jamk.fi

Ennustemallin suunnittelu ja toteutus Bislenz-verkkopalvelussa

Leevi Kukkonen

Opinnäytetyö
Toukokuu 2021
Tietojenkäsittely ja tietoliikenne
Insinööri (AMK), Tieto- ja viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Kukkonen, Leevi	Opinnäytetyö, AMK	Toukokuu 2021
	Sivumäärä 64	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Ennustemallin suunnittelu ja toteutus Bislenz-verkkopalvelussa

Tutkinto-ohjelma

Tieto- ja Viestintätekniikka, insinööri AMK

Työn ohjaaja(t)

Kari Niemi, Häkkinen Antti

Toimeksiantaja(t)

Tridea Oy

Tiivistelmä

Digitalisaatio ja Internet ovat muuttaneet asiakkaiden käyttövaatimuksia tuotteilta. Pienten yritysten pitää yltää suurten palvelutasolle, mikä lisää työtaakkaa, mutta antaa myös mahdollisuuksia innovaatioon.

Toimeksiantajayritys Tridea Oy:n Bislenz-verkkopalvelu kokoaa asiakkaiden määrittämiä markkinointi- ja sometilastoja, tallentaa ne tietokantaansa ja kokoaa ne visuaaliseen muotoon sivustolle. Asiantuntijoita konsultoimalla tilasto todettiin riittäväksi tulevaisuusdatan generointiin ennustemallialgoritmilla, johon yritys sai valmiin ohjelmistokoodin yhteistyökumppaneiltaan.

Työn tehtävänä oli ottaa tämä ohjelmistokoodin sisältämä ennustemalli käyttöön Bislenz-verkkopalvelun AI Forecast -näkymässä ja kehittää näkymän ja ennustemalliin liittyviä tekniikoita asiakaskäyttöä varten. Tavoite oli toteuttaa näkymä, jossa käyttäjä voisi valita historiadatan ja generoida sen avulla tulevia datarivejä. Työ suoritettiin kehittävänä tutkimuksena, jossa keskityttiin yhden toimivan ratkaisun toteuttamiseen.

Tuloksena toteutetussa näkymässä asiakas voi määrittää haluamansa ennusteen, jonka tiedot tallennettiin tietokantatauluun. Tämän taulun avulla verkkopalvelin luo ennusteen ajastetusti joka yö käyttäen ARIMA-ennustemallia. Tulosdata tallennetaan tietokantaan, joka haetaan AI Forecast -sivulla latauksen yhteydessä.

Ennustemallin yhdistäminen sivustolle osoitettiin mahdolliseksi ja aikaansaatu ratkaisu toimi, vaikka ennustemallin automatisointi jäikin kesken. Lopputulos on kuitenkin jatkokäsiteltävissä työssä tehdyn toimintasuunnitelman mukaan.

Avainsanat (asiasanat)

Algoritmit, analyysi, ennusteet, JavaScript, R-kieli, regressioanalyysi, verkkopalvelut

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Kukkonen, Leevi	Bachelor's thesis	May 2021
		Language of publication: Finnish
	Number of pages: 64	Permission for web publication: x

Title of publication

Design and Implementation of the Forecast Model in the Bislenz Web Service

Degree programme

Bachelor of Engineering

Supervisor(s)

Kari Niemi, Häkkinen Antti

Assigned by

Tridea Oy

Abstract

Due to digitalisation and the internet, customer expectations for work services have changed. Small enterprises must fulfil the service expectations of large companies which adds to estimated workload but also adds possibilities for innovations to leverage.

Thesis was assigned by Tridea Oy, whose Bislenz-service compiles customer-defined data of marketing and social media statistics into a visual website. By consulting experts of analytics, the data was found sufficient for generating data using a forecast-model and provided the company with a tailor-made script.

The task was to develop the AI Forecast -view of the Bislenz-web service and the structure to allow the user to select history data and generate future values using the script. The work was carried out as a developmental study according to the solution-oriented nature of the topic. This was selected due to the assignment involving ready-made solution which was modified to solve the problem at hand.

In the resulted view, the given forecast model was used to select and define sources of data, with which server ran the forecast model. The resulting data was saved into the database so that the page could utilize the data on load.

Combining the forecast model to the page was demonstrated to be possible and the resulting page could be used on service, even if the forecast model automation was left unfinished. It is possible to finish the project by following the plans of action made during the project.

