BOTIT JA NHL-URHEILUBOTIN KEHITYSPROSESSI SEKÄ SAAVUTETTAVUUS

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Hämeenlinnan korkeakoulukeskus, Tietojenkäsittely

kevät, 2021

Mikko Hoikkala

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittely
Hämeenlinnan korkeakoulukeskus

Tekijä Mikko Hoikkala **Vuosi** 2021

Työn nimi Botit ja NHL-urheilubotin kehitysprosessi sekä saavutettavuus

Työn ohjaaja/t Lauri Salminen

TIIVISTELMÄ

Opinnäytetyö käsittelee sovellusta, joka hakee Internetistä tuoreimmat NHL:n tulokset ja tulostaa ne matkapuhelimella käytettävään käyttöliittymään. Sovellus tehtiin Android-käyttöjärjestelmälle. Sovellus myös listaa suomalaiset maalintekijät ja maalinsyöttäjät ja muodostaa suomalaisista maalintekijöistä ja maalinsyöttäjistä kokonaisia lauseita. Opinnäytetyö käsittelee lisäksi botteja ja niiden toiminnallisuuksia.

Sovellus saatiin toiminnalliseksi useiden erinäisten yritysten jälkeen halutulla tavalla. Sovelluksen tekemisen yhteydessä oli tarkoitus miettiä myös saavutettavuutta ja sen roolia mobiilisovelluksissa.

Opinnäytetyössä selostetaan tekniikat, joita käytettiin tiedon hakemiseksi internetistä. Opinnäytetyössä myös kerrotaan sovelluksen kehittämisen aikana kohdatuista haasteista. Sovelluksessa on myös toiminto, että muodostetut uutiset luetaan ääneen käyttäen jotakin Androidin tekstistä puheeksi -ominaisuutta. Tekstistä puheeksi -ominaisuuden kieleksi valittiin englanti, koska sen käyttöön liittyvää lähdeaineistoa oli löydettävissä helposti.

Opinnäytetyön tuotoksena syntynyt sovellus kirjoitettiin Javalla. Myös C#:n käyttöä tuli testatuksi, mutta sovellus saatiin toiminnalliseksi Javalla, joten se valittiin lopulliseksi ohjelmointikieleksi.

Opinnäytetyössä oli tehdyn lisäksi tarkoitus hakea myös NBA:n ja Valioliigan tuloksia ohjelmaan mutta tämä osoittautui turhan haastavaksi, joten se jätettiin tekemättä. Muita syitä edellä mainitun toteuttamatta jättämiseksi olivat opinnäytetyön tekemiseen käytetty aika sekä se, että NBA:n ja Valioliigan tuloksista ei ollut helposti löydettävissä yhtä helppokäyttöistä rajapintaa kuin NHL:n tuloksista.

Avainsanat NHL, Java, Jsoup, JSON, Android

Sivut 39 sivua, joista lähteitä ja liitteitä 17 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Business Information Technology
Hämeenlinna University Centre

Author Mikko Hoikkala **Year** 2021

Subject Bots and the development process of an NHL sports bot and accessibility

Supervisors Lauri Salminen

ABSTRACT

This thesis project is about an application for Android that fetches the latest results of the NHL and prints them out to a user interface used with a mobile phone. The application was made for the Android operating system. It also lists the Finnish players who have scored or given a pass leading to a goal and forms complete sentences from the scorers and goal passers of matches. In addition, the thesis is about bots in general and their functionality.

The application was made functional in the desired way after several attempts. The goal of the thesis was also to think about accessibility and its role in applications used with mobile phones.

The thesis describes the techniques used to fetch the results of the NHL from the internet. It also describes the different unfunctional methods tried when making the application. The application also includes the property of reading the formed sentences out loud using a text-to-speech Android framework. The language of the text-to-to-speech framework was selected to be English because there were many resources available related to using a text-to-speech framework with English as its language.

The produced application was written in Java. The use of C# was also tested but the application was made functional using Java, so it was chosen as the used programming language.

There was also the intention to get the results of the NBA and Premiere League as part of the thesis project, but this proved to be too difficult, so it was left out of the project. Other reasons for excluding the results of the NBA and Premiere League included the time used to write the thesis and the fact that an API for the results from the NBA and Premiere League was not found at least not as easily as the API for the results from the NHL.

Keywords NHL, Java, Jsoup, JSON, Android

Pages 39 pages including sources and appendices 17 pages

SISÄLLYS

Liitteet

1 JOHDANTO

Tutkimusaiheena opinnäytetyössä ovat botit sekä opinnäytetyön tuotoksena syntynyt Androidilla käytettävä sovellus, joka on urheilubotti, joka listaa NHL-jääkiekkoliigan pelien tuloksia sekä suomalaisia maalintekijöitä ja maalinsyöttäjiä. Bottien yksi toiminta-alue on urheilutulokset, joten opinnäytetyön tekijä pääsi syventymään hyvin tähän aihealueeseen.

Opinnäytetyö käsittelee ohjelmistoa, joka hakee NHL:n tuloksia listaavalta Internet-sivulta tuoreimmat tulokset sekä suomalaiset maalintekijät ja maalinsyöttäjät ja tulostaa ne matkapuhelimella käytettävään käyttöliittymään. Lisäksi ohjelma lukee hakemansa tiedot matkapuhelimen näytöltä ääneen. Tämän lisäksi opinnäytetyön aiheena on yleisesti botit sekä niiden yleistyminen yhteiskunnassa sekä mitä hyviä ja huonoja puolia tässä asiassa on.

Opinnäytetyöni aihe on mielestäni kiinnostava, koska bottien toiminnallisuus ja yleistyminen yhteiskunnassa ovat mielestäni kiinnostavia asioita. Opinnäytetyöni kertoo yleisesti boteista sekä niiden toiminnallisuudesta. Lisäksi oli mielestäni kiinnostavaa ja opettavaista kehittää itse Androidilla toimiva ohjelmisto, jonka toiminnallisuus on hyödyllinen ja esimerkiksi urheilusta ja erityisesti jääkiekosta kiinnostuneelle henkilölle jopa tarpeellinen.

Myös käyttöliittymäsuunnittelu oli tärkeässä asemassa opinnäytetyön tekemisen aikana. Käyttöliittymäsuunnitteluun liittyy yhtenä tärkeänä osana saavutettavuus, jota myös käsitellään opinnäytetyössä.

Opinnäytetyön tuloksena syntynyt ohjelma toteutettiin hyödyntäen Androidin sovelluskehityksessä usein käytetyn Java-ohjelmointikielen sisältämiä ominaisuuksia käsitellä JSON-muodossa olevaa dataa. Toinen Androidin sovelluskehityksessä käytetty ohjelmointikieli on Kotlin mutta opinnäytetyön kirjoittajalla ei ollut aikaisempaa kokemusta Android-sovellusten tekemisestä Kotlin-ohjelmointikielellä, joten ohjelma päätettiin tehdä Java-ohjelmointikielellä, josta opinnäytetyön kirjoittajalla oli kokemusta muun muassa koulun kautta. (YourTeamInIndia, 2020)

Opinnäytetyön aihe saatiin ammattikorkeakoululta, kun taas itse ehdotettu Unity-pelimoottoriin ja siinä yleisesti käytettyyn C#-ohjelmointikieleen liittyvä valinta opinnäytetyön aiheeksi olivat asioita, jotka oli jo käyty läpi aikaisemmin omalla kurssillaan.

Aluksi opinnäytetyössä puhutaan yleisesti boteista sekä niiden hyvistä ja huonoista puolista. Tämän jälkeen puhutaan saavutettavuudesta sekä sen eri osa-alueista ja merkityksestä sovellusten käyttöliittymän suunnittelussa. Seuraavaksi puhutaan erilaisista käytetyistä teknisistä menetelmistä. Tämän jälkeen on käytännön osuus, joka sisältää itse ohjelman tekemisen sekä sen testaamisen. Lopuksi on päätelmät ja jatkokehitysideoita ohjelmalle.

Opinnäytetyö toteutettiin toiminnallisena opinnäytetyönä. Käytettyjä ohjelmointikieliä ovat Java ja C# ja käytetty kehitysympäristö on Android Studio. Tutkimuskysymykset ovat:

- Millainen käyttöliittymä kannattaa suunnitella ohjelmalle, joka nappia painamalla hakee NHL:n tuoreimmat tulokset Internetistä?
- Missä muodossa tieto tulee ja missä muodossa se esitetään?
- Mitä botit yleisesti ottaen ovat ja mihin käyttötarkoituksiin niitä nykypäivänä käytetään?
- Miten bottien yleistyminen on helpottanut asioiden hoitamista?
- Mitä mahdollisia haittapuolia bottien yleistymisellä on?

2 **BOTIT**

Botit ovat oleellinen osa opinnäytetyötäni, joten opinnäytetyössä kerrotaan niistä seuraavaksi. Botti tulee englannin kielen sanoista bot tai web robot, joka tarkoittaa tietokoneohjelmaa, joka on ohjelmoitu vastaamaan tiettyihin käyttäjän tekstimuodossa antamiin syötteisiin. (Sulava, 2017)

Esimerkki botin toiminnallisuudesta:

```
Botti: Miten voin auttaa?

Ihminen: Milloin lentoni AY123 lähtee?

Botti: Lentosi lähtee klo 19:00 portilta A19 ja on aikataulussa.
```

(Sulava, 2017)

Botit on ohjelmoitu tiettyjen tehtävien suorittamiseen. Ne ovat automatisoituja eli suorittavat tehtäviä ilman tarvetta ihmisiltä tuleville komennoille. Botit usein matkivat tai korvaavat ihmisten suorittamia tehtäviä. Tyypillisesti ne suorittavat toistuvia tehtäviä ja pystyvät tähän paljon nopeammin kuin ihminen pystyisi.

Botit toimivat yleensä verkon yli; yli puolet Internetin liikenteestä on bottien suorittamaa sisällön skannausta, vuorovaikutusta verkkosivustojen kanssa, keskustelua verkkosivustojen käyttäjien kanssa, tai hyökkäyskohteiden etsimistä. Jotkut botit ovat hyödyllisiä; esimerkiksi hakukonebotit, jotka indeksoivat sisältöä hakuja varten tai asiakaspalvelubotit, jotka auttavat verkkosivustojen käyttäjiä. Toiset botit taas ovat ”pahoja” ja on ohjelmoitu murtautumaan käyttäjätileille, skannaamaan verkkoa yhteystietojen hakemiseksi roskapostin lähettämistä varten, tai suorittamaan muita haitallisia toimenpiteitä. Jos botti on yhteydessä Internetiin, sillä on myös oma IP-osoitteensa. (CloudFlare, 2020)

Botit voivat olla:

- Chattibotteja: Botteja, jotka simuloivat ihmisten välisiä keskusteluja vastaamalla tiettyihin syötteisiin ohjelmoiduilla vastauksilla
- Internetiä läpikäyviä botteja (Googlebotteja): Botteja, jotka skannaavat verkkosivustojen sisältöjä eri puolilta Internetiä
- Sosiaalisia botteja: Botteja, jotka toimivat erilaisissa sosiaalisen median palveluissa
- Haitallisia botteja: Botteja, jotka käyvät Internetin sisältöjä läpi, lähettävät roskapostia tai suorittavat eri palveluihin käytettävien tunnusten kalasteluja (CloudFlare, 2020)

2.1 **Bottien haitallinen toiminta**

Mikä tahansa automatisoitu toiminta, joka on botin suorittama ja joka poikkeaa verkkosivuston omistajan haluamasta toiminnallisuudesta tai rikkoo verkkosivuston käyttöehtoja tai verkkosivuston robots.txt-tiedoston sääntöjä liittyen bottien käyttäytymiseen voidaan nähdä haitallisena toimintana. Botit, jotka yrittävät suorittaa kyberrikollisuutta, kuten identiteettivarkauksia tai käyttäjätilien haltuunottoa, ovat myös ”pahoja” botteja. Vaikka osa näistä toiminnallisuuksista on laittomia, ei bottien tarvitse rikkoa olemassa olevia lakeja ollakseen haitallisia. (CloudFlare, 2020)

Lisäksi ylenmääräinen bottiliikenne voi ylikuormittaa Internet-palvelimen resurssit, hidastaen tai pysäyttäen palveluita hyväntahtoisilta verkkosivustoa tai -palvelua käyttäviltä ihmiskäyttäjiltä. Joissain tapauksissa tämä on tarkoituksellista esimerkiksi palvelunestohyökkäyksen muodossa. (CloudFlare, 2020)

Haitallinen bottitoiminta sisältää:

- Käyttäjätunnusten testaamista eri verkkopalvelussa murtautumistarkoituksessa (Cloudflare, 2020 (2))
- Verkkosivustojen sisältöjen keräämistä ja varastoimista (Cloudflare, 2020 (3))
- Palvelunestohyökkäyksiä (CloudFlare, 2020)
- Niin kutsuttuja Brute Force -salasananmurtamisyrityksiä (CloudFlare, 2020 (4))

- Tietovarastojen sisältöjen keräämistä (Cloudflare, 2020)
- Roskapostin lähettämistä (CloudFlare, 2020)
- Sähköpostiosoitteiden keräämistä (CloudFlare, 2020)
- Niin kutsuttujen napsautuspetosten suorittamista (CloudFlare, 2020(5))

Näiden hyökkäysten suorittamiseksi ja hyökkäysten lähteen naamioimiseksi ”pahat” botit voivat esiintyä niin kutsutuissa bottiverkoissa (botnet), mikä tarkoittaa, että botista tehdyt kopiot ajetaan useissa eri laitteissa usein niin, ettei laitteen omistaja ole edes tietoinen siitä. Koska jokaisella laitteella on oma IP-osoitteensa, bottiverkkojen liikenne tulee tuhansista eri IP-osoitteista, mikä tekee haitallisen bottiliikenteen lähteen paikallistamisen hankalaksi. (Cloudflare, 2020)

Napsautuspetokset ovat tapahtumia, joissa yksilö, tietokoneohjelma tai skripti käyttää hyväkseen verkkomainostajia napsauttamalla jatkuvasti niin kutsuttua pay-per-click -mainosta aiheuttaakseen epärehellisiä laskuja uhrilleen. Napsautuspetokset nostavat mainostuskuluja, alentavat konversioarvoja ja vuotavat käyttäjätietoja verkossa toimiville yrityksille. (BigCommerce, 2020)

Napsautuspetoksia suorittavat joissain tilanteissa esimerkiksi yritysten kilpailijat. Kilpailija saattaa jatkuvasti napsautella yrityksen verkkomainoksia yrityksenään nostaa hintaa, jonka yritys maksaa kyseisestä hakutermistä. Jos napsautuspetos suoritetaan riittävän tehokkaasti, se voi jopa työntää kilpailevan yrityksen ulos markkinoilta. (BigCommerce, 2020)

