sen rakenteessa on otettu huomioon myös mobiililaitteet. Eteneminen käyttöliittymässä on virtaviivaista ja se päättyy kuvan asennusohjeisiin. Työn lopputulos oli tavoitteiden mukainen, ja palvelu on ollut käytössä 13.06.2019 alkaen.

Tarvittaessa palvelua on mahdollista laajentaa tukemaan muita kuvanmuokkaus operaatioita.Tämä ei vaadi suurta olemassa olevan koodin muokkausta, vain uusien ominaisuuksien lisäämistä. Myös tuettujen tiedostoformaattien määrää voidaan kasvattaa, mutta se saattaa vaatia uusien kuvanmuokkaus-työkalujen etsimistä tai kehittämistä.

LÄHTEET

Amazon Web Services 2019a. Global Infrastructure [viitattu 11.8.2019]. Saatavissa: https://aws.amazon.com/about-aws/global-infrastructure/

Amazon Web Services 2019b. Cloud computing with AWS [viitattu 11.8.2019]. Saatavissa: https://aws.amazon.com/what-is-aws/

Amazon Web Services 2019c. Building Applications with Serverless Architectures [viitattu 11.8.2019]. Saatavissa: https://aws.amazon.com/lambda/serverless-architectures-learn-more

Amazon Web Services 2019d. Serverless [viitattu 11.8.2019]. Saatavissa: https://aws.amazon.com/serverless/

Amazon Web Services 2019e. Introduction to Amazon S3 [viitattu 18.8.2019]. Saatavissa: https://docs.aws.amazon.com/AmazonS3/latest/dev/Introduction.html

Amazon Web Services 2019f. Amazon S3 [viitattu 18.8.2019]. Saatavissa: https://aws.amazon.com/s3/

Amazon Web Services 2019g. Amazon S3 Storage Classes [viitattu 18.8.2019]. Saatavissa: https://aws.amazon.com/s3/storage-classes/

Amazon Web Services 2019h. Amazon S3 Features [viitattu 18.8.2019]. Saatavissa: https://aws.amazon.com/s3/features/

Amazon Web Services 2019i. Working with Amazon S3 Buckets [viitattu 18.8.2019]. Saatavissa: https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html

Amazon Web Services 2019j. Bucket Policy Examples [viitattu 18.8.2019]. Saatavissa: https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies.html

Amazon Web Services 2019k. Access Control List (ACL) Overview [viitattu 18.8.2019]. Saatavissa: https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html

Amazon Web Services 2019l. Working with Amazon S3 Objects [viitattu 24.8.2019]. Saatavissa: https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingObjects.html

Amazon Web Services 2019m. Object key and Metadata [viitattu 25.8.2019]. Saatavissa: https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html

Amazon Web Services 2019n. What Is AWS Lambda [viitattu 1.9.2019]. Saatavissa: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Amazon Web Services 2019o. Security Overview of AWS Lambda [viitattu 1.9.2019]. Saatavissa: https://pages.awscloud.com/rs/112-TZM-766/images/Overview-AWS-Lambda-Security.pdf

Amazon Web Services 2019p. Actions [viitattu 1.9.2019]. Saatavissa: https://docs.aws.amazon.com/lambda/latest/dg/API_Operations.html

Amazon Web Services 2019q. AWS Lambda Execution Context [viitattu 1.9.2019]. Saatavissa: https://docs.aws.amazon.com/lambda/latest/dg/running-lambda-code.html

Amazon Web Services 2019r. What is a Container? [viitattu 5.10.2019]. Saatavissa: https://aws.amazon.com/containers/

Amazon Web Services 2019s. AWS SDK for Javascript in the Browser [viitattu 5.10.2019]. Saatavissa: https://aws.amazon.com/sdk-for-node-js/

Amazon Web Services 2019t. Tools to Build on AWS [viitattu 5.10.2019]. Saatavissa: https://aws.amazon.com/tools/

Amazon Web Services 2019u. What Is Amazon API Gateway? [viitattu 5.10.2019]. Saatavissa: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

Amazon Web Services 2019v. Use API Gateway Lambda Authorizers [viitattu 6.10.2019]. Saatavissa: https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html

Amazon Web Services 2019w. AWS Customer Agreement [viitattu 18.8.2019]. Saatavissa: https://aws.amazon.com/agreement/

Amazon Web Services 2019x. Controlling and Managing Access to a REST API in API Gateway [viitattu 6.10.2019]. Saatavissa: https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-control-access-to-api.html

Amazon Web Services 2019y. What is Amazon CloudFront [viitattu 17.10.2019]. Saatavissa: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html