Keywords/tags (subjects)

Algorithms, analysis, forecast, JavaScript, R-language, regression analysis, web-service

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

Sanasto

Algoritmi: Ohjelmoitu kokonaisuus järjestyksessä ajettavia komentoja, joilla saavutetaan haluttu lopputulos, kuten esimerkiksi algoritmin aikana palautetut tai muutetut arvot.

ARIMA-malli: Aikajakson mukaisen datan ennustavaan analysoimiseen kehitetty malli, joka yhdistää autoregressiivisen ja liukuvan keskiarvon. Hyödynnetään ennustedatan generoinnissa.

Boolean: Tietokoneessa käytettyjen ohjelmointikielien perusmuuttujatyyppi, jolla käsitellään kyllä/ei -rakenne. Sen arvot voi olla true (tosi) tai false (epätosi). Joissain järjestelmissä Boolean-arvot korvataan tinyInt-muuttujatyypillä, jonka arvot ovat 1 (tosi) tai 0 (epätosi).

Differointi (Differencing): Analyyttisen geometrian toiminto, jossa epästationäärinen data muutetaan stationääriseksi. Tämä tapahtuu tyypillisesti korvaamalla arvot arvojen muutoksella.

Ennustemalli: Dataa ennustaessa usein hyödynnetty algoritmi, joka tekee ennustetun datan määritellylle ajanjaksolle. Malli käyttää datarivejä generoidakseen ennustettuja lukemia tulevaisuudelle. Sen lähteenä käytettävää dataa kutsutaan historiadataksi ja sen generoimaa dataa kutsutaan tulevaisuusdataksi.

Front-end: Sovelluksen tai palvelun osat, jotka käsittävät palvelun käyttöliittymän, sen ulkonäön ja siinä tehtävät toiminnot.

Funktio: sovelluskehityksessä tai matematiikassa esiintyvä rakenne, joka tuottaa tuloksen parametrilukeman mukaan. Funktiolla voidaan määritellä sarja komentoja,

jotka joudutaan toistamaan samassa järjestyksessä, jotta vältytään identtisten toimintasarjojen toistamiselta. Funktioita voidaan käyttää useamman parametrin kanssa tai ilman parametreja.

HTML (Hyper-Text-Markup-Language): Yleinen verkkosivujen rakenteen määrittelyyn soveltuva merkintäkieli, jolla määritellään verkkosivun sisältö. Verkkosivut koostuvat mm. teksti-, kuva-, otsikko-, painike- ja linkkielementtejä. HTML ei siis määrittele elementtienulkonäköä tai toiminnallisuutta, ainoastaan sisällön. Sen kanssa käytetään CSS- ja JavaScript-kieliä.

JavaScript: Yleinen ohjelmointikieli, jota hyödynnetään tyypillisesti verkkosivujen toimintojen toteuttamisessa. JavaScript on synkronoitu ohjelmointikieli eli sen tiedostot ajetaan samanaikaisesti verkkosivulla.

JSON: JavaScript Object Notation eli JSON on JavaScript-ohjelmistokielessä käytettävä malli, jolla käsitellään JavaScript-objekteja. Objektit koostuvat tunnisteista (tag), joille voidaan määrittää arvo (value). Tunnisteelle voidaan määrittää vain yksi arvo kerrallaan, mutta tätä arvoa voidaan muuttaa vapaasti käytön aikana. JSON-objektit voidaan määrittää omaan tiedostoonsa ”.json”-tiedostopäätteellä.

JSX: JSX eli JavaScript XML on JavaScriptin kehitetty laajennuskirjasto, jolla voidaan kirjoittaa HTML-merkintäkieltä sen omalla syntaksillaan JavaScript-sovelluksissa. JSX on standardi mm. React.JS-kirjastossa.

Käyttöliittymä: Kaikki sovelluksen tai verkkosivun osat, jotka näkyvät käyttäjälle, mukaan lukien käyttöliittymän tyylimäärittelyt, toiminnallisuudet sekä dynaamiset (muokattavat) ja staattiset (ei-muokattavat) elementit.

MySQL: Yleinen relatiivinen tietokantajärjestelmä, jota käytetään ohjelmistojen ja verkkopalveluiden tietovarastona mm. kirjautumisjärjestelmissä. Sisältää ominaisuudet tiedon tallentamiseen tietokantatauluina, joiden tietoa päivitetään taulujen välisten suhteiden mukaan.