Tämän lisäksi napsautuspetoksia voivat suorittaa myös mainosjulkaisijat yrityksenään ”peluuttaa” maksettua hakumainontaa. Tämä on yleistä tytäryhtiöverkoissa, joissa mainostajilla on joskus rajallinen pääsy mainoksiin liittyvään dataan. (BigCommerce, 2020)

Lopuksi asiakkaat saattavat suorittaa toimintaa, joka muistuttaa napsautuspetoksia. Tämä tapahtuu, kun yksi verkkosivuston asiakas jatkuvasti napsauttelee maksettuja hakumainoksia vieraillakseen tietyllä verkkosivustolla sen sijaan että menisi sivustolle suoraan tai hakukoneen kautta. Vaikka tämä ei perinteisessä mielessä täytä napsautuspetoksen kriteerejä,

voivat hakukoneet silti nähdä tällaisen toiminnan mahdollisesti epärehellisenä johtaen aiheutuvien napsautusten tekemiseksi sellaisiksi, etteivät ne aiheuta laskuja. (BigCommerce, 2020)

2.2 **Yritykset haitallisten bottien toiminnan estämisessä**

Bottienhallintasovellukset mahdollistavat haitallisten bottien toiminnan erottamisen ihmiskäyttäjien toiminnasta ja ”hyvien” bottien toiminnasta koneoppimisen avulla. Niiden pitäisi tunnistaa ja estää haitallisten bottien toiminta perustuen käyttäytymisanalyysiin, joka paikallistaa poikkeavuuksia ja silti sallia hyödyllisten bottien pääsy verkkosivuston toiminnallisuuksiin. (Cloudflare, 2020)

Yritysten työ haitallisilta boteilta suojautumisessa on jatkuva, monimutkainen tehtävä. Koska jokainen toimiala on erilainen, ja haitalliset botit suorittavat mitä erinäisempiä tehtäviä, ei ole olemassa yhtä, kaikille toimivaa ratkaisua. (Bozicevic, 2019)

On olemassa joitakin toimia, joita voi tehdä haitallisilta boteilta suojautumisessa:

- **Ymmärrä haavoittuvuutesi:** Yritysten on jatkuvasti arvioitava ja kehitettävä turvallisuustoimintojaan pysyäkseen hakkereiden edellä. On tärkeää ymmärtää uhkien luonne ja olla varautunut hyvällä toimintasuunnitelmalla internetissä olevien haavoittuvaisten asioiden suojelemiseksi. (Bozicevic, 2019)

- **Havaitse, kategorisoi ja hallitse:** Bottiliikenteen havaitseminen on ensimmäinen askel. Kun bottiliikennettä on paikallistettu, seuraava askel on kategorisoida bottiliikenne. Jos se on tunnettua bottiliikennettä – kuten hakukonebotit – sen kulku pitäisi sallia, mutta tunnettujen haitallisten tai tuntemattomasta lähteestä tulevien bottien liikennöinti pitäisi estää. Lopuksi haitallisten bottien liikennettä täytyy hallinnoida. (Bozicevic, 2019)

- **Arvioi mahdollinen tilanne, jossa botti aiheuttaa haitallista toimintaa:** Haitallisiin botteihin liittyvä ongelma on varustekilpaa. Pahantahtoiset toimijat työskentelevät ahkerasti joka päivä hyökätäkseen maailman eri internetsivuja vastaan. Hyökkäyksiin

> käytettävät työkalut kehittyvät jatkuvasti, haitallisten bottien liikenne alkaa toistaa samoja kuvioita ja lähteet hyökkäyksille muuttuvat; lisäksi edistyneiden bottien suorittama toiminta voi muistuttaa ihmisen suorittamaa toimintaa. Hakkerit, jotka hyödyntävät botteja hyökätäkseen sivustoasi vastaan ovat levittäytyneet ympäri maailmaa ja heidän tahtonsa hyökkäysten suorittamiseen on korkea. Entisinä aikoina oli mahdollista suojata sivustoja muutamilla hienosäädöillä, mutta nuo ajat ovat jo kauan sitten menneet. Nykypäivänä on lähes mahdotonta pysyä jatkuvasti kehittyvien uhkien perässä yksin. (Bozizevic, 2019)

Loppupäätelmänä voi sanoa, että haitalliset botit ovat kasvava uhka yrityksille maailmanlaajuisesti. Niitä on usein vaikea havaita ja niiden aiheuttama vahinko voi kaataa yrityksen liiketoiminnan. Yritysten täytyy pysyä kehittyvien trendien perässä suojatakseen itsensä ja käyttäjiensä datan haitallisten bottien hyökkäyksiltä asianmukaisesti. (Bozizevic, 2019)

3 SAAVUTETTAVUUS

Saavutettavuus on tärkeä osa muun muassa nykyaikaisten mobiilisovellusten suunnittelussa, joten se oli tärkeässä asemassa myös opinnäytetyön tuotoksena syntyneen sovelluksen suunnittelussa. (Blair, BuildFire, 2020)

Saavutettavuus on nykypäivänä tärkeässä asemassa myös oikeudellisesta näkökulmasta, sillä kaikkien viranomaisten sivustojen Internetissä tulee olla saavutettavia 23.9.2020 mennessä. Tämän säätävä niin kutsuttu digipalvelulaki vaatii, että verkkopalvelut täyttävät saavutettavuusvaatimukset, että palvelun saavutettavuuden tila esitellään savutettavuusselosteessa ja että palvelussa on sähköinen palautekanava saavutettavuuspalautteelle. Kun digipalvelulakia noudatetaan, hyötyvät tästä myös esimerkiksi vammaiset ja iäkkäät. (Saavutettavuusvaatimukset, 2020)

Saavutettavuus tarkoittaa verkkopalveluissa teknisesti virheetöntä toteutusta, selkeää ja hahmotettavaa käyttöliittymää ja ymmärrettävää sisältöä. Kun saavutettavaa verkkopalvelua suunnitellaan, huomioitavia asioita ovat juuri tekninen toteutus, helppokäyttöisyys ja sisältöjen selkeys ja ymmärrettävyys. (Saavutettavuusvaatimukset, 2020)

Tekninen saavutettavuus tarkoittaa muun muassa sitä, että verkkopalvelun lähdekoodi on virheetöntä ja loogista. Tähän kuuluu se, että HTML-standardia ja WCAG-ohjeistusta on noudatettu ja palvelua voi käyttää hyvin erilaisilla päätelaitteilla ja avustavilla tekniikoilla, kuten puheohjauksella ja ruudunlukuohjelmilla. (Saavutettavuusvaatimukset, 2020)

Suurin osa ihmisistä pystyy käyttämään verkkopalveluita ilman ongelmia. Kaikille se ei kuitenkaan ole yhtä helppoa. Esimerkiksi Suomessa on yli miljoona ihmistä, jotka tarvitsevat saavutettavuutta verkkopalveluiden käytössä. (Poutapilvi, 2019)

Yksinkertaistettuna voidaan sanoa, että saavutettavuudella tarkoitetaan helppokäyttöisempiä verkkopalveluita, jotka ovat paremmin suunniteltuja ja tehtyjä kuin tavalliset verkkopalvelut. Esimerkkejä erilaisista tavoista toteuttaa saavutettavuutta ovat sokeiden selaimet,

ruudunlukuohjelmat ja puhallusohjaus. Jotkut taas käyttävät palveluita ainoastaan näppäimistöllä, yhdellä kädellä tai silmäohjauksella. (Poutapilvi, 2019)

3.1 **Sokeat**

Useimmat sokeat ihmiset toimivat pelkästään kuulon varassa. Saavutettavuus tarkoittaa tällöin sitä, että verkkopalveluita käytetään ruudunlukuohjelmalla, joka lukee verkkopalvelun sisällön. Verkkopalvelun on tässä tapauksessa oltava koneellisesti luettava. (Poutapilvi, 2019)

Ruudunlukuohjelma ei kuitenkaan lue ainoastaan tekstiä, vaan tämän lisäksi kaiken verkkopalvelun sisällön kuvista navigaatioelementteihin. Kuvat on tästä syystä nimettävä kuvaavasti, eikä jättää nimeksi esimerkiksi XYZ321. Hiiren käyttäminen on sokeille melko haastavaa, joten verkkopalvelun on oltava käytettävissä pelkällä näppäimistöllä. (Poutapilvi, 2019)

3.2 **Muut näkövammaiset**

Näkövammaisten tarpeet ovat erilaisia kuin sokeiden tarpeet. Kun sokeille on täysin yhdentekevää, miltä verkkopalvelu näyttää, on näkövammaiselle visuaalisuuden sen sijaan oltava mahdollisimman selkeä. (Poutapilvi, 2019)

Verkkopalvelusta kannattaa tehdä näkövammaiselle rauhallinen ja ohjaava. Siihen ei kannata sisällyttää ruudun täydeltä erilaisia elementtejä. Sisällön verkkopalvelussa on edettävä lineaarisesti. (Poutapilvi, 2019)

Myös kontrastien verkkopalvelussa täytyy olla tarpeeksi vahvoja. Tällä tavalla varmistetaan, että teksti pystytään erottamaan taustasta ja muun muassa linkit muusta tekstistä. (Poutapilvi, 2019)
Myös värisokeille vaikeiden väriyhdistelmien (kuten punainen ja vihreä) käyttöä on syytä välttää. Tiettyjen toimintojen ilmaiseminen verkkopalvelussa on lisäksi osoitettava muutoin kuin pelkillä eroavilla väreillä. Linkit esimerkiksi eivät voi vain olla eri värisiä kuin muu teksti vaan ne pitää lisäksi erotella toisella tavalla. (Poutapilvi, 2019)

3.3 **Fyysiset haasteet ja motoriset häiriöt**

Fyysisiä ja motorisia haasteita voi kenelle tahansa ilmaantua pitkin päivää. Verkkopalveluita käytetään monissa erilaisissa tilanteissa: puhelin otetaan taskusta esille samalla, kun kävellään kadulla kiireessä keskellä kirkasta päivää tai kun illalla harjataan unisena hampaita. Aina ei ole mahdollista käyttää kahta kättä, eikä sorminäppäryyskään aina ole riittävä. (Poutapilvi, 2019)

Klikattavien alueiden on oltava tarpeeksi isoja, jotta verkkopalvelu olisi saavutettava. Niiden koon tulee olla niin suuri, ettei niihin osumiseen tarvita sorminäppäryyttä, vaan se onnistuu helposti. Verkkopalvelua tulisi pystyä käyttämään myös pelkällä näppäimistöllä. Sormi- tai hiirinäppäryyden ei tulisi olla vaatimus palvelun käytölle. (Poutapilvi, 2019)

3.4 **Kuulovammaiset**

Kuuron henkilön äidinkieli on viittomakieli eikä suomi. Tästä syystä monet sanonnat ja puheessa käytetyt ilmaisut ovat kuuroille ihmisille vieraita. Sanontoja ja puheessa käytettyjä ilmaisuja tulisi tämän vuoksi välttää ja pyrkiä tuottamaan selkokieltä. (Poutapilvi, 2019)

Huomioitavaa on myös se, ettei saavutettavan sivuston minkään sisällön tulisi olla ainoastaan äänimuodossa. Videoissa tulee olla tekstitykset, jotta niissä oleva tieto pystytään välittämään kuurolle henkilölle. Saavutettavassa verkkopalvelussa kannattaa käyttää kuvailevia otsikoita ja loogisesti eteneviä otsikkotasoja. (Poutapilvi, 2019)

3.5 **Lukihäiriöiset**

Lukihäiriöiselle henkilölle pitkät ja vaikeaselkoiset tekstimassat tuottavat ongelmia. Tästä syystä kannattaa pyrkiä tuottamaan tekstiä, joka on helposti ymmärrettävissä, hahmotettavissa ja luettavissa. Teksti kannattaa tasata vasempaan reunaan. Fontin tulisi olla helppolukuista. Sisältöä kannattaa lisäksi elävöittää kuvilla, jotka tukevat verkkopalvelun sisältöä. Ennen kaikkea verkkosivun sisältö kannattaa pitää lyhyenä, selkeänä ja yksinkertaisena. Käytännössä tämä tarkoittaa lyhyitä sanoja, lauseita ja kappaleita. (Poutapilvi, 2019)

3.6 **Muut saavutettavuutta tarvitsevat sekä iäkkäät**

Saavutettavuus helpottaa myös sellaisten ihmisten Internetin käyttöä, jotka eivät välttämättä tarvitsisi saavutettavuuteen liittyviä ominaisuuksia. Esimerkkeinä tästä ovat mobiililaitteiden ja muiden laitteiden käyttäjät, joiden laitteiden ruudun koko ja syötetavat eroavat perinteisestä tietokoneen ruutukoosta ja tavasta antaa syötteitä, vanhemmat ihmiset, joiden kyvyt käyttää laitteita ovat muuttuneet iän myötä, käyttäjät, joilla on jokin väliaikainen vamma tai puutos, kuten murtunut käsi tai kadonneet silmälasit, käyttäjät, joilla on tilanteeseen liittyviä rajoitteita, kuten paikka, jossa ei voi kuunnella äänipohjaisia aineistoja ja käyttäjät, joilla on hidas Internet-yhteys tai joiden Internet-yhteyden kaista on rajattu tai Internet-yhteys on kallis. (W3, 2020)

3.7 **Esteettisyys verkkopalveluissa**

Esteettisyys tarkoittaa periaatteita, jotka ohjaavat taiteilijan työtä sisältäen värit, kontrastin, grafiikan ja ulkoasun. Verkkosivuston esteettisyys voi vaikuttaa tuotemerkin uskottavuuteen ja asiakkaiden vastaanottoon. Tutkimukset ovat itse asiassa osoittaneet, että hyvän suunnittelun ja verkkosivuston uskottavuuden välillä on yhteys. Lisäksi suunnitteluperiaatteiden noudattaminen vaikuttaa verkkosivuston yleiseen käyttökokemukseen ja määrittää verkkosivuston suunnittelijan taidon tason. Hyvältä näyttävä ja fiksu suunnittelu antaa verkkosivuston käyttäjälle mahdollisuuden positiiviseen kokemukseen. (Martin, 2020)

Suurin osa ihmisistä nykypäivänä arvostaa käytettävyyttä enemmän kuin hyvää ulkomuotoa, kun puhutaan verkkosivustoista. On selvää, että hyvä käytettävyys on välttämätöntä, jotta verkkosivusto olisi onnistuneesti tehty. Verkkosivuston pitäisi antaa kävijöille mahdollisuuden ymmärtää tuotteitasi tai palveluasi niin nopeasti ja helposti kuin mahdollista. Verkkosivustojen käyttäjät eivät enää hyväksy sivustoja, jotka latautuvat hitaasti tai joissa on vaikea navigoida. Verkkosivuston käytettävyys voi jopa vaikuttaa onnistumiseesi verkkomaailmassa. (Martin, 2020)

Verkkopalveluita suunniteltaessa esteettisesti hyvät verkkosivut jättävät yleensä kävijöihinsä hyvän vaikutelman. Ensivaikutelma verkkosivustosta on tärkeä, koska sen perusteella kävijät joko jäävät sivustolle tai lähtevät sivustolta ja tämä päätös tehdään usein vain sekunneissa.