Amazon Web Services 2019z. How CloudFront Delivers Content [viitattu 17.10.2019]. Saatavissa: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/HowCloudFrontWorks.html

Amazon Web Services 2019å. Amazon S3 Frequently Asked Questions [viitattu 18.8.2019]. Saatavissa: https://aws.amazon.com/s3/faqs/

Angular 2019a. Architecture overview [viitattu 6.10.2019]. Saatavissa: https://angular.io/guide/architecture

Angular 2019b. Introduction to components [viitattu 5.10.2019]. Saatavissa: https://angular.io/guide/architecture-components

Angular 2019c. Template Syntax [viitattu 5.10.2019]. Saatavissa: https://angular.io/guide/template-syntax

Angular 2019d. Browser support [viitattu 12.10.2019]. Saatavissa: https://angular.io/guide/browser-support

Angular 2019e. Deployment [viitattu 12.10.2019]. Saatavissa: https://angular.io/guide/deployment

Angular 2019f. Component Styles [viitattu 12.10.2019]. Saatavissa: https://angular.io/guide/component-styles

Angular 2019g. NgModules [viitattu 12.10.2019]. Saatavissa: https://angular.io/guide/ngmodules

Angular 2019h. Feature Modules [viitattu 12.10.2019]. Saatavissa: https://angular.io/guide/feature-modules

Angular 2019i. Lazy Loading Feature Modules [viitattu 12.10.2019]. Saatavissa: https://angular.io/guide/lazy-loading-ngmodules

Angular 2019j. Theming your Angular Material app [viitattu 13.10.2019]. Saatavissa: https://material.angular.io/guide/theming

Deeleman, P. & Noring, C. 2017. Learning Angular. 2. painos. Birmingham: Packt Publishing.

Deeleman, P. 2016. Learning Angular Second Edition. Birmingham: Packt Publishing.

Ecma International 2017. The JSON Data Interchance Syntax. 2. uudistettu painos. Ecma International. Saatavissa: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

Kemppi Oy. 2019. Tietoa Kempistä [viitattu 3.8.2019]. Kemppi Oy. Saatavissa: https://www.kemppi.com/fi-FI/yritys/kemppi/tietoa-yrityksesta/

Mathew, S. 2019. Overview of Amazon Web Services [viitattu 11.8.2019]. Amazon Web services. Saatavissa: https://d1.awsstatic.com/whitepapers/aws-overview.pdf

Monroe, C. & Lovaas, P. 2018. 13th Annual Symposium of Information Assurance: Securely moving forward with Serverless Architecture. Kokoelma [viitattu 11.8.2019]. Saatavissa: https://www.albany.edu/iasymposium/proceedings/2018/ASIA2018_Proceedings.pdf

Mozilla 2019a. Window.requestAnimationFrame() [viitattu 22.9.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame

Mozilla 2019b. Performance fundamentals [viitattu 22.9.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/Performance/Fundamentals

Mozilla 2019c. Polyfill [viitattu 12.10.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Glossary/Polyfill

Mozilla 2019d. Pointer events [viitattu 12.10.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/Pointer_events

Mozilla 2019e. Gamepad API [viitattu 12.10.2019]. Saataivssa: https://developer.mozilla.org/en-US/docs/Web/API/Gamepad_API

Mozilla 2019f. Vibration API [viitattu 12.10.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/Vibration_API

Mozilla 2019g. Battery Status API [viitattu 12.10.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/Battery_Status_API

Mozilla 2019h. CSS: Cascading Style Sheets [viitattu 12.10.2019]. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/CSS

Serverless 2019a. Documentation [viitattu 5.10.2019]. Saatavissa: https://serverless.com/framework/docs/

Serverless 2019b. Services [viitattu 5.10.2019]. Saatavissa: https://serverless.com/framework/docs/providers/aws/guide/services/

Serverless 2019c. Serverless.yml Reference [viitattu 5.10.2019]. Saatavissa: https://serverless.com/framework/docs/providers/aws/guide/serverless.yml/

Serverless 2019d. AWS – Functions [viitattu 5.10.2019]. Saatavissa: https://serverless.com/framework/docs/providers/aws/guide/functions/

Serverless 2019e. AWS – deploy [viitattu 5.10.2019]. Saatavissa: https://serverless.com/framework/docs/providers/aws/cli-reference/deploy/

Serverless 2019f. AWS – Invoke [viitattu 5.10.2019]. Saatavissa: https://serverless.com/framework/docs/providers/aws/cli-reference/invoke/