Merkintäkieli: Merkintäkielillä tarkoitetaan verkkosivun tai tietokoneohjelman sisällön määrittelemiseen tarkoitettuja tietorakenteita. Merkintäkielet eivät kuulu ohjelmointikielien joukkoon, koska ne eivät itsessään saa tietokonetta suorittamaan algoritmeja tai muuttamaan tietoja. Merkintäkieliä ovat mm. HTML ja Markdown.

Palvelin: Tietokone, jonka dataa lähetetään internetissä kyselyjen (query) avulla toisille tietokoneille selaimen avulla. Palvelimet voidaan laittaa tarjoamaan WWW-sivua julkisesti, jolloin ne näkyvät julkisessa internet-osoitteessa. Palvelimella tarjotaan myös tietokantapalveluita.

Parametri: Funktiolle tai yhtälölle annetut tiedot, joiden perusteella lopputulosta muutetaan. Funktio voi toimia myös ilman parametrejä.

R-kieli: Ohjelmointikieli, jota hyödynnetään tyypillisesti matemaattisessa laskennassa ja mallien kuvantamisessa.

Runtime-kirjasto: Ohjelmiston ajo-ohjelmisto, jolla helpotetaan tai nopeutetaan ohjelmointirakenteiden toteuttamista. Runtime-kirjastoilla voidaan tyypillisesti hakea käyttäjien tekemiä ohjelmaskriptejä eli kirjastoja, joita voidaan käyttää vapaasti sovelluksissa.

Sessio: Palvelimen määrittelemä ajokokonaisuus, joka alkaa selaimen avaamisesta aina sen sulkemiseen asti. Session aikana voidaan määritellä sessiomuuttujia, jotka

säilyvät palvelimen muistissa. Sessio ja sessiomuuttujat suljetaan, kun käyttäjän selain on pysynyt suljettuna palvelimessa määritellyn ajan.

Sessiomuuttuja: Selaimen käytön aikana tallennettavat tiedot, jotka sivu muistaa sivujen vaihtamisen aikana aina session loppumiseen asti. Sessiomuuttujiin tallennetaan mm. kirjautumiset tai käyttäjän tekemät asetusvalinnat.

Stationäärisyys: Analytiikan termi, jolla kuvataan datan muutoskohtaisuutta. Stationäärisessä datassa datan keskiarvo on vakio, kun taas epästationäärisessä datassa keskiarvo vaihtelee datan aikana. Jotkin ennustemallit vaativat datan olevan stationääristä toimiakseen.

Tietokanta: Tietorivien varastoimiseen käytettävä tietovarasto, jolla voidaan hallita sovelluksen tai verkkosivun tallentamaa dataa. Tietokantaa käytetään esimerkiksi historiatiedon, postauksien, käyttäjätietojen ja sessioiden tallentamiseen.

Verkkopalvelu: Verkko-osoitteessa toimiva sovellus, jota käytetään selaimella. Niihin liittyy yleensä kirjautumisjärjestelmä ja asiakkaalle mukautettuja sivuja.

1 Työn lähtökohdat

1.1 Tausta ja toimeksiantajan kuvaus

Digitalisaatio ja Internet ovat muuttaneet radikaalisti toimintamalleja ja kulutuskäyttäytymistä toimialasta riippumatta. Maailman parhaiden palveluiden tasosta on tullut uusi standardi kaikille palveluille. Ihmiset ovat tottuneet nopeaan, jopa reaaliaikaiseen, monikanavaiseen ja personoituun palveluun ja kaikkien yritysten on pyrittävä maailman huippujen ja johtavien yritysten palvelutasolle vastatakseen kuluttajien muuttuviin kulutustottumuksiin. Toisaalta tämä tarjoaa pienillekin yrityksille mahdollisuuden heilauttaa markkinoita nopeasti. (Kananen & Puolitaival, 2019, 73)

Tridea Oyn kehittämä Bislenz-verkkopalvelu on kehitetty selkeyttämään yritysten markkinointitilastoja ja asiakaspolkua keräämällä asiakkaiden käyttämät markkinointitiedot ja -tilastot yhteen paikkaan. Palvelu hakee asiakkaiden syöttämillä tietolähdekanavien tunnuksilla markkinointikanavien, varausjärjestelmien ja asiaspalautteiden lukemat ja esittää ne verkkopalvelun kuvaajissa ja taulukoissa. Bislenz tarjoaa myös työkaluja asiakaspalauttekyselyiden luomiseen, niiden yhteenvetoihin ja maailmanlaajuisen matkailudatan kartoittamiseen. Sovellusta kehitetään jatkuvasti tukemaan asiakkaiden tarpeita. (Bislenz, 2021)