Visuaalisesta esteettisyydestä huolimatta on kuitenkin myös sivun hyvä toiminnallisuus pidettävä mielessä. (DigitalHill, 2020)

Jos verkkosivuillasi myydään tuotetta tai palvelua, voi hyvältä näyttävä sivusto jopa muuttaa selailevat kävijät maksaviksi asiakkaiksi. Tämän ennakkoehtona on kuitenkin se, ettei sivuston hyvä esteettisyys tee siitä vaikeasti käytettävää. Vaikka verkkosivusto olisi esteettisessä mielessä täydellinen, eivät kävijät jää sivustolle, jos se on vaikeasti käytettävä mobiililaitteilla, sisältää epäintuitiivisen ulkoasun, sisältää sivuja, jotka latautuvat hitaasti tai sisältää toimimattomia linkkejä, liikaa sivuston vierittämistä tai muita ongelmia tai virheitä. (DigitalHill, 2020)

Verkkosivustojen käyttökokemussuunnittelussa (UX design) on tärkeää muistaa, että estetiikkaa ja toiminnallisuutta ei tule erottaa toisistaan. Verkkosivuston estetiikka sisältää sen visuaalisen suunnittelun ja halutun houkuttelevuuden. Toiminnallisuus taas tarkoittaa sivuston ominaisuuksia ja toimintoja. Toiminnallisuutta ja estetiikkaa tulisi verkkosivustojen suunnittelussa pitää samanarvoisina. (DigitalHill, 2020)

Vaikka verkkosivustot on kirjoitettu näennäisesti luonnottomilla kielillä, kuten HTML, on muistettava, että sivuston loppukäyttäjät ovat ihmisiä. Ennen julkaisuaan verkkosivustoa täytyy testata. Näin tulisi tehdä jokaisen sivustolle tehdyn muutoksen jälkeen. Testausprosessin aikana huomioitavia asioita ovat yhdenmukaisuus, eli se, että sivuston elementtien korrelaatio on toimivaa, se, että sivusto on luonnollisen tuntuinen, eli että se ei esimerkiksi sisällä jäykkyyttä toiminnallisuudessa tai teräviä, suurikontrastisia kulmia ja se, että verkkosivuston ulkoasu on yleisestikin ottaen järkevä eli että jokainen elementti, painike ja sovellus sijaitsevat paikoissa, joista käyttäjä todennäköisimmin etsisi niitä, koska intuitiiviset prosessit verkkosivustoilla ovat avainasemassa. (DigitalHill, 2020)

Julkisesti saatavilla olevia verkkosivustoja on paljon (yli miljardi), joten jos sen kävijöillä on epäselvyyttä sen käyttötarkoituksen suhteen, on hyvin todennäköistä, että he eivät jää sivustolle pitkäksi aikaa. Asioita, jotka kasvattavat ensimmäistä kertaa verkkosivustolla käyvien henkilöiden todennäköisyyttä muuttua maksaviksi asiakkaiksi, ovat hyvin näkyvät, korkean resoluution kuvat, jotka liittyvät sivustosi tarjoamiin tärkeimpiin tuotteisiin, selkeä,

asianmukaisen kokoinen ja luettava teksti, käyttökelpoinen, hyvin organisoitu ulkoasu, joka tekee sivustolla liikkumisesta intuitiivista ja sulavaa, selkeästi esillä olevat niin kutsutut call to action -painikkeet ja -ominaisuudet, kuten verkkokauppa tai katalogi, tai alue, josta voi tilata sivuston uutiskirjeen ja yksinkertaisen mutta selkeän tarjousominaisuuden sisällyttäminen sivustolle. (DigitalHill, 2020)

Suurin syy verkkosivustojen julkaisemiselle on saada sen käyttäjät suorittamaan tietyntyyppistä toimintaa. Verkko- ja käyttökokemussuunnittelu eivät sisällä ainoastaan väreihin, varjostuksiin ja animaatioihin liittyviä asioita vaan myös käytettävyys, mobiilivasteellisuus ja sivuston tarkoitus ovat yhtä tärkeässä asemassa. Jos verkkosivusto on esteettisesti tyylikäs mutta käyttäjät eivät viivy sillä kauaa, on syytä uudelleenarvioida sivuston toiminnallisuus, jotta sivusto olisi sekä esteettisesti tyylikäs että helposti käytettävä. (DigitalHill, 2020)

3.8 **Loppuajatukset saavutettavuudesta**

Yhteiskunta digitalisoituu kovaa vauhtia, joten on tärkeää, että varsinkin tärkeät verkkopalvelut ovat saavutettavia, jotta mahdollisimman moni pystyy käyttämään niitä helposti. Internetin sivustoilla on edelleen saavutettavuuspuutteita, selviää vuonna 2019 tehdystä Euroopan laajuisesta kyselystä liittyen verkkopalvelujen saavutettavuuteen. Tehty kysely kohdistui uusiin verkkosivustoihin, jotka kuuluvat saavutettavuusvaatimusten piiriin. Näistä palveluista vain 20 % täytti saavutettavuusvaatimukset. (Saavutettavuusvaatimukset, 2020)

Pääsyy saavutettavuuden puuttumiseen oli heikko tietoisuus saavutettavuuteen liittyvistä vaatimuksista sekä organisaatioiden verkkosivustojen laajuus ja rajalliset resurssit saavutettavuuteen liittyvän työn tekemiseksi. Tilastoja suomalaisten verkkopalveluiden saavutettavuudesta ei esimerkiksi ole toistaiseksi saatavilla, koska valvontakausi on vielä kesken. Saavutettavuusvalvonnan yksikön päällikkö Emilia Ojala kertoo, että saavutettavuuslain vaatimuksia ei vielä tunneta riittävän hyvin, jotta ne osattaisiin täyttää asianmukaisin tavoin. (Saavutettavuusvaatimukset, 2020)

4 KÄYTETYT TEKNISET MENETELMÄT

Käytettyihin teknisiin menetelmiin kuuluvat REST API, HTTP, JSON, Rasa, HTML, CSS ja PHP. Näitä menetelmiä käytettiin, koska opinnäytetyön tuloksena syntynyt sovellus käyttää REST API -tekniikan GET-metodia tiedon hakemiseen palvelimelta, JSON-tietomuotoa palvelimelta saapuvan tiedon tietomuotona, Rasaa viitteenä etsiessäni tietoja boteista sekä niiden toiminnallisuudesta, HTML:ää esimerkkinä siitä, millaista kieltä verkkosivujen rakentamisessa käytetään, CSS:ää esimerkkinä siitä, miten verkkosivustoja voidaan tyylitellä ja PHP:tä esimerkkinä siitä, mitä kieliä voi käyttää palvelimella dynaamisesti käsiteltävän tiedon käsittelyyn.

4.1 Rest API

Rest API on ohjelmointirajapinta, joka käyttää HTTP-pyyntöjä voidakseen hyödyntää HTTP-metodeja GET, PUT, POST ja DELETE. GET-metodia käytetään resurssin hakemiseen, PUT-metodia käytetään muuttamaan tai päivittämään resurssi, joka voi olla olio, tiedosto tai blokki(koodilohko), POST-metodia käytetään resurssin luomiseen ja DELETE-metodia käytetään poistamaan resurssi. Rest API, johon viitataan myös nimellä Rest-webpalvelu, perustuu REST-teknologiaan, joka on arkkitehtuurinen tyyli ja lähestymistapa verkkopalveluissa käytettävissä yhteyksissä. REST-teknologiaa suositaan yleensä enemmän kuin Simple Object Access Protocol -protokollaa (SOAP), koska se vie vähemmän kaistaa, mikä tekee siitä sopivan Internetin käyttöä varten. (TechTarget, 2016)

REST API:n (REpresentational State Transfer Application Programming Interface) esitteli ensimmäisenä Roy Fielding vuonna 2000 kuuluisassa julkaisussaan. Jotta web-palvelu olisi Rest Api:n ehdot täyttävä palvelu, sen täytyy täyttää muutamia ehtoja. Näitä ovat:

Asiakas-palvelin-malli: Kun käyttöliittymään ja tietovarastoihin liittyvät asiat erotellaan selkeästi toisistaan, parannetaan käyttöliittymän siirrettävyyttä eri alustojen välillä skaalautuvuutta yksinkertaistamalla palvelinkomponentteja.

Tilattomuus: Jokaisen pyynnön asiakkaalta palvelimelle tulee sisältää kaikki tarvittava tieto pyynnön ymmärtämiseksi, eikä pyyntöjen tule hyödyntää palvelimelle tallennettuja tietoja. Tästä seuraa, että istuntojen tilat säilytetään yksinomaan asiakkaalla.

Välimuistiin tallentamisen mahdollisuus: Välimuistiin liittyvät rajoitukset vaativat, että palvelimelta asiakkaan pyyntöön tulevan vastauksen sisältämä data on implisiittisesti tai eksplisiittisesti merkitty välimuistiin tallennettavaksi tai ei-välmuistiin-tallennettavaksi. Jos vastaus on välimuistiin tallennettava, annetaan asiakaspuolen välimuistille oikeus uudelleen käyttää kyseistä vastausdataa myöhempiin, vastaavanlaisiin pyyntöihin.

Yhtenäinen rajapinta: Kun sovelletaan yleistävää toimintamallia sovellusten kehittämisessä liittyen komponenttien rajapintoihin, yleinen järjestelmän arkkitehtuuri yksinkertaistuu ja vuorovaikutusten näkyvyys parantuu. Jotta saavutettaisiin yhtenäinen rajapinta, tarvitaan monia arkkitehtuurisia rajoituksia ohjaamaan komponenttien käyttäytymistä. REST määritellään neljällä rajapintarajoituksella: resurssien identifiointi, resurssien muokkaaminen käyttäen representaatioita, itseään kuvaavat viestit ja hypermedian käyttö sovellusten tilan pohjana.

Kerroksiin perustuva järjestelmä: Kerrostettu järjestelmän tyyli sallii REST-arkkitehtuurin koostumisen hierarkkisista kerroksista rajoittaen komponenttien käyttäytymistä siten, että mikään komponentti ei ”näe” seuraavan, vuorovaikutuksessa olevan kerroksen, yli.

Koodia tarpeen mukaan (valinnainen): REST sallii asiakaspuolen toiminnallisuuden laajentamisen lataamalla ja suorittamalla koodia applettien ja skriptien muodossa. Tämä yksinkertaistaa asiakaspuolta vähentämällä valmiiksi vaadittujen ominaisuuksien määrää. (Restfulapi.net, 2020)

4.2 **HTTP**

HTTP (HyperText Transfer Protocol) on sovellustason protokolla hypermedian, kuten esimerkiksi HTML:n, lähettämiseen. Se suunniteltiin verkkoselainten ja web-palvelimien väliseen kommunikointiin, mutta sitä voidaan käyttää myös muihin tarkoituksiin. HTTP noudattaa klassista asiakas-palvelin-mallia, jossa asiakas avaa yhteyden pyynnön tekemiseksi ja odottaa

tämän jälkeen vastausta palvelimelta. HTTP on tilaton protokolla eli se ei säilytä dataa eri pyyntöjen välillä. Vaikka HTTP usein toimii TCP/IP-kerroksessa, voidaan sitä käyttää missä tahansa luotettavassa kuljetuskerroksessa, eli kerroksessa, joka ei kadota viestejä, kuten esimerkiksi UDP. (Mozilla, 2020)

Viestien lähettäminen HTTP:ssä tapahtuu yleensä TCP/IP(Transmission Control Protocol/Internet Protocol)-yhteyksien yli. Oletusportti HTTP:lle on 80, mutta myös muiden porttien käyttäminen on mahdollista. Tämä ei estä HTTP:n käyttämistä minkä tahansa muun protokollan päällä Internetissä tai muissa verkoissa. (IETF, 2020)

4.3 **JSON**

JSON (JavaScript Object Notation) on kevyt ohjelmointikielestä riippumaton tiedon välitysformaatti, jota ihmisen on helppo ymmärtää. Koneiden on lisäksi helppo kääntää ja tuottaa sitä. JSON perustuu JavaScript-ohjelmointikielen standardiin ECMA-262, versio 3, joulukuulta 1999. JSON on tekstiformaatti, joka on täysin ohjelmointikieliriippumaton, mutta joka käyttää käytäntöjä, jotka ovat tuttuja C-perheen ohjelmointikielille, kuten C, C++, C#, Java, JavaScript, Perl, Python ja moni muu. Nämä ominaisuudet tekevät JSONista ideaalin datankäsittely ja -lähetyskielen. (JSON, n.d.)

JSON koostuu nimi-arvo -pareista, jotka erotetaan toisistaan käyttäen pilkkua (,). Aaltosulut pitävät sisällään objekteja, joiden ensimmäisen osan (nimen) jälkeen käytetään kaksoispistettä (:). JSON on XML:ää vähäsanaisempaa, joten JSONin kirjoittaminen on ohjelmoijille nopeampaa kuin XML:n kirjoittaminen. (Tutorialspoint, n.d.)

Esimerkki JSONista:

```
{
“henkilo”: {
        ”nimi” : ”John”,
         ”ika” : 30,
         ”kaupunki”: ”New York”
         }
}
```

Koodiesimerkki 1, Esimerkki JSONista

Sama esimerkki käyttäen XML:ää:

```
<henkilo>
    <nimi>John</nimi>
    <ika>30</ika>
    <kaupunki>New York</kaupunki>
</henkilo>
```

Koodiesimerkki 2, esimerkki XML:stä

JSON:n tiedostopääte tiedostojärjestelmässä on .json ja sen niin kutsuttu IANA (Internet Assigned Names Authority) -media (tai MIME) -tyyppi on application/json. JSON-tiedosto voi sisältää tekstiä, aaltosulkeita, neliösulkeita, kaksoispisteitä, pilkkuja, lainausmerkkejä ja mahdollisesti joitain muitakin merkkejä. (REST API Tutorial, 2020)

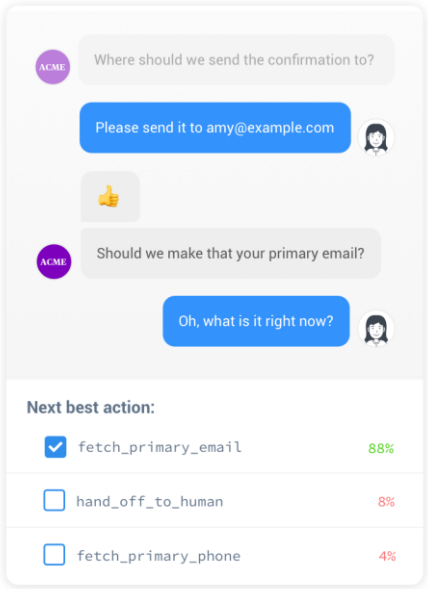
Pääasiallisesti JSON-dokumentti voi sisältää kahden tyyppisiä rakenteita:

1. Olio, jota ympäröivät aaltosulkeet ja joka sisältää useita nimi/arvo pareja. Monissa ohjelmointikielissä tämä voidaan nähdä record-, struct-, dictionary-, hash table-, keyed list-, tai associative array-tyyppisenä tietona. (REST API Tutorial, 2020)

2. Taulukko tai niin kutsuttu ordered list arvoista, jotka on ympäröity neliösulkeilla. Monissa ohjelmointikielissä tämä tarkoittaa vektoria, listaa tai sarjaa. (REST API Tutorial, 2020)

4.4 **Rasa**

Rasa on tekoälyä hyödyntävä chat-bottien tekemiseen käytetty viitekehys. Sitä käytetään keskustelubottien tekemiseen. Sen vastaukset keskusteluihin perustuvat aikaisemmista keskusteluista opittuun tietoon eli sen toiminta ei perustu ainoastaan ennalta määritettyihin if/else -lauseisiin. (Rasa, 2019)



Kuva 1. Esimerkki Rasan toiminnasta.