Bislenzin vuonna 2020 toteutetuissa kehityskeskusteluissa konsultoitiin data-analytiikan ammattilaisia. Keskusteluissa huomattiin että Bislenz-verkkopalvelun tietokantaan tallennettu data soveltuu ARIMA-ennustemallialgorimin käyttöön ja näiden ammattilaisten R-ohjelmointikielellä kehitetty algoritmi sovitettiin verkkopalveluun. Algoritmin käyttöönotto jäi kuitenkin kesken Bislenzin kiireellisempien kehitystehtävien vuoksi ja sen loppuunvieminen jätettiin opiskelijaprojektiin.

Toimeksiantaja Tridea Oy on markkinointiin ja asiakaskokemuksen kehittämiseen erikoistunut yritys, joka tarjoaa yrityksille markkinoinnin asiantuntijapalveluita, asiakaskokemuksen koulutuksia sekä markkinoinnin ja matkailun seuraamisen työkaluja. Yrityksen päätavoitteena on selkeyttää, ohjeistaa ja kehittää yritysten asiakaskokemusperustaa ja markkinointia. Näin yritykset saavat keskittyä tiedon keräämisen sijaan markkinointistrategian suunnittelemiseen. (Tridea Oy, 2021)

1.2 Tehtävänanto

Työn aloitusvaiheessa Bislenz-verkkopalveluun oli alustettu asiakkaille AI Forecast-niminen näkymä, jossa näytettiin yhteenveto algoritmin aikaisemmin generoimasta datasta. Asiakas määritteli itselleen merkittävät tietolähteet ja päivitti ne käyttöliittymässä, jotta sovellus kykeni muuttamaan niitä seuraavaa ajoa varten. Näkymä toimi, mutta oli puutteellinen asiakaskäyttöön seuraavilla tavoilla:

- Datamäärän luomiseen algoritmia käyttäen kului liikaa aikaa (n. 10 minuuttia) Bislenz-palvelimella
- Datan generointialgoritmin parametrejä ei voitu määrittää tai hienosäätää asiakkaan toimesta
- Algoritmin ajamista ei automatisoitu, jolloin kehittäjä joutui päivittämään algoritmin generoiman datan manuaalisesti
- Yhteenvetosivun komponentteja ei ollut yhtenäistetty Bislenz-palvelun muun käyttöliittymän mukaiseksi.

Työn tehtävänä on kehittää ennustavan analytiikan algoritmia hyödyntävää generointityökalua, generointiprosessia, näkymän käyttöliittymää ja tietokantarakennetta, jotta se korjaa yllä mainitut käyttöongelmat. Jos työn aikana jokin näistä ongelmista ilmenee mahdottomaksi korjata, määritellään vaihtoehtoisia tapoja tämän ongelman ratkaisuun tai ohittamiseen.

Aiheen toteutuksessa on luvallista käyttää sovelluksessa valmiina löytyviä materiaaleja. Toteutuksesta ohitetaan myös tutkimuksen aikana kehitetty Bislenz -matkapuhelinsovellus, johon ennustemalli toteutetaan erikseen.

Työn dokumentoinnissa on kunnioitettava toimeksiantajan, toimeksiantajan työntekijöiden sekä asiakkaiden yksityisyyttä. Tutkimuksessa ei voida paljastaa yksityisistä lähteistä tallennettua tietoa, jos niiden omistaja on mahdollista selvittää muista lähteistä.

1.3 Tutkimuksen tavoitteet

1. Miten ARIMA-ennustemallialgoritmi otetaan käyttöön verkkopalvelussa?
2. Kuinka ennustemallialgoritmi toimii?
3. Soveltuuko algoritmi verkkopalvelun tarpeisiin?

Aihe sisältää piirteitä sekä konstruktiivisesta että kehittävästä tutkimusmenetelmästä, jotka perustuvat tutkimuksessa esiteltyihin ongelmien ratkaisuun.

Konstruktiivisessa tutkimuksessa työ keskittyy toimeksiantajan määrittelemien ongelmien ratkaisuvaihtoehtojen tutkimiseen. Tutkimuksen tarkoitus on tutkia ongelman lähteitä, erilaisia ratkaisukeinoja sekä näiden tehokkuutta käytössä. Tutkimuksen lopputulosta käytetään vastaavissa tilanteissa, jotta toimenpiteiden valinta ja vertaaminen helpottuu. (Bister. T. 2019)

Kehitystutkimuksessa työ keskittyy valmiin sovelluksen, palvelun tai rakenteen parantamiseen, kohentamiseen tai loppuun viemiseen. Työn tuloksena syntyy ennakkotapaus, jota voidaan hyödyntää muissa saman aiheen projekteissa. Tutkimuksessa esitellään toteutus ja siihen vaadittavat teknologiat ja teoriapohja sekä ratkaisun toteutus ja tulokset. (Mt.)

Kehittävässä- ja konstruktiivisessa tutkimuksessa on paljon samankaltaisuuksia. Molemmat menetelmät keskittyvät käytännönläheisyyteen ja aikaisempiin toteutuksiin ja pyrkivät esittämään ratkaisun määriteltyyn ongelmaan. Erot ovat aiheen käsittelyssä. Kehittävässä tutkimuksessa tutkitaan ratkaisun tuottamista, sen tilanteen tarkkaa esittelyä ja soveltuvuutta, kun taas konstruktiivisessa tutkimuksessa tarjotaan useita ratkaisuvaihtoehtoja, joita vertaillaan keskenään. (Mt.)

Tehtävänanto on ratkaisujohteinen, mutta käsittää ratkaisuvaihtoehtojen tarkastelemisen sijaan yhden toimivan ratkaisun löytämisen ja sen toimivuuden arvioimiseen. Työhön määritellyissä tavoitteissa ei vaadita ratkaisulta konstruktiiviseen tutkimukseen vaadittavaa yleistettävyyttä, joten sen käyttö ei ole perusteltavaa. Niinpä tutkimus suoritetaan kehittävän tutkimusmenetelmän mukaan ratkaisun löytämiseen.

Tutkimuksessa tarvittavat tiedonlähteet kerätään julkisista tiedonlähteistä hyödyntäen alan kirjallisuutta, kirjaston digitaalista aineistoa sekä luotettuja internet-lähteitä. Internetlähteissä suositaan lähteitä, jotka tulevat tutkijoilta tai kehittäjiltä itseltään. Valinnan yhteydessä lähteistä arvioidaan kriittisesti kirjoittajien motiivi, jotta kirjoituksia voidaan pitää asiasisältöön soveltuvina. Työn aikana tekemiä prosesseja voidaan selittää käyttämällä aiheeseen liittyviä tekstejä.

2 Ennustemalli

2.1 Yleistä

Ennustava analytiikka (Predictive Analytics) on tilastollisen analytiikan osa-alue, jossa pyritään selvittämään tulevaisuutta matemaattisilla menetelmillä. Tutkimalla nykytilannetta, aikaisempia tuloksia sekä arvojen välisiä muutoksia löydetään säännönmukaisuuksia, joiden avulla voidaan laskea todennäköisimmät tulevaisuusdatan arvot. (Hyndman & Athanasopoulos, 2021)

Ennustavassa analytiikassa ennustamismenetelmät jaotellaan kvantitatiiviseen ja kvalitatiiviseen ennustamiseen. Jos tilanteessa ei ole käytettävissä olevaa dataa tai olemassa oleva data ei ole ennusteelle merkittävää, käytetään kvalitatiivisen menetelmän ennustamista. Kvantitatiivisessa ennustusmenetelmässä tutkimus pohjustetaan aiemmin kerättyyn historiadataan. (Mt.)

Ennustemalli (Forecast Model) on nimitys kvantitatiivisen ennusteen algoritmille, joka tutkii historiadataa ja sen muutoksia. Ennustemalli pyrkii kuvaamaan tarkasti vain jotakin tiettyä todellisuuden ilmiöön vaikuttavaa mekanismia, joka muuttuu ajan mukaan. Käytössä ennustemalli luo kahden muuttujan välisen muutoskuvaajan ja generoi tulevaisuuden datalukemia sovittamalla ne taulukkoon tämän kuvaajan mukaisesti. Ennustemallin sisällä käytetyt tekniikat voivat vaihdella tilanteen ja tarkoituksen mukaan. (Kuorikoski & Reijula. 2020)

2.2 Käyttötapaukset

Ennustavan analytiikan sovellukset soveltuvat minkä tahansa säännöllisesti tallennetun datan tallentamiseen. Jos kahden muuttujan välillä huomataan riippuvuus, voidaan niiden välille myös luoda ennustemalli.