Rasa:n ja Pythonin avulla voi kehittää monipuolisia chat-botteja. Rasan käyttämä NLU (Natural Language Understanding) -ominaisuus on osa Rasaa, joka huolehtii halutun toiminnallisuuden luokittelun, entiteettien käsittelyn ja vastauksen noutamisen. NLU ottaa lauseen, kuten ”I am looking for a French restaurant in the center of town” ja palauttaa seuravaanlaista strukturoitua dataa:

```
{
          “intent”: “search_restaurant”,
          “entities”: {
          “cuisine”: “French”,
          “location”: “center”
    }
}
```

NLU-mallien rakentaminen on haastavaa ja vielä haastavampaa on rakentaa tuotantovalmiita NLU-malleja. Siinä kannattaa ottaa seuraavat asiat huomioon:

Oikeisiin keskusteluihin perustuvaa kehitystä (Conversation-Driven Development, CDD): Tämä tarkoittaa, että oikeat keskustelut ohjaavat botin kehitysprosessia. Tätä varten on kannattavaa kerätä dataa oikeista ihmisten välisistä keskusteluista. Vaikka botti tekee alussa virheitä tätä lähestymistapaa käytettäessä, on botin jatkokehitys ja tulevien keskustelujen vastausten yleistäminen helpompaa, kun käytetään tätä lähestymistapaa.

Kerää dataa keskusteluista kehitystiimin ulkopuolelta mahdollisimman aikaisessa vaiheessa: Mahdollisimman aikaisessa vaiheessa tapahtuva botin kehitystiimin ulkopuolella käytävien keskustelujen tietojen kerääminen on kannattavaa, koska on vaikea arvata, millaisia vastauksia eri ihmiset antavat eri tilanteissa. (Rasa, 2020)

4.5 **HTML**

HTML(Hyper Text Markup Language) on standardin mukainen merkintäkieli verkkosivujen luomiseen. Se kuvaa verkkosivujen rakennetta merkintäkieltä käyttäen. HTML-elementit ovat HTML-sivujen tekemiseen käytettyjä komponentteja. Ne esitetään käyttäen niin kutsuttuja tageja(”tags”). HTML-tagit nimeävät verkkosivujen osia, kuten esimerkiksi otsikko (”heading”), kappale (”paragraph”), taulukko(”table”) ja niin edelleen. Selaimet eivät näytä HTML-tageja mutta käyttävät niitä verkkosivujen sisällön tuottamiseen. (W3Schools.com, 2019)

HTML:n yhteydessä voidaan myös käyttää erilaisia attribuutteja tagien nimien sisällä, esimerkiksi ” Tämä on kappale . Tällöin voidaan HTML-elementtejä

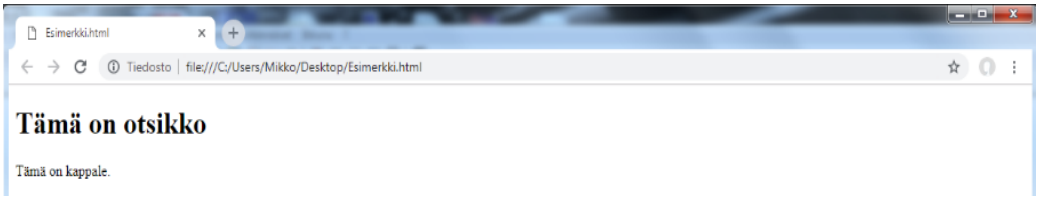
tyylitellä CSS:ssä class-määreiden avulla(.). Myös id:n (#) käyttö on mahdollista, jos halutaan kohdentaa tyylejä hyvin tarkasti yhteen HTML-elementtiin. (Mozilla, 2020)

Esimerkki HTML-sivusta:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Tämä on otsikko.</h1>
<p>Tämä on kappale</p>
</body>
</html>
```

Koodiesimerkki 3, esimerkki HTML:stä.

Edeltävä koodi verkkoselaimessa:



Kuva 2. Muotoilematon HTML-sivu.

4.6 **CSS**

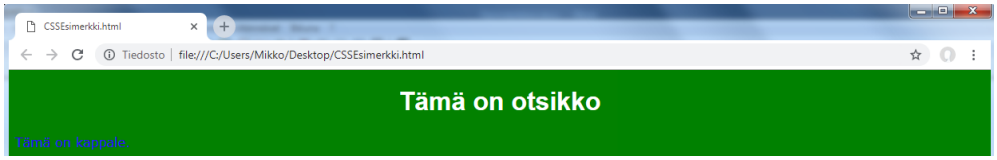
CSS(Cascading Style Sheets) kuvaa kuinka HTML-elementit tulostetaan ruudulle, paperille tai muulle medialle. Se säästää paljon työtä, koska sillä voi kuvata useita verkkosivuja samanaikaisesti. Ulkoiset CSS-tiedostot tallennetaan .css-muodossa. (W3Schools.com 2019)

Esimerkki CSS:llä muotoillusta verkkosivusta:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
          background: green;
}
h1 {
         color: white;
         font-family: arial;
         text-align: center;
}
p {
        color: blue;
        font-family: verdana;
}
</style>
</head>
<body>
<h1>Tämä on otsikko</h1>
<p>Tämä on kappale.</p>
</body>
</html>
```

Koodiesimerkki 4, esimerkki HTML-sivusta, joka on muotoiltu CSS:llä.

Edeltävä koodi verkkoselaimessa:



Kuva 3. CSS:llä muotoiltu HTML-sivu.

CSS:llä voidaan myös tyylitellä elementtejä perustuen class- tai id-määreisiin, joista mainittiin aikaisemmin. Tällöin käytetään CSS:ssä pistettä (.) tietyn class-määreen omaavan elementin tyylittelyyn ja risuaitaa (#) tietyn id-määreen omaavan elementin tyylittelyyn.

```
<!DOCTYPE html>
<html>
<head>
<link rel=”stylesheet” href=”style.css”>
</head>

<body>

<h1>Tämä on otsikko ilman tyylimäärityksiä</h1>
<h1 class=”vihreä-teksti”>Tämä on otsikko tyylimäärityksien kanssa</h1>

<p>Tämä on kappale ilman tyylimäärityksiä.</p>
<p id=”sininen-teksti”>Tämä on kappale tyylimäärityksien kanssa.</p>

</body>
</html>
```

Koodiesimerkki 5, esimerkki HTML-sivusta, jossa on class- ja id-määreitä käyttäviä element-tejä.

```
.vihreä-teksti {
          color: green;
}

#sininen-teksti {
          color: blue;
}
```

Koodiesimerkki 6. Edellisen koodiesimerkin tyylitiedosto, joka on kirjoitettu CSS:llä (style.css).

4.7 **PHP**

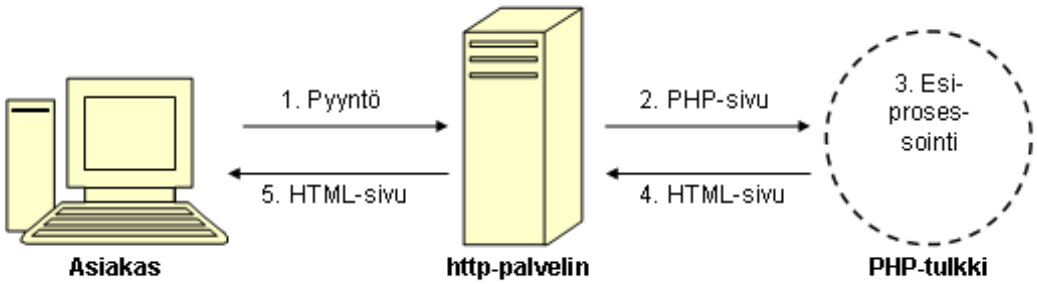
PHP on palvelinpuolen ohjelmointikieli, jota käytetään monipuolisten ja dynaamisten verkkosivujen teossa. PHP on laajasti käytetty, ilmainen, ja tehokas vaihtoehto verrattuna kilpailijoihinsa, kuten esimerkiksi Microsoftin ASP:hen.
PHP 7 on uusin vakaa versio PHP:stä. (W3Schools.com, 2019)

PHP muistuttaa syntaksiltaan C-kieltä, Javaa ja Perliä. PHP:llä on mahdollista tehdä myös komentorivisovelluksia ja jopa graafisia käyttöliittymiä, mutta se on parhaimmillaan web-sovelluskehityksessä. (PHP:n perusteet, n.d.)

PHP on lisenssitön open source -tuote eli kuka tahansa voi käyttää sitä maksutta myös kaupallisiin tarkoituksiin. PHP-tuki on saatavilla kaikkiin yleisimpiin web-palvelimiin. Tämän lisäksi PHP on saatavilla myös useille eri käyttöjärjestelmille. (PHP:n perusteet, n.d.)

PHP sisältää muun muassa tuen useimmille yleisesti käytetyille tietokannoille. Se sisältää myös olio-ominaisuuksia, mutta yleisesti käytössä olevissa versioissa ne ovat vielä jossain määrin puutteelliset. (PHP:n perusteet, n.d.)

Tavallisessa tapauksessa asiakkaan pyytäessä palvelimelta staattista HTML-sivua, palvelin vain yksinkertaisesti palauttaa asiakkaalle halutun sivun. PHP-sivuilla toiminnallisuuden suorittava koodi kirjoitetaan HTML:n sekaan. Sivut nimetään yleensä .php-päätteisiksi, mutta joskus näkee käytettävän esimerkiksi päätteitä .php3 tai .php4, jotka ilmaisevat PHP:n version. HTTP-palvelimen asetuksissa on mahdollista määritellä, minkä päätteen omaavat tiedostot lähetetään PHP-tulkin käsiteltäviksi. Kun PHP-sivua kutsutaan, palvelin toimii seuraavan kuvan mukaisesti. (PHP:n perusteet, n.d.)

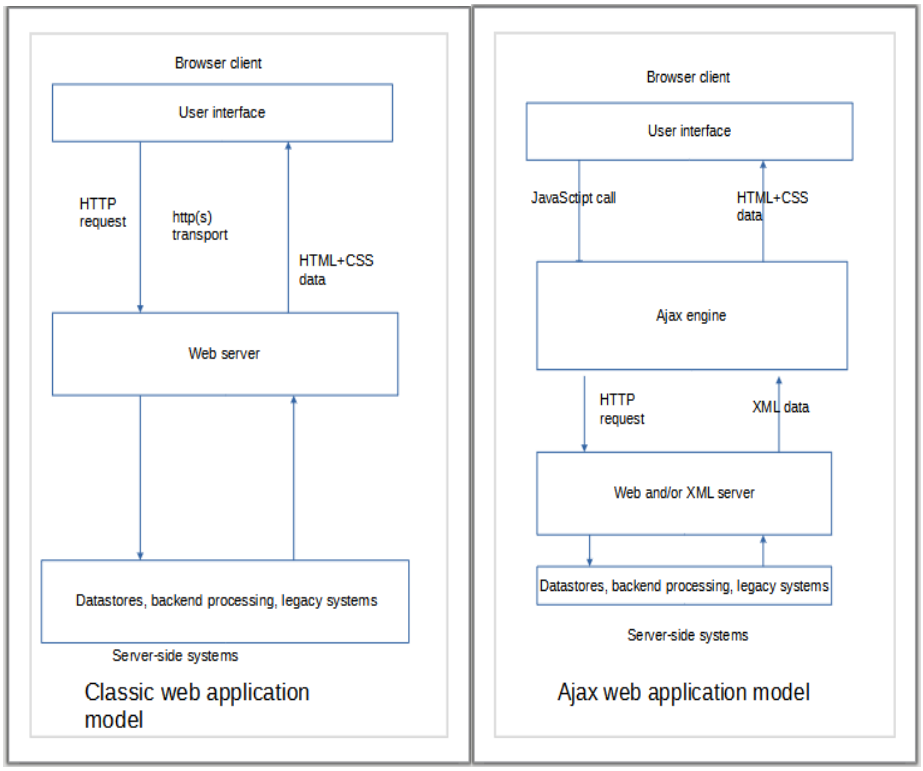


Kuva 4. PHP:n toiminta (PHP:n perusteet, Jyväskylän yliopisto, 2020)

Http-palvelin tunnistaa palvelua pyydettäessä tiedoston päätteen perusteella, että käsiteltävä sivu on PHP-sivu. Tässä tapauksessa palvelin välittää ensimmäisenä PHP-tulkille pyynnön esiprosessoida pyydetty sivu. Tämän jälkeen PHP-tulkki kääntää ja suorittaa tiedoston sisältämän PHP-koodin. Lopputuloksena PHP-tulkki palauttaa http-palvelimelle ainoastaan HTML-koodia. Lopuksi http-palvelin palauttaa asiakkaalle lopullisen HTML-koodin. PHP on olemassa siis ainoastaan palvelimella eli asiakas ei näe sitä missään vaiheessa. (PHP:n perusteet, n.d.)