Tyypillisimmät käyttökohteet matemaattisessa ennustamisessa ovat markkina- ja myyntilukemien, tuotannon, säätietojen, liikennemäärien tai datan käytön ennustamisessa. Uusia käyttötapauksia ennusteille keksitään kuitenkin jatkuvasti.

Usein käyttötapauksiin liittyy tarkentavia kysymyksiä, joita kannattaa pitää mielessä ennusteen kohdetta pohtiessa. Hyndman & Athanasopoulos painottavat suunnittelijoita määrittelemään tarkennuksia ennusteen tarpeesta. Esimerkiksi, jos ennusteeksi halutaan lukemia kaupan alan tuotantoympäristöstä, on hyvä selvittää, ennustetaanko:

- tuotteen vai tuoteryhmän mukaan?
- myymälän, alueen, vai kokonaismyynnin tarkkuudella?
- viikoittain, kuukausittain vai vuosittain?

Kannattaa myös pohtia, kuinka kauas tulevaisuuteen halutaan ennustaa. Tarvitaanko ennusteet kuukaudeksi, kuudeksi kuukaudeksi vai kymmeneksi vuodeksi etukäteen? Tilanteessa voidaan tarvita erityyppinen malli riippuen siitä mikä ajanjakso on tärkein. (Hyndman & Athanasopoulos. 2021)

2.3 ARIMA

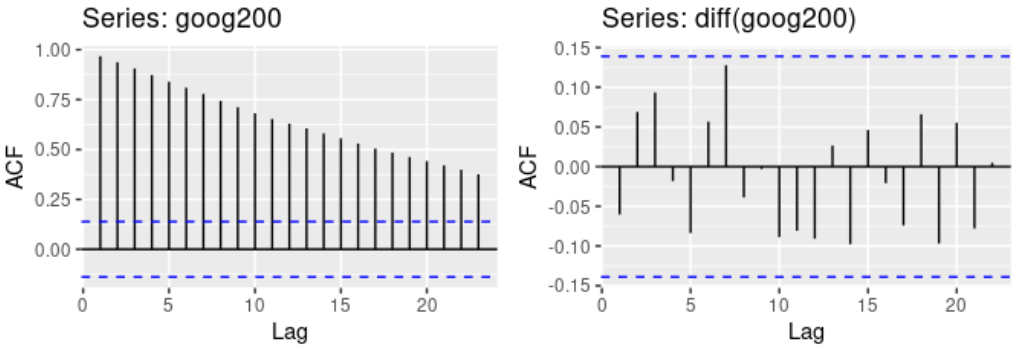
Ennustaminen voidaan suorittaa useilla eri menetelmillä, joista saadut arvot vaihtelevat. Menetelmien erilaisuuden vuoksi ennusteita varten generointituloksia yhdistellään keskenään, jotta aikaansaatu data on luotettavaa. (Hyndman & Athanasopoulos, 2021)

Autoregressive integrated moving average eli ARIMA on analyysimalli, jolla pyritään kuvaamaan datan sisällä ilmaantuvia korrelaatioita. Hyndmanin & Athanasopoulos kuvaavat ARIMA-malleja yhdeksi yleisimmistä ratkaisuista ajan mukaan tehtäviin ennusteisiin. (Mt.)

ARIMA yhdistää autoregression ja liikkuvan keskiarvon generointimallit, jotka lasketaan integroidusta historiadatasta. ARIMA-ohjelmoinnissa integroinnilla tarkoitetaan lähdedatalle tehtävää differointia, jota hallitaan ennustemallille syötettävällä parametrilla *d*. (Mt.)

Ennen differointia on hyvä selittää, mitä datan stationäärisyys tarkoittaa. Stationäärisen datan lukemien keskiarvo on vakio, mutta jos datan keskiarvo nousee tai laskee, se on epästationääristä. Kuviossa 1 on esitelty data, goog2020, jonka arvojen keskiarvo vähenee tasaisesti lukemien mukaan. Kuvaaja on siis epästationäärinen.

Kun tämä kuvaaja differoidaan saamme uuden arvojoukon diff(goog2020) Kuviossa 1, joka näyttää arvojen sijaan peräkkäisten arvojen välisen muutoksen. Tässä kuvaajassa keskiarvo pysyy lähellä nollaa, joten data on stationäärinen ja se soveltuu ARIMA-ennustemallin käyttöön. (Mt.)