4.8 **Ajax**

Ajax(Asynchronous Javascript and XML) on joukko web-tekniikoita, jotka toimivat yhdessä. Näihin tekniikoihin kuuluvat XHTML ja CSS, DOM(Document Object Model), XML ja XSLT, asynkroninen datan noutaminen käyttäen XmlHttpRequest-objektia ja JavaScript. Klassinen websovellus toimii seuraavasti: useimmat käyttäjän suorittamat toiminnot käyttöliittymässä lähettävät HTTP-pyynnön palvelimelle. Palvelin käsittelee tietoa – se hakee dataa, suorittaa laskutoimituksia ja keskustelee erilaisten järjestelmien kanssa – ja palauttaa HTML-sivun käyttäjälle. (Jason Furnell, Adaptive Path, 2008)



Kuva 5. Klassinen webpalveluiden toiminta vs. AJAX-pohjainen webpalveluiden toiminta. (Furnell, Adaptive Path, 2008)

Kun klassisessa webpalveluiden toimintamallissa on teknisesti järkeä, ei se ole kovin käyttäjäystävällinen. Kun palvelin suorittaa tehtäväänsä, joutuu käyttäjä odottamaan tehtävän valmistumista; ja tehtävien lisääntyessä joutuu käyttäjä odottamaan yhä pidempään. (Furnell, Adaptive Path, 2008)

AJAX-pohjainen websovellus eliminoi aloita-lopeta-aloita -luontoisen tiedon siirtämisen hyödyntämällä niin kutsuttua AJAX-moottoria käyttäjän ja palvelimen välillä. Ylimääräisen komponentin lisäämisen voisi ajatella tekevän webpalvelusta vähemmän responsiivisen, mutta asia on päinvastoin. Sen sijaan, että webpalvelussa ladattaisiin uuden session alkaessa verkkosivu uudestaan, ladataan AJAX-moottori, joka on kirjoitettu JavaScriptillä. Tämä moottori on vastuussa sekä verkkosivun käyttöliittymän renderöimisestä että keskustelemisesta palvelimen kanssa. AJAX-moottori mahdollistaa asynkronisen vuorovaikutuksen käyttäjän ja

palvelimen välillä riippumatta palvelimen kanssa käytävästä keskustelusta. (Furnell, Adaptive Path, 2008)

AJAX-pohjaisissa websovelluksissa jokainen käyttäjän suorittama toiminto, joka yleensä aiheuttaisi HTTP-pyynnön lähettämisen, muutetaan JavaScript-kutsuksi AJAX-moottorille sen sijaan. Kaikki vastaukset palvelimelta käyttäjälle, jotka eivät vaadi keskustelua palvelimen kanssa – kuten yksinkertainen datan validointi, muistissa olevan datan muokkaaminen ja jopa osa navigaatiosta – suoritetaan AJAX-moottorissa. Jos AJAX-moottori tarvitsee jotain tietoa palvelimelta vastatakseen palvelimelta tulevaan kyselyyn – oli se sitten tiedon lähettämistä käsiteltäväksi, käyttöliittymän koodin lataamista, tai uuden tiedon hakemista – se suorittaa nämä pyynnöt asynkronisesti, yleensä käyttäen XML:ää, häiritsemättä käyttäjän vuorovaikutusta websovelluksen kanssa. (Furnell, Adaptive Path, 2008)

4.9 **Android**

Lokakuussa 2003, paljon ennen kuin termi ”älypuhelin” yleistyi, ja useita vuosia ennen kuin Apple esitteli ensimmäisen iPhonensa ja iOS-käyttöjärjestelmänsä, yritys nimeltä Android perustettiin Palo Altoon, Kaliforniaan. Sen neljä perustajaa olivat Rich Miner, Nick Sears, Chris White ja Andy Rubin. Rubin sanoi yrityksen perustamisen aikaan, että tämä uusi yritys nimeltä Android kehittäsi ”älykkäämpiä mobiililaitteita, jotka ovat tietoisempia omistajansa sijainnista ja mieltymyksistä. Vaikka tämä kuulostaa älypuhelimen perusominaisuudelta, Rubin paljasti vuonna 2013 Tokiossa pitämässään puheessa, että Android-käyttöjärjestelmä oli alun perin tarkoitettu parantamaan digikameroiden käyttöjärjestelmiä. Android-niminen yritys piti myyntipuheita sijoittajille vuonna 2004, jotka näyttivät kuinka Android-käyttöjärjestelmä, joka oli asennettu digikameraan, muodostaisi langattomasti yhteyden tietokoneeseen. Tämän jälkeen tietokone muodostaisi yhteyden ”Android-datakeskukseen”, jonne kameran omistajat voisivat säilöä valokuvansa Internetin pilvipalvelimelle. (Callaham, 2019)

Android-työryhmä ei aluksi ajatellut valmistavansa käyttöjärjestelmää, joka toimisi kokonaisen mobiilikäyttöjärjestelmän pohjana. Mutta jo vuonna 2003 digikameroiden markkinat olivat laskussa ja yritys nimeltä Android päätti ruveta käyttämään käyttöjärjestelmäänsä matkapuhelimissa. (Callaham, 2019)

Vuonna 2005 seuraava merkittävä luku Androidin historiassa kirjoitettiin, kun Google osti alkuperäisen Android-yrityksen. Rubin ja muut perustajajäsenet jäivät jatkokehittämään käyttöjärjestelmää uuden omistajan alaisuudessa. Päätös käyttää Linuxia Androidin pohjana tehtiin, ja tämä tarkoitti sitä, että myös Android itsessään oli mahdollista tarjota kolmannen osapuolen matkapuhelinvalmistajille ilmaiseksi. Google ja Android-ryhmä uskoivat voivansa ansaita rahaa tarjoamalla muita käyttöjärjestelmän käyttämiä palveluita, kuten sovelluksia. (Calllaham, 2019)

4.10 **Android Studio**

Android Studio on virallinen IDE (Integrated Development Environment) sovelluskehitykseen Androidilla. IDE on ohjelmisto, jonka avulla ohjelmoidaan ja testataan ohjelmistojen koodia. Android Studion perustana on Java IDE nimeltä IntelliJ Idea. Tukeakseen sovelluskehitystä Androidilla Android Studio käyttää Gradle-pohjaista järjestelmää, emulaattoria, koodipohjia ja GitHub-integraatiota. Gradle on avoimen lähdekoodin testausautomaatiotyökalu ja GitHub on Git-versionhallintaa käyttävien ohjelmistojen verkossa toimiva varastointipalvelu. Android Studiolla tehdyt ohjelmat muunnetaan APK-formaattiin Google Play Storeen lähettämistä varten. (Gradle User Manual, 2019; GitHub Guides, 2019)

Android Studio julkaistiin joulukuussa 2014. Se on saatavilla Mac-, Windows- ja Linux-alustoille ja on ladattavissa suoraan Googlen sivuilta. (Tech Target, 2018)

4.11 **Java**

Java on vuonna 1995 kehitetty suosittu Oraclen omistama ohjelmointikieli. Se on käytössä yli kolmessa miljardissa laitteessa. Sitä käytetään mobiiliapplikaatioiden tekemiseen (erityisesti Android), työpöytäsovelluksiin, web-sovelluksiin, web-palvelimiin ja sovelluspalvelimiin, tietokantayhteyksiin ja paljon muuhun. Syitä opetella Javaa ovat se, että Java toimii eri alustoilla (Windows, Mac, Linux, Raspberry Pi jne), se että se on yksi maailman suosituimmista ohjelmointikielistä, se, että se on helppo oppia ja helppo käyttää, se, että se on avointa lähdekoodia ja ilmainen, se, että se on turvallinen, nopea ja tehokas ja se, että sillä on laaja yhteisön tuki (kymmeniä miljoonia kehittäjiä). (W3schools.com, 2019)

Javan käytettävyysaste on itse asiassa eri ohjelmointikielien välillä korkein, koska Javalla ohjelmoivia kehittäjiä on paljon ja Java on käytössä monissa eri järjestelmissä. Lisäksi suurin osa yrittäjistä palkkaavat mielellään Java-kehittäjiä, jotta saisivat yritykselleen räätälöityjä ohjelmistoratkaisuja. (YourTeamInIndia, 2020)

Java on johtavassa asemassa yritysten ohjelmistokehityksessä, koska esimerkiksi Google käyttää yhä Javaa mobiilisovellusten kehittämiseen Androidille. Java on lisäksi tärkeässä roolissa sähköisissä kauppapaikoissa, kuten eBay ja Amazon. Tämän lisäksi Java on tärkeässä asemassa avoimen lähdekoodin yhteisössä monilla resursseillaan ja hauskoilla työkaluillaan. (YourTeamInIndia, 2020)

Java toimii ”kirjoita kerran, aja kaikkialla” -periaatteella, joka on alustariippumaton. Se on yleiskäyttöisenä tietokoneiden ohjelmointikielenä rinnakkainen, luokkiin perustuva ja oliopohjainen. Java-alusta sisältää moottorin ajon, kääntäjän ja joukon erilaisia kirjastoja. Java itse asiassa perii syntaksinsa C- ja C++-ohjelmointikieliltä. Java-koodi on tarkoitettu käännettäväksi tavukoodiksi, joka suoritetaan tämän jälkeen Javan virtuaalikoneessa (JVM, Java Virtual Machine). Java on yhä myös TIOBE-indeksin mukaan yhä yksi suosituimmista ohjelmointikielistä ja toisiksi haetuin ohjelmointikieli PYPL:ssä (Popularity of Programming Languages, Ohjelmointikielien suosio). (YourTeamInIndia, 2020)

Javassa ohjelmia ei käännetä suoritettaviksi tiedostoiksi vaan sen sijaan ne käännetään tavukoodiksi. Tämän jälkeen Javan virtuaalikone suorittaa ne ajonaikaisesti. Myös Java-kääntäjää käytettäessä Javalla kirjoitettu lähdekoodi käännetään tavukoodiksi. (YourTeamInIndia, 2020)

Java on lisäksi turvallinen, koska se käyttää julkiseen avaimeen perustuvaa salausta. Tämä varmistaa sen, että se ei sisällä viruksia ja on kajoamaton. Java on myös vankka, koska se yrittää mahdollisuuksien mukaan poistaa virhetilanteet keskittymällä ajonaikaiseen virheentarkistukseen. Javassa on myös monisäikeinen eli useita moniajo-ohjelmia voidaan luoda samanaikaisesti. Javalla kirjoitetut ohjelmat voivat lisäksi sisältää paljon ajonaikaista dataa ja ovat dynaamisempia kuin C:llä ja C++:lla kirjoitetut ohjelmat. (YourTeamInIndia, 2020)

Javaa voidaan käyttää ohjelmistokehitysympäristöissä, kuten NetBeans, Enide Studio 2014, BlueJ ja DrJava. Se on käytössä sovelluksissa kuten Google, NASA World Wind, Uber, Spotify, LinkedIn, Pinterest ja Tripadvisor. (YourTeamInIndia, 2020)

Esimerkki Java-ohjelmasta:

```
public class Ohjelma {
public static void main (String[] args) {
System.out.println(“Hello world!”);
  }
}
```

Edeltävä esimerkki tulostaa konsoliin tekstin ”Hello world!”. Tämä esimerkkiohjelma on monen ohjelmoijan ensimmäiseksi kirjoittama ohjelma Javassa ja muissakin ohjelmointikielissä.

4.12 **Jsoup**

Jsoup on Java-kirjasto, jota käytetään HTML:n käsittelyssä. Se tarjoaa käytännöllisen ohjelmointirajapinnan tiedon purkamiseen ja käsittelyyn käyttäen DOMia, CSS:ää ja jQueryn kaltaisia metodeja. Jsoup käyttää WHATWG HTML5:n spesifikaatiota ja muuntaa HTML:n samanlaiseksi DOMiksi kuin nykyaikaiset selaimetkin. Jsoup pystyy keräämään ja muuntamaan HTML:n URL-osoitteesta, tiedostosta tai merkkijonosta, löytämään ja purkamaan tietoa käyden läpi DOMia tai CSS-valitsimia, muokkaamaan HTML-elementtejä, attribuutteja ja tekstiä, puhdistamaan käyttäjän lähettämää tietoa estääkseen XSS-hyökkäyksiä ja tulostamaan selkeää HTML:ää. Jsoup on suunniteltu käsittelemään kaikenlainen HTML; koskemattomasta HTML:stä HTML:n validointiin ja kirjoitusasultaan vääränlaiseen HTML-tietoon, jsoup muodostaa johdonmukaisen puurakenteen. (Hedley, 2018)

4.13 **Volley**

Volley on HTTP-kirjasto, joka tekee datan lähettämisen ja vastaanottamisen verkossa Android-sovelluksissa hyvin helpoksi ja nopeaksi. Sen kehitti Google vuonna 2013 Google I/O-tapahtumassa. Se kehitettiin, koska Android SDK:ssa ei ole verkossa liikkuvan datan

lähettämistä ja vastaanottamista käsittelevää luokkaa, joka toimisi häiritsemättä käyttäjäkokemusta. Vaikka Volley on osa Android Open Source Projectia (AOSP), Google ilmoitti tammikuussa 2017, että Volley siirretään itsenäiseksi kirjastokseen. Volley hallitsee verkkopyyntöjen käsittelyn ja välimuistiin tallentamisen ja auttaa kehittäjiä säästämään aikaa siten, että kehittäjien ei tarvitse kirjoittaa samaa verkkopyyntö- ja välimuistiintallennuskoodia uudestaan ja uudestaan. (GeeksforGeeks, 2020)

4.14 **XPath**

XPath (XML Path Language) on tärkeä elementti XSLT -standardissa. XPathia voidaan käyttää XML-dokumentin elementtien ja attribuuttien läpi navigoimiseen. XPath käyttää ”polkumaista” syntaksia identifioimaan ja navigoimaan XML-dokumentin eri osissa. XPath sisältää yli 200 sisäänrakennettua funktiota ja on W3C:n suositus. (W3Schools, 2020)

XPath sisältää funktioita merkkijonojen, numeerisen tiedon, boolean-tyyppisen tiedon, päivämäärä- ja aikatiedon ja monen muun tyypin tiedon käsittelyyn. Nykypäivänä XPathin funktioita voidaan käyttää yhdessä JavaScriptin, Javan, XML Scheman, PHP:n, C:n, C++:n ja monen muun ohjelmointikielen kanssa. (W3Schools, 2020)

5 KÄYTÄNNÖN OSUUS JA OHJELMISTON KEHITYS JA TESTAUS

Aluksi suoritettiin pieniä kokeita Android Studiolla. Tehtiin koe, jossa JSON-muotoista tietoa haettiin Internetistä. Tehtiin myös testiohjelma NetBeansilla ja Eclipsellä. Varsinaista ohjelmaa muistuttava testiohjelma saatiin toimimaan sekä Netbeansissa että Eclipsessä. Ainoa ero varsinaisen ohjelman ja testiohjelman välillä oli se, että varsinainen ohjelma käyttää Volley-kirjastoa, kun testiohjelma käyttää kirjoitettua metodia. Testaamisessa käytettiin lisäksi Visual Studiota.

Ohjelmointikielinä käytettiin Javaa ja C#:a. Molempia ohjelmointikieliä käytettiin NHL:n tulosten hakemiseksi ohjelmaan.

Jsoupia testattiin kirjoittamalla Internetistä löytyvä esimerkkiohjelma. Tähän käytettiin NetBeansia.