Kuvio 1. Differoinnin vaikututus lähdedataan (Hyndman & Athanasopoulos, 2021, https://otexts.com/fpp2/stationarity.html)

Differointi on data-analytiikan termi, jolla tarkoitetaan epäjaksollisen datan muuttamista jaksolliseksi muuttamalla lähdedatan lukuarvot arvojen väliseksi muutokseksi. Lopputuloksena syntyy stationäärinen data, jota voidaan hyödyntää ARIMAN laskutoimituksissa, joita autoregressio ja liikkuva keskiarvo tarvitsee. Hyödyntäen differointia ARIMA-malleilla voidaan tutkia myös ei-stationäärisen datan muutoksia. (Mt.)

Autoregressio-mallilla lasketaan uudet arvot muuttujan aikaisempien arvojen mukaan. ARIMA-mallissa voidaan määritellä, montako aiempaa riviä otetaan huomioon autoregression tutkimien rivien tarkastelussa. Autoregressiota hallitaan ARIMA-mallissa parametrillä p (Mt.)

Liikkuva keskiarvo -mallissa keskitytään aikaisempien arvojen sijaan arvojen välisten ennustusvirheiden laskemiseen. Ennustevirheellä tarkoitetaan ennustavassa analytiikassa ennustusmallirivin eroa todelliseen lukemaan. Liikkuva keskiarvo käy läpi ajon aikana ARIMA-ennustemallille syötetyn datan, luo ennusteen ja laskee sitten ennustevirheen vähentämällä oikean rivin ennusteesta. ARIMA-mallissa voidaan määritellä, montako aiempaa ennustevirhettä otetaan huomioon liikkuvan keskiarvon tarkastelussa. Tämä tehdään käyttämällä ennustemallille annettavaa parametriä q. (Mt.)

ARIMA-ennustemalli voidaan laskea matemaattisesti yhtälöllä, joka näytetään kuvissa 2

$$y'_t = \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \quad (1)$$

missä y' on differoitu taulukon arvo,

ϕ on muutos edellisestä arvosta,

θ on ennusteen laskettu virhe edellisestä arvosta,

ε_t on valkoista kohinaa, joka tarkoittaa satunnaislukua.

Kuvio 2: ARIMA-ennustemallin laskukaava (Hyndman & Athanasopoulos, 2021, https://otexts.com/fpp2/non-seasonal-arima.html)

ARIMA-malli on yhdistetty muutamaan muuhun ennustemalliin. SARIMA eli Seasonal ARIMA yhdistää ennustemalliin tarkasteluun kausiluontoisuuden lisäparametreilla P, D ja Q. Nämä parametrit lisätään ARIMA-ennustemalliin kertomalla ne ARIMA-yhtälössä.

ARIMA-mallit yhdistetään yleensä myös dynaamisiin ennustemalleihin. Näissä rakenteissa voidaan ottaa huomioon muita dataan liittyviä sääntöjä dummy-muuttujien avulla. Dummy-muuttuja on datariveihin lisättävä uusi rivi, jolla tunnistetaan poikkeukselliset datarivit. Niiden ovat yleensä joko Boolean-arvoja tai TinyInt-arvoja riippuen dummy-muuttujan luonteesta. Yleisimmät käyttötapaukset niille ovat esimerkiksi viikkonumeron, juhlapäivän, epäaktiivisen otoksen tai loma-ajan tunnistaminen.

2.4 Toimintaperiaate

ARIMA-malli ajetaan ARIMA (p, d, q) -muodossa, jossa p on autoregression vertailulukemien määrä, d on lähdedatan differointien lukumäärä ja q on liikkuvan keskiarvon

vertailulukemien määrä. Kaikki nämä arvot ovat positiivisia kokonaislukuja ja voivat olla myös nolla-arvoja.

Esimerkiksi: Ajetaan komennolla ARIMA (3,3,3). Ennustemalli differoi lähdedatan kolme kertaa, eli suorittaa erotuksen jokaiselle taulukon arvolle edellisen arvon kanssa ja saa aikaan uuden taulukon. Tämä operaatio toistetaan kolmesti. Differoitu taulukko annetaan autoregressio-mallille, joka laskee jokaisen uuden arvon erottamalla kolmen viimeisen luvun mukaan. Differoitu taulukko annetaan myös liukuva keskiarvo -mallille, joka laskee jokaisen uuden arvon kolmen viimeisen ennustevirheen mukaan. Kerätyt arvot tallennetaan taulukkoon, joka palautetaan ennustettuina datariveinä.

Ennustemallin ajamisessa on hyvä pitää mielessä datan tarkoitusperä, jotta osataan valita oikeat arvot ajamiseen. Arvojen nostaminen lisää laskentamäärää eksponentiaalisesti, mutta tuottaa tarkempia tuloksia. Dataa ei pidä myöskään yliopettaa, jolloin arvot vaihtelevat vähän tai ei lainkaan.

ARIMA-rakenteen käytössä voidaan pudottaa jokin parametreistä asettamalla se arvoon 0, jolloin se jätetään pois algoritmin ajamisesta. ARIMA-ennustemallin ajamisessa on muutama erikoistapaus, jotka eritellään taulukossa 1.

Taulukko 1. ARIMA-ennustemallin poikkeustapaukset

Ennustemallin komento:	**Aikaansaatu kuvaaja:**
ARIMA (0, 0 ,0)	Valkoista kohinaa (satunnaisesti generoitu data)
ARIMA (0, 1, 0)	Satunnaiskulku (satunnaisesti generoitu data + edellinen arvo)
ARIMA (p, 0, 0)	Autoregressio
ARIMA (0, 0, q)	Liikkuva keskiarvo

2.5 Vaatimukset

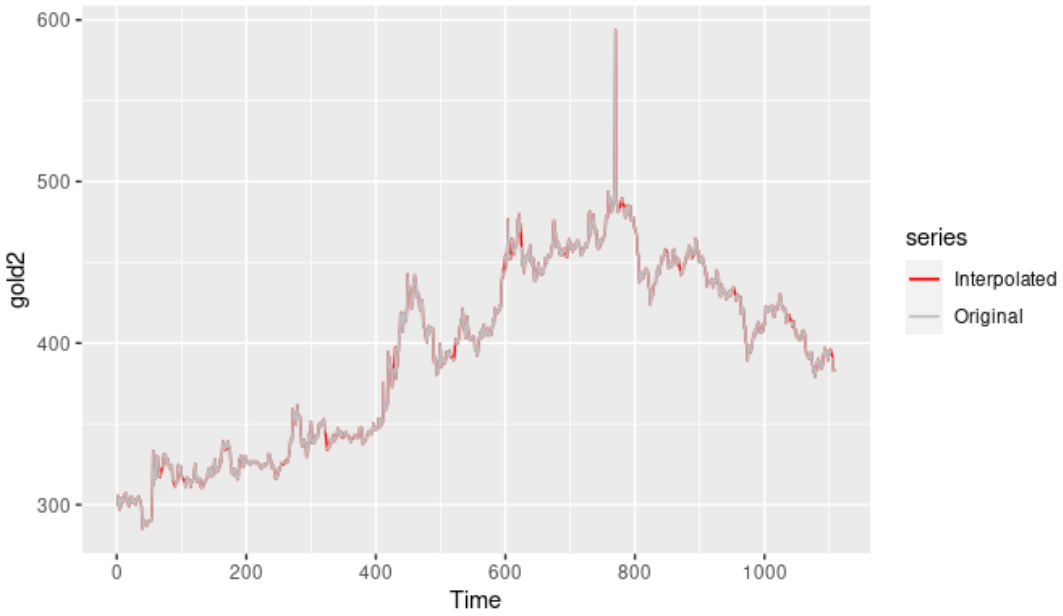
ARIMA-ennustemallin ajamiseen vaaditaan ennustemallin parametrit p, d ja q sekä säännöllinen otos dataa, joka sisältää mahdollisimman vähän nolla-arvoja ja tyhjiä arvoja. Lähdedatan pitää olla tarpeeksi pitkä ennustedatan generoimiseen, jotta ennustemallin tuotokset ovat luotettavia. Vaikka ennustemalli toimisikin pienen datamäärän kanssa, tämä data on lähes aina käyttökelvotonta. Dataa voi olla myöskin liian paljon, jolloin generointi vie enemmän aikaa ja tulosdata vaihtelee liian vähän. Nyrkkisääntönä on, että lähdedataa on oltava enemmän kuin generoitavia rivejä, mutta tarkka lukumäärä on tutkittava tilanteen mukaan.

Jos huomataan tilanne, että data ei ole säännöllinen, puuttuvat arvot korvataan nolla-arvoilla datan säännöllisyyden ylläpitämiseksi. Tämä toimenpide saa ennustemallin toimimaan, mutta pyöristää generoitua dataa kohti nollaa.

2.6 Heikkoudet ja rajoitteet