Android Studioon piti lisätä Jsoup-kirjastot, jotta se tunnistaisi Jsoupiin liittyvät komennot. Jsoupissa piti ensin määrittää haettava dokumentti komennolla *Document doc = Jsoup.connect(”https://www.nhl.com/fi/scores”).get();*. Sen jälkeen piti erotella dokumentista ensimmäinen pääotsikko komennolla *Element title = doc.select(”div.article-item__top > h1”).first();*. Tämän jälkeen haettiin dokumentista ensimmäisen toisen tason otsikon komennolla *Element subtitle = doc.select(”div.article-item__top > h2”).first();*. Ensimmäinen komento muodostaa Document-tyyppisen olion ja yhteyden osoitteeseen https://www.nhl.com/fi/scores. Toinen komento etsii doc-oliosta kaikki div-elementit, joiden class-attribuutti on ”article-item__top” ja hakee niistä ensimmäisen. Samalla tavoin seuraava komento etsii doc-oliosta kaikki div-elementit, joiden class-attribuutti on ”article-item___top” ja hakee niistä ensimmäisen. Ensimmäisen tason otsikot nimetään edellä title-nimisiksi olioiksi ja toisen tason otsikot nimetään subtitle-nimisiksi olioiksi.

Opinnäytetyössä tehtiin päätös, että vain NHL:n tuloksia käsitellään, koska tulosten saaminen NBA:n ja Valioliigan sivuilta osoittautui haasteelliseksi.

Suomalaisista NHL-pelaajista tehtiin ArrayList-tyyppinen lista, joka hakee halutun suomalaisen NHL-pelaajan nimen, jos tämä tekee maalin tai antaa maaliin johtaneen syötön.

5.1 Jsoup, XPath ja C# ohjelman teossa

Aluksi kokeiltiin tallentaa dynaamisesti muodostettua NHL:n HTML-sivua paikallisesti tietokoneelle, josta Jsoup lukisi otteluiden maalit Android-sovellukseen. Tämä ei kuitenkaan oikein onnistunut, joten siirryttiin eteenpäin seuraavaan vaiheeseen.

Koodiesimerkin 7 koodia käytettiin yritettäessä hakea joukkueita ja otteluiden tuloksia.

Koodiesimerkin 7 ajamisen jälkeen Visual Studiossa aiheutui virhe ”System.InvalidOperationException: ’session not created: Chrome version must be between 70 and 73’”. Tämä johtui luultavimmin siitä, että tietokoneessa oli asennettuna Chromen versio 81, ja koodi tarvitsi Chromen versiota väliltä 70 ja 73. Työskentelyä jatkettiin kokeillen muita mahdollisia vaihtoehtoja.

```
final StringBuilder builder = new StringBuilder();

String fileName = "C:\\Users\\Mikko\\Desktop\\scores.htm";

try {
      Document doc = Jsoup.parse(new File(fileName), "utf-8");
      Elements spanTags = doc.getElementsByClass
      ("g5-component--nhl-     scores__team-score");
      builder.append(spanTags.text());
  } catch (IOException e) {
         e.printStackTrace();
      }
```

Koodiesimerkki 8, koodi tulosten hakemiseen paikallisesta tiedostosta.

Edeltävää koodia kokeiltiin tulosten hakemiseksi paikallisesti tallennetusta tiedostosta. Siinä ensin muodostetaan uusi Stringbuilder-olio nimeltä builder, minkä jälkeen String-tyyppinen

muuttuja nimeltä fileName, joka osoittaa paikallisen tiedoston sijainnin, tallennetaan fileName-muuttujaan. Tämän jälkeen muodostetaan Document-tyyppinen olio doc, johon käytetään metodia ”Jsoup.parse(new File(fileName), ”utf-8”);”, joka muuntaa String-tyyppisen muuttujan fileName Document-tyyppiseksi olioksi doc, jonka merkistö on utf-8. Seuraavaksi muodostetaan Elements-tyyppinen muuttujajoukko nimeltä spanTags kaikista olion doc elementeistä, joiden class-attribuutti on ”g5-component—nhl-scores__team-score”. Tämän jälkeen builder-olioon lisätään spanTags-elementtien tekstit komennolla ”builder.append(spanTags.text());”. Jos virheitä ilmaantuu, ne otetaan kiinni catch-osiossa ja tulostetaan konsoliin virheen tuottama virheilmoitus. Yllä olevaa koodia ei saatu toimimaan, joten jatkettiin kokeilemalla muita vaihtoehtoja.

5.2 **Ohjelman tekemisen aloittaminen ja toimivan työskentelymetodin valinta**

Sovellus saatiin lukemaan ääneen teksti ”edit text”, joka on yksi Android Studion graafinen komponentti, kun painiketta ”lue ääneen” painetaan. Myös Internetistä ladattu JSON-muodossa olevan tietokokoelma, joka oli Android Studiossa muuttuja nimeltä resultNHL, saatiin luetuksi ääneen käyttäen Android Studion ”Text-To-Speech”-ominaisuutta.

```
btnReadAloud.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
    speak();
   }
});

private void speak() {
    String text = resultNHL.getText().toString();
    mTTS.speak(text, TextToSpeech.QUEUE_FLUSH, null, null);
}
```

Koodiesimerkki 9, koodi, jolla sovellus lukee ääneen näytöllä olevan tekstin.

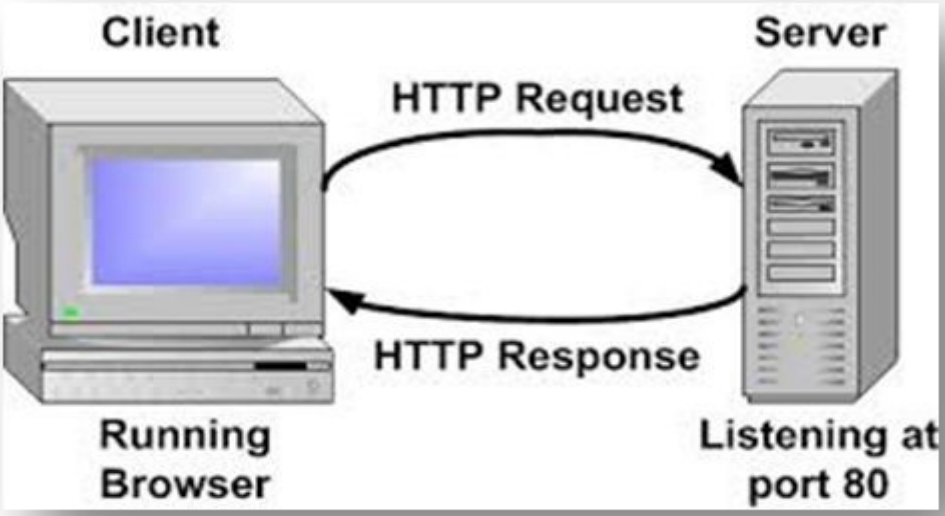
Edeltävä koodi asettaa uuden napsautuksia kuuntelevan toiminnon painikkeelle nimeltä btnReadAloud ja tätä painiketta painettaessa suorittaa speak-metodissa määritellyn toiminnallisuuden, jossa String-tyyppiselle muuttujalle text asetetaan arvoksi internetistä ladattu JSON-muodossa oleva teksti ja muunnetaan se String-tyyppiseksi metodilla toString(). Sovellus saadaan lukemaan teksti ääneen käyttäen mTTS-muuttujaan metodia speak(text, TextToSpeech.QUEUE_FLUSH, null, null).

Android Studioon lisättiin lisäksi Volley-kirjasto, jolla JSON-muodossa oleva tieto muunnetaan Android Studion TextView-komponentissa näytettäväksi tekstiksi.

Erinäisten kokeilujen jälkeen sovellus saatiin toimimaan halutulla tavalla, kun tiedon hakemismuotona käytettiin JSON:ää ja Internet-osoitteena osoitetta https://nhl-score-api.herokuapp.com/api/scores/latest. Internet-osoite on JSON API(Application Programming Interface, ohjelmointirajapinta), joka sisältää tiedot tuoreimman pelatun NHL-kierroksen tuloksista ja maaleista.

Tiedon hakemiseen käytetään HTTP-protokollan GET-metodia, jota käytetään resurssin hakemiseen.

Sovelluksessani asiakkaana on Android-käyttöjärjestelmän sisältävä älypuhelin ja palvelimena NHL-otteluiden tiedot JSON-muodossa sisältävä palvelin.



Kuva 6: HTTP-pyyntö palvelimelle ja HTTP-vastaus palvelimelta

Tietoihin sisältyy aluksi JSON-objekti nimeltään date, joka sisältää kaksi String-tietotyyppistä avain-arvo-paria. Ensimmäisen avain-arvo-parin avain on raw ja arvo on pelatun pelin päivämäärä muodossa ”vuosi-kuukausi-päivä”, esimerkiksi ”2020-03-11”. Toisen avain-arvo-parin avain on pretty ja arvo on pelatun pelin päivämäärä muodossa ”viikonpäivä kuukausi päivä”, esimerkiksi ”Wed Mar 11”.

Toinen osa tiedosta on JSONArray-tyyppisessä taulukossa oleva tietue nimeltä games. Tämä oli oleellinen osa opinnäytetyötä, koska juuri se sisälsi tiedot tuoreimmista pelatuista peleistä, kuten esimerkiksi lopputulokset ja peleissä tehdyt maalit. Games-tietue oli jaettu NHL-kierroksella pelattujen viiden pelin perusteella viiteen JSON-objektiin. Nämä oli nimetty 0, 1, 2, 3 ja 4 perustuen siihen, että taulukoissa indeksien laskeminen aloitetaan nollasta, josta jatketaan eteenpäin. Näiden JSON-objektien ensimmäinen tieto oli status eli pelin senhetkinen tilanne. Tämä sisälsi String-tietotyyppisen tiedon nimeltä state. Sen arvoksi tulee ”FINAL”, kun kyseinen peli on pelattu loppuun. Toisena tietona oli itsenäinen String-tyyppinen tieto startTime. Se sisälsi pelin aloitusajankohdan päivämäärän ja kellonajan, esimerkiksi ”2020-03-12T00:00:00Z”.
Kolmantena tietona oli JSONArray-tyyppisessä taulukossa oleva tietue nimeltä goals. Se sisältää pelissä tehtyihin maaleihin liittyvät tiedot JSON-objekteina, joiden nimeäminen alkaa nollasta. Näiden JSON-objektien ensimmäinen tieto oli String-muodossa oleva maalin tehneen

joukkueen kolmikirjaiminen lyhenne nimeltä team. Toinen tieto oli String-tyyppinen tieto period eli erä, jossa maali tehtiin nimeltä period. Kolmas tieto oli JSON-objekti nimeltä scorer, joka sisälsi tietoja maalin tehneestä pelaajasta. Näihin tietoihin kuuluivat pelaajan nimi String-tyyppisenä tietona nimeltä player ja kyseisen pelaajan kuluvan kauden siihen mennessä saavuttamat yhteispisteet int-muodossa. Neljäs tieto oli JSONArray-tyyppinen taulukko nimeltä assists, joka sisälsi tietoja maalisyötön antaneista pelaajista. Se sisälsi tietoa JSON-objekteina. JSON-objektin ensimmäinen tieto oli maalisyötön antaneen pelaajan nimi eli player ja toinen tieto oli tämän pelaajan kauden siihen mennessä saavuttamat yhteispisteet. Lisäksi JSON-objektiin sisältyi tieto maalin syntyajasta minuuteissa (min) ja sekunneissa (sec) ja tieto siitä, jos maali oli tehty esimerkiksi ylivoimalla.
Neljäs tieto oli JSON-objekti nimeltä scores, joka sisälsi String-tyyppisenä tietona ottelun lopulliset maalimäärät.
Tiedon käsittely aloitettiin tekemällä metodi nimeltä jsonParse, joka ei palauta mitään. Metodin sisältämä ensimmäinen tieto oli String-tyyppinen muuttuja nimeltä url, joka sai arvokseen Internet-osoitteen, joka sisälsi ohjelmassa käytetyn JSON-tiedon eli https://nhl-score-api.herokuapp.com/api/scores/latest. Jotta JSON-muotoista Internetistä haettua tietoa pystyttiin käsittelemään, piti ohjelmaan sisällyttää JSON-tiedon käsittelyyn tarvittava Volley-niminen kirjasto koodirivillä import com.android.volley.toolbox.JsonObjectRequest. Tämän jälkeen oli mahdollista käyttää JsonObjectRequest-luokkaa ohjelmassa.

5.3 **Ohjelman tekeminen Android Studiolla**

Ohjelman koodi oli jo kertaalleen kirjoitettu, mutta ohjelmaa varten tehtiin uusi projekti Android Studiolla, koska kirjoitettu koodi sisälsi myös käyttämätöntä ja toimimatonta koodia.

Projektiin tehtiin ScrollView-komponentti nimeltä m_Scroll, TextView-komponentti nimeltä resultNHL, johon NHL-tulokset tulostetaan, painike nimeltä btnGetNHLData, joka hakee tuoreimmat NHL-tulokset ja tulostaa ne matkapuhelimen näytölle, painike nimeltä btnReadAloud, joka lukee ääneen matkapuhelimen näytöllä olevan tekstin, painike nimeltä btnEmptyData, joka tyhjentää näytöllä olevan tekstin, RequestQueue-komponentti nimeltä mQueue ja TextToSpeech-komponentti nimeltä mTTS, joka käsittelee ääneen luettavaan tekstiin liittyvää dataa. Projektiin tehtiin lisäksi kaksi List-tyyppistä listaa nimeltä listOfPlayersWhoScored ja listOfPlayersWhoAssisted, joihin tallennetaan pelaajat, jotka

tekevät maalin tai antavat maaliin johtaneen syötön ja ArrayList-tyypinen lista nimeltä players, johon lisättiin kaikki suomalaiset NHL-pelaajat maalivahteja lukuun ottamatta.

Ohjelman toimintalogiikka toteutettiin siten, että ensin players-listaan lisätään kaikki suomalaiset NHL-pelaajat koodirivillä, joka on muotoa players.add(”Pelaajan Nimi”). Sen jälkeen käytetään Volley-kirjaston ominaisuutta käsitellä Json-tyyppistä tietoa ja lähetetään JsonObjectRequest-tyyppinen pyyntö osoitteeseen https://nhl-score-api.herokuapp.com/api/scores/latest. Tämä palauttaa Json-objektina tiedot pelatuista peleistä, jos pyynnön käsittelyssä ei ilmaannu virheitä. Jos virheitä ilmaantuu, ne käsitellään ohjelmaan tehdyssä try-catch-blokin catch-osiossa, joka käsittelee JsonException-tyyppisiä poikkeuksia. Ohjelmassani virheen tapahtuessa vain tulostetaan virheen tiedot konsoliin.

Kun ohjelma on kahden for-loopin avulla käynyt läpi Json-objektin tiedot ja löytänyt tiedon maalin tehneestä pelaajasta, lisätään se listOfPlayersWhoScored-listaan. Tämän jälkeen verrataan listOfPlayersWhoScored-listassa olevien pelaajien nimiä players-listassa oleviin suomalaispelaajiin ja katsotaan, löytyykö jonkun suomalaisen pelaajan nimi kummastakin listasta. Mikäli näin on, tulostetaan matkapuhelimen näytölle pelaajan nimi ja teksti ”teki maalin yön NHL-kierroksella”. Jos taas samalla logiikalla toimivasta listOfPlayersWhoAssisted-listasta löytyy suomalainen pelaaja, tulostetaan matkapuhelimen näytölle pelaajan nimi ja teksti ”oli avustamassa joukkueen tekemässä maalissa”.

```
players.add("Markus Granlund");
players.add("Antti Suomela");
players.add("Aleksi Saarela");
```

Koodiesimerkki 10, suomalaisten pelaajien lisääminen players-listaan.

Liitteiden kuvassa 6 nähdään, että ohjelmaan tehtiin toiminnan kannalta oleellisista painikkeista riittävän isot saavutettavuus mielessä pitäen. Myös tulosten teksti on riittävän suurella fontilla, jotta sen erottuisi lukemaan hyvin.

Kuvassa 7 (NHL-tulokset -näkymä) taas nähdään, että kun NHL-painiketta painetaan, hakee ohjelma internetistä tuoreimman pelatun NHL-kierroksen suomalaiset maalintekijät ja maalinsyöttäjät. (pvm 12.3.2020) Kun painetaan Lue ääneen painiketta, ohjelma lukee ääneen näytöllä olevan tekstin. Vastaavasti kun painetaan Tyhjennä painiketta, tyhjennetään näytön sisältämä teksti.

Ohjelmaa kehitettiin vielä eteenpäin siten, että se tulostaa näytölle kierroksen pelatun viiden pelin lopulliset tulokset.

Ohjelma saatiin asentumaan matkapuhelimelle ja ohjelma puhumaan suomalaisella korostuksella, kun käytettiin matkapuhelimen helppokäyttöisyystoimintoja.

Kuvissa 8 ja 9 nähdään viiden tuoreimman pelatun pelin lopulliset tulokset sekä suomalaiset maalintekijät ja maalinsyöttäjät.

5.4 **Ohjelman testaaminen**

Ohjelma saatiin toimimaan hyvin tietokoneella käytettävällä emulaattorilla, joten ohjelmaa siirryttiin testaamaan oikealla matkapuhelimella. Ohjelma toimi myös matkapuhelimella hyvin, tosin ilman helppokäyttöisyystoimintoa englanninkielisellä korostuksella ja välillä niin, että toimintojen suorittamiseksi painiketta piti painaa useaan kertaan.

6 PÄÄTELMÄT JA KEHITTÄMISIDEOITA

Opin opinnäytetyötä tehdessäni uusia asioita myös aiheista, joita en ottanut mukaan lopulliseen opinnäytetyöhön, kuten Python ja Beautiful Soup. Opinnäytetyön tekeminen osoittautui kuitenkin aluksi melko haasteelliseksi, koska en saanut Jsoupin osalta kirjoittamaani koodia toimimaan haluamallani tavalla.

Yritin pitkään tehdä opinnäytetyötä siten, että haen tarvittavat tiedot suoraan NHL:n virallisilta Internet-sivuilta, mutta tämän osoittauduttua liian hankalaksi päädyin käyttämään yhtä löytämääni ohjelmointirajapintaa ja JSONia sekä Volleyta. Sain koodin onneksi kuitenkin toimimaan, kun käytin JSONia ja Volley-kirjastoa. Jatkokehitysideana voisi olla, että ohjelma hakee NHL:n lisäksi tulokset myös NBA:sta ja Valioliigasta. Toinen jatkokehitysidea voisi olla, että ohjelma lukisi sujuvalla suomen kielellä ääneen tulokset, kun nyt se lukee ne englanninkielisellä korostuksella. Näiden lisäksi jatkokehitysideana voisi olla, että suomalaiset pelaajat olisi tallennettu valmiiksi johonkin tiedostoon, jotta niitä ei tarvitsisi lisätä yksitellen ohjelmaan jokaisen ajon yhteydessä. Koodin olisi voinut myös jakaa eri toiminnallisuuksia käsitteleviin luokkiin, mikä on järkevä tapa toimia, kun ohjelmistoja kehitetään, varsinkin jos niistä tulee kooltaan suuria. Toteutin ohjelman kuitenkin niin, että toiminnallisuudet ovat yhdessä tiedostossa, mikä toisaalta mielestäni omalla tavallaan teki ohjelman kehittämisestä helpompaa. Viimeisenä kehitysideana voisi olla, että en kuluttaisi liian paljon aikaa yrittäessäni ratkaista jotain ongelmaa vaan siirtyisin eteenpäin kokeilemaan muita vaihtoehtoja, minkä opin järkeväksi tavaksi toimia työstäessäni opinnäytetyötä.

Opinnäytetyön tekemiseen ja testaukseen vaikutti myös koronapandemia. Se vaikutti merkittävimmin tehdyn ohjelman testaukseen, koska NHL:ssä oli pandemian takia tauko peleissä välillä maaliskuu 2020 – elokuu 2020, joten uusista peleistä ei tullut tietoja käyttämääni ohjelmointirajapintaan. Lisäksi parempi suunnittelu ja tämän suunnittelun mukaan toimiminen olisivat tehostaneet ja nopeuttaneet opinnäytetyön tekoprosessia.

LÄHTEET

Adaptive Path. (n.d.). AJAX: A New Approach to Web Applications. Haettu 1.2.2019 osoitteesta https://adaptivepath.org/ideas/ajax-new-approach-web-applications/

Aluehallintovirasto. (21.9.2020). Viranomaisten verkkosivustojen on oltava lain mukaan saavutettavat 23.9.2020 – Saavutettavuusvaatimukset. Haettu 26.9.2020 osoitteesta https://www.saavutettavuusvaatimukset.fi/viranomaisten-verkkosivustojen-on-oltava-lain-mukaan-saavutettavat-23-9-2020/

BigCommerce. (n.d.). What is click fraud? How to Identify and prevent Fraudulent Clicks. Haettu 26.9.2020 osoitteesta https://www.bigcommerce.com/ecommerce-answers/what-is-click-fraud-identify-and-prevent-fraudulent-clicks/

Blair, Ian. (n.d.). Mobile App Accessibility is a Must in App Development. Haettu 1.11.2020 osoitteesta https://buildfire.com/app-accessibility-mobile-development/

Callaham, John. (n.d). The history of Android OS: its name, origin and more. Haettu 3.2.2019 osoitteesta https://www.androidauthority.com/history-android-os-name-789433/

CloudFlare. (n.d.). What Is a Bot? | Bot Definition. Haettu 26.9.2020 osoitteesta https://www.cloudflare.com/learning/bots/what-is-a-bot/

CloudFlare. (n.d.). What Is Content Scraping? | Web Scraping. Haettu 26.9.2020 osoitteesta https://www.cloudflare.com/learning/bots/what-is-content-scraping/

CloudFlare. (n.d.). | What Is Credential Stuffing – Credential Stuffing vs Brute Force Attacks. Haettu 26.9.2020 osoitteesta https://www.cloudflare.com/learning/bots/what-is-credential-stuf fing/?utm_referrer=https://www.google.com/

Codeacademy. (n.d.). | Python Tutorial: Learn Python For Free. Haettu 30.1.2019 osoitteesta https://www.codecademy.com/learn/learn-python

GeeksforGeeks (n.d.). - Volley Library in Android. Haettu 13.5.2020 osoitteesta https://www.geeksforgeeks.org/volley-library-in-android/

Git Handbook. (n.d.). – GitHub Guides. Haettu 3.2.2019 osoitteesta https://guides.github.com/introduction/git-handbook/

Gradle. (n.d.). Gradle User Manual. Haettu 3.2.2019 osoitteesta https://docs.gradle.org/current/userguide/userguide.html

Ilmainen Sanakirja. (n.d.). Ilmainen sanakirja. Haettu 27.1.2019 osoitteesta https://ilmainen-sanakirja.fi/

JSON.org. (n.d.). JSON. Haettu 25.1.2019 osoitteesta https://www.json.org/

Jonathan Hedley. (n.d.). jsoup Java HTML Parser, with best of DOM, CSS, and jquery. Haettu 10.2.2019 osoitteesta https://jsoup.org

Jyväskylän yliopisto. (n.d.). PHP:n perusteet. Haettu 26.9.2020 osoitteesta http://users.jyu.fi/~kolli/ITK215_05/php/

Martin, Jeff. (n.d.). The Importance of Aesthetics and Usability in Web Design – Epic Marketing. Haettu 3.11.2020 osoitteesta https://marketingepic.com/the-importance-of-aesthetics-and-usability-in-web-design/

MDN. (n.d.). HTTP. Haettu 1.11.2020 osoitteesta https://developer.mozilla.org/en-US/docs/Web/HTTP

Python For Beginners. (28.8.2020). Beautiful Soup 4 Python. Haettu 31.1.2019 osoitteesta https://www.pythonforbeginners.com/beautifulsoup/beautifulsoup-4-python

Rasa. (n.d.). Rasa: Open source conversational AI. Haettu 27.1.2019 osoitteesta https://rasa.com/

Rasa. (n.d.). Rasa | Generating NLU Data. Haettu 4.11.2020 osoitteesta https://rasa.com/docs/rasa/generating-nlu-data

REST API Tutorial. (n.d.). What Is JSON. Haettu 25.10.2020 osoitteesta (https://restfulapi.net/introduction-to-json/

Rouse, Margaret. (n.d.). What Is Android Studio? – Definition from WhatIs.com. Haettu 3.2.2019 osoitteesta https://searchmobilecomputing.techtarget.com/definition/Android-Studio

Rouse, Margaret. (n.d.). What is RESTful API? – Definition from WhatIs.com. Haettu 23.1.2019 osoitteesta https://searchmicroservices.techtarget.com/definition/RESTful-API

Saavutettavuusdirektiivi. (n.d.). Mitä saavutettavuus on?. Haettu 23.1.2019 osoitteesta https://saavutettavuusdirektiivi.fi/mita-on-saavutettavuus/

Shawn Lawton, Henry. (n.d.). Introduction to Web Accessibility | Web Accessibility Initiative (WAI) | W3C. Haettu 3.11.2020 osoitteesta https://www.w3.org/WAI/fundamentals/accessibility-intro/

Sulava. (n.d.). Botit yritysmaailmassa – tätä päivää vai tulevaisuutta?. Haettu 23.1.2019 osoitteesta https://www.sulava.com/botit-yritysmaailmassa/

Tutorialspoint. (n.d.). JSON Comparison with XML. Haettu 27.1.2019 osoitteesta https://www.tutorialspoint.com/json/json_comparison.htm

W3Schools. (n.d.). CSS Introduction. Haettu 27.1.2019 osoitteesta https://www.w3schools.com/css/css_intro.asp

W3Schools. (n.d.). Introduction to HTML. Haettu 27.1.2019 osoitteesta https://www.w3schools.com/html/html_intro.asp

W3Schools. (n.d.). PHP 5 Tutorial. Haettu 27.1.2019 osoitteesta https://www.w3schools.com/php/default.asp

W3Schools. (n.d.). XPath Tutorial. Haettu 1.11.2020 osoitteesta https://www.w3schools.com/xml/xpath_intro.asp

Yegulalp, Sergar. (n.d.). Google open-sources language-agnostic, scalable software tool | InfoWorld. Haettu 27.1.2019 osoitteesta https://www.infoworld.com/article/2983495/development-tools/google-open-sources-language-agnostic-scalable-software-tool.html

YourTeamInIndia. (n.d.). Java vs Kotlin – The Best Language for Android App Development?. Haettu 26.9.2020 osoitteesta https://www.yourteaminindia.com/blog/java-vs-kotlin/

Zeil, Steven J. (n.d.). Integrated Development Environments. Haettu 3.2.2019 osoitteesta https://www.cs.odu.edu/~zeil/cs350/f17/Public/IDEs/index.html

Vedran Bozicevic. (10.5.2019). What Bad Bots Do (and How You Can Stop Them). Haettu 16.12.2020 osoitteesta https://www.globaldots.com/blog/what-bad-bots-do

```
package com.example.nhl;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Build;
import android.os.Bundle;

import android.speech.tts.TextToSpeech;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ScrollView;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {

    private Button btnGetNHLData;
    public TextView resultNHL;
    private Button btnReadAloud;
    private RequestQueue mQueue;
    private Button btnEmptyData;
```

```
    ScrollView m_Scroll;

    private TextToSpeech mTTS;

    private ArrayList<String> players = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        resultNHL = findViewById(R.id.resultNHL);
        btnGetNHLData = findViewById(R.id.btnGetNHLData);
        btnReadAloud = findViewById(R.id.btnReadAloud);
        btnEmptyData = findViewById(R.id.btnEmptyData);
        m_Scroll = new ScrollView(this);
        mQueue = Volley.newRequestQueue(this);
        btnGetNHLData.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
              jsonParse();
            }
        });
        mTTS = new TextToSpeech(this, new TextToSpeech.OnInitListener() {
           @Override
           public void onInit(int status) {
               if (status == TextToSpeech.SUCCESS) {
                   int result = mTTS.setLanguage(Locale.US);

                   if (result == TextToSpeech.LANG_MISSING_DATA || result == Text-
ToSpeech.LANG_NOT_SUPPORTED) {
                       Log.e("TTS", "Language not supported");
                   }
               } else {
                       Log.e("TTS", "Initialization failed");
                   }
           }
        });

        btnReadAloud.setOnClickListener(new View.OnClickListener() {
           @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
```

```
    @Override
    public void onClick(View v) {
         speak();
    }
});

btnEmptyData.setOnClickListener(new View.OnClickListener() {
    @Override
     public void onClick(View v) {
        resultNHL.setText("");
    }
});

players.add("Markus Granlund");
players.add("Antti Suomela");
players.add("Aleksi Saarela");
players.add("Roope Hintz");
players.add("Mikko Koivu");
players.add("Mikael Granlund");
players.add("Miika Salomaki");
players.add("Erik Haula");
players.add("Markus Hannikainen");
players.add("Ville Meskanen");
players.add("Oula Palve");
players.add("Sebastian Repo");
players.add("Jani Hakanpaa");
players.add("Joni Tuulola");
players.add("Niko Mikkola");
players.add("Sami Niku");
players.add("Alex Lintuniemi");
players.add("Sami Vatanen");
players.add("Juuso Riikola");
players.add("Oliver Kaski");
players.add("Vili Saarijarvi");
players.add("Teemu Kivihalme");
players.add("Joel Kiviranta");
players.add("Henrik Borgstrom");
players.add("Eeli Tolvanen");
players.add("Patrik Laine");
players.add("Janne Kuokkanen");
```

```
players.add("Otto Koivula");
players.add("Kasper Bjorkvist");
players.add("Joona Koppanen");
players.add("Valtteri Filppula");
players.add("Joel Armia");
players.add("Artturi Lehkonen");
players.add("Jesperi Kotkaniemi");
players.add("Otto Somppi");
players.add("Kalle Kossila");
players.add("Petrus Palmu");
players.add("Juuso Valimaki");
players.add("Olli Juolevi");
players.add("Miro Heiskanen");
players.add("Tarmo Reunanen");
players.add("Otto Leskinen");
players.add("Henri Jokiharju");
players.add("Rasmus Kupari");
players.add("Mikael Hakkarainen");
players.add("Kristian Vesalainen");
players.add("Joona Luoto");
players.add("Eetu Luostarinen");
players.add("Leo Komarov");
players.add("Kaapo Kakko");
players.add("Alexander Barkov");
players.add("Aleksi Heponiemi");
players.add("Arttu Ruotsalainen");
players.add("Kasperi Kapanen");
players.add("Olli Maatta");
players.add("Ville Heinola");
players.add("Markus Nutivaara");
players.add("Niclas Almari");
players.add("Urho Vaakanainen");
players.add("Rasmus Ristolainen");
players.add("Lassi Thomson");
players.add("Joonas Donskoi");
players.add("Sebastian Aho");
players.add("Teuvo Teravainen");
players.add("Mikko Rantanen");
players.add("Esa Lindell");
players.add("Julius Honka");
```

```
        players.add("Jesse Puljujarvi");
        players.add("Saku Maenalanen");
    }

    private void jsonParse() {
        if (resultNHL.getText().length() == 0) {
            resultNHL.append("NHL-tulokset");
        }
        String url = "https://nhl-score-api.herokuapp.com/api/scores/latest";
        JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url,
null,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        try {
                            JSONArray jsonArrayOfGames = response.getJSO-
NArray("games");
                            List<String> listOfPlayersWhoScored = new Ar-
rayList<>();
                            List<String> listOfPlayersWhoAssisted = new Ar-
rayList<>();
                            for (int a = 0; a < jsonArrayOfGames.length(); a++) {
                                JSONObject gameScoredIn = jsonArrayOfGames.get-
JSONObject(a);
                                JSONObject scoresInGame = gameScoredIn.getJSONOb-
ject("scores");
                                JSONObject teams = gameScoredIn.getJSONOb-
ject("teams");
                                JSONObject awayTeam = teams.getJSONObject("away");
                                JSONObject homeTeam = teams.getJSONObject("home");
                                String awayTeamAbbr = awayTeam.getString("abbrevi-
ation");
                                String homeTeamAbbr = homeTeam.getString("abbrevi-
ation");
                                int awayScore = scoresInGame.getInt(awayTeamAbbr);
                                int homeScore = scoresInGame.getInt(homeTeamAbbr);
```

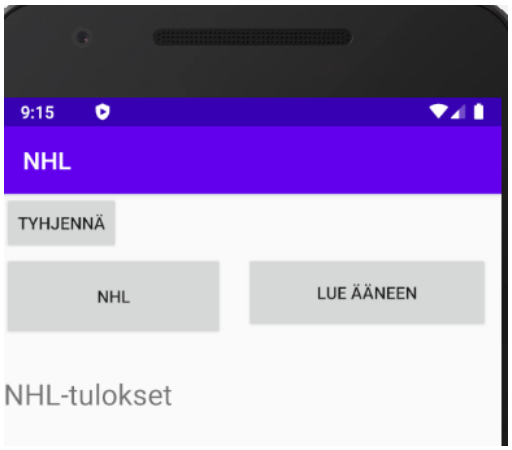


```
                                int numberOfGame = a + 1;
                                resultNHL.append("\n" + "Game " + numberOfGame +
"\n" + "Final score: " + "\n" + "Away(" + awayTeamAbbr + "): " + awayScore + "\n"
```

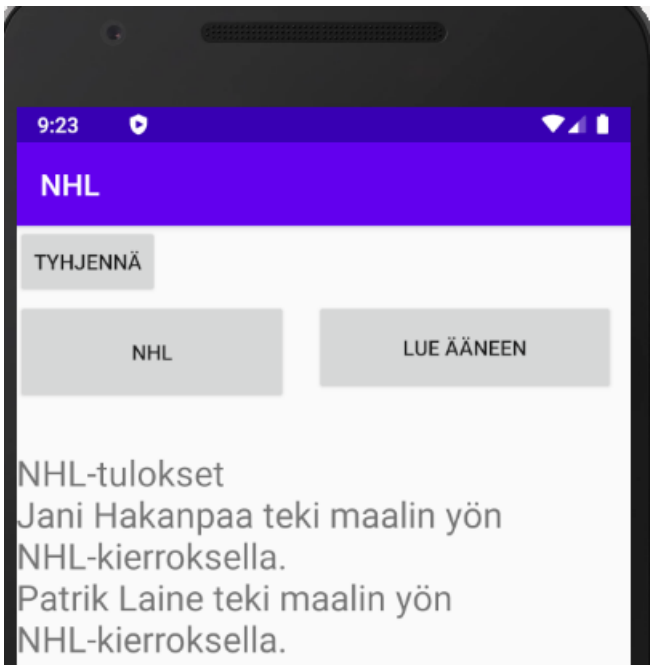
```
+ "Home(" + homeTeamAbbr +                                    6(8)
                                        "): " + homeScore + "\n");
                                JSONArray goalsScored = gameScoredIn.getJSO-
NArray("goals");
                                for (int b = 0; b < goalsScored.length(); b++) {
                                    JSONObject goalScoredByFinnishPlayer =
goalsScored.getJSONObject(b);
                                    JSONObject scorer = goalScoredByFinnish-
Player.getJSONObject("scorer");
                                    String playerWhoScored =
scorer.getString("player");
                                    listOfPlayersWhoScored.add(playerWhoScored);
                                }
                            }
                            resultNHL.append("\n");
                            for (int c = 0; c < players.size(); c++) {
                                if (listOfPlayersWhoScored.contains(play-
ers.get(c))) {
                                    resultNHL.append(players.get(c) + " teki
maalin yön NHL-kierroksella." + "\n");
                                }
                            }
                            for (int i = 0; i < jsonArrayOfGames.length(); i++) {
                                JSONObject gameAssistedIn = jsonArrayOfGames.get-
JSONObject(i);
                                JSONArray goalsAssisted = gameAssistedIn.getJSO-
NArray("goals");
                                for (int j = 0; j < goalsAssisted.length(); j++) {
                                    JSONObject goalAssistedByFinnishPlayer =
goalsAssisted.getJSONObject(j);
                                    JSONArray assists = goalAssistedByFinnish-
Player.getJSONArray("assists");
                                  for (int k = 0; k < assists.length();  7(8) k++)
{
                                        JSONObject assist = assists.getJSONOb-
ject(k);
                                        String playerWhoAssisted = as-
sist.getString("player");
                                        listOfPlayersWhoAssisted.add(playerWhoAs-
sisted);
```

```
                                }
                            }
                        }
                        for (int l = 0; l < players.size(); l++) {
                            if (listOfPlayersWhoAssisted.contains(play-
ers.get(l))) {
                                resultNHL.append(players.get(l) + " oli
avustamassa joukkueen tekemässä maalissa.");
                            }
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            error.printStackTrace();
        }
    });
    mQueue.add(request);
}

@RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
private void speak() {
    String text = resultNHL.getText().toString();
    mTTS.speak(text, TextToSpeech.QUEUE_FLUSH, null, null);
}

@Override
protected void onDestroy() {
    if (mTTS != null) {
        mTTS.stop();
        mTTS.shutdown();
    }

    super.onDestroy();
}
```

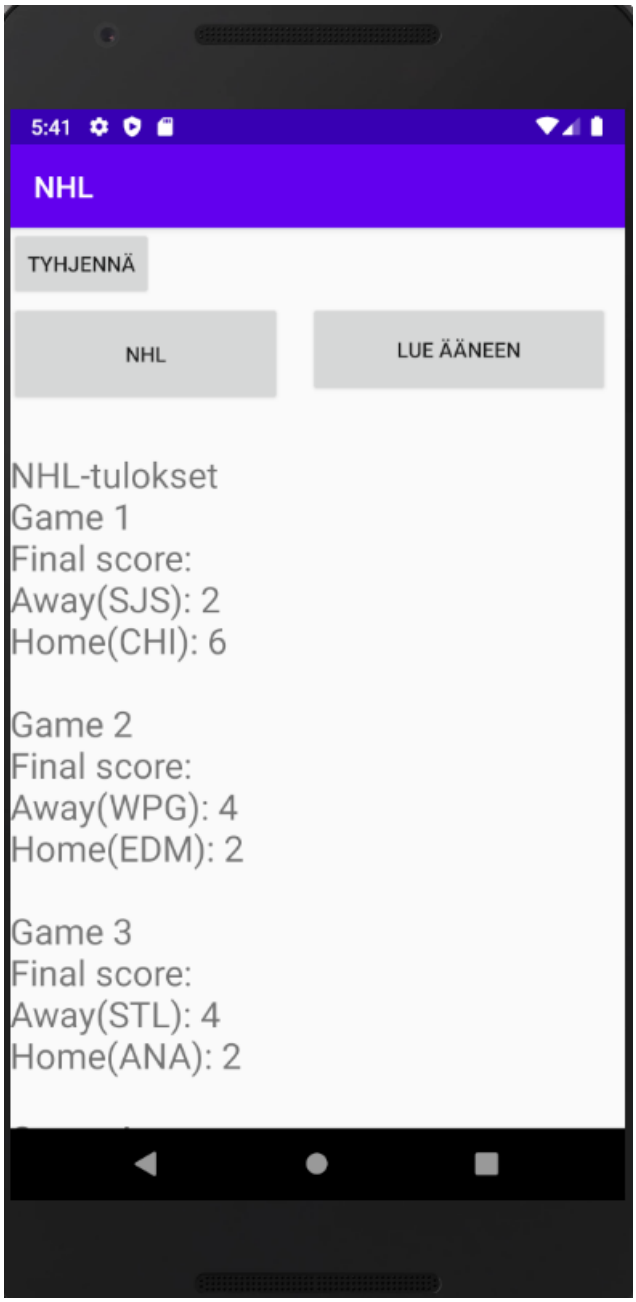
```
    @Override
    protected void onPause() {
        if (mTTS != null) {
            mTTS.stop();
            mTTS.shutdown();
        }
        super.onPause();
    }
}
```



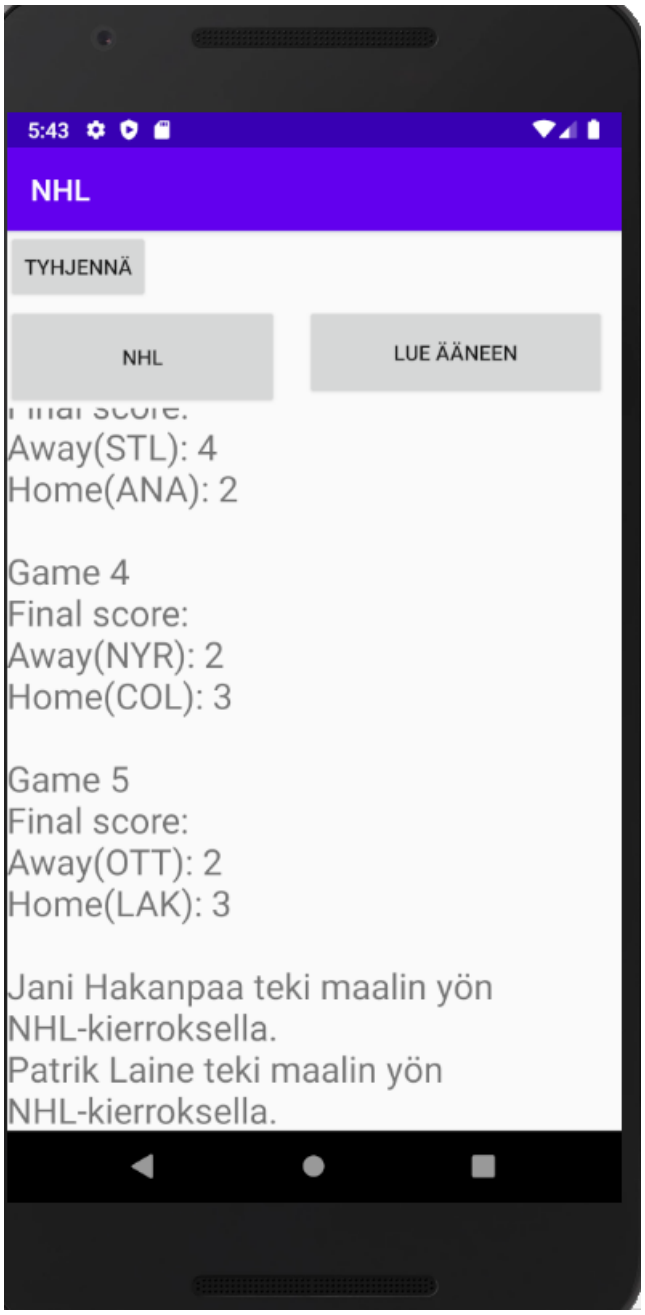
Kuva 6: Sovelluksen käyttöliittymä virtuaalisessa Android-puhelimessa.



Kuva 7: NHL-tulokset näkymä



Kuva 8: Kierroksen kolme ensimmäistä pelattua peliä.



Kuva 9: Kierroksen pelit neljä ja viisi sekä suomalaiset maalintekijät ja maalinsyöttäjät.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support;
using OpenQA.Selenium;
using OpenQA.Selenium.Support.UI;
using System.Text.RegularExpressions;

namespace NHL
{
          class Test
      {
          IWebDriver driver = new ChromeDriver();

          public void AvaaNHL()
        {
           driver.Url = "https://www.nhl.com/fi/scores/2019-03-06";

           System.Threading.Thread.Sleep(1000);
        }

           public void HaeJoukkue()
        {
            try
            {
                for (int i = 19; i < 30; i++)
                {
                                        string vieras = driver.FindEle-
                    ment(By.XPath("//*[@id='20180210" + i + "']/div[2]/sec-
                    tion[1]/section[1]/div/div/span[1]")).Text;
                    string vierasTulos = driver.FindEle-
                    ment(By.XPath("//*[@id='20180210" + i + "']/div[2]/sec-
                    tion[1]/section[1]/span")).Text;

                    string koti = driver.FindEle-
                    ment(By.XPath("//*[@id='20180210" + i +
```

```
                    "']/div[2]/section[1]/sec-
                    tion[3]/div/div/span[1]")).Text;
                    string kotiTulos = driver.FindEle-
                    ment(By.XPath("//*[@id='20180210" + i + "']/div[2]/sec-
                    tion[1]/section[3]/span")).Text;
                Console.WriteLine("");

                Console.WriteLine(vieras + " " + vierasTulos);
                Console.WriteLine(koti + " " + kotiTulos);
            }
        }

        catch
        {

        }
    }

    class Program
    {
        IWebDriver driver = new ChromeDriver();

        static void Main(string[] args)
        {
            var t = new Test();
            t.AvaaNHL();

            t.HaeJoukkue();
        }
    }
}
```

Koodiesimerkki 7, koodi joukkueiden ja tulosten hakemiseen C#:lla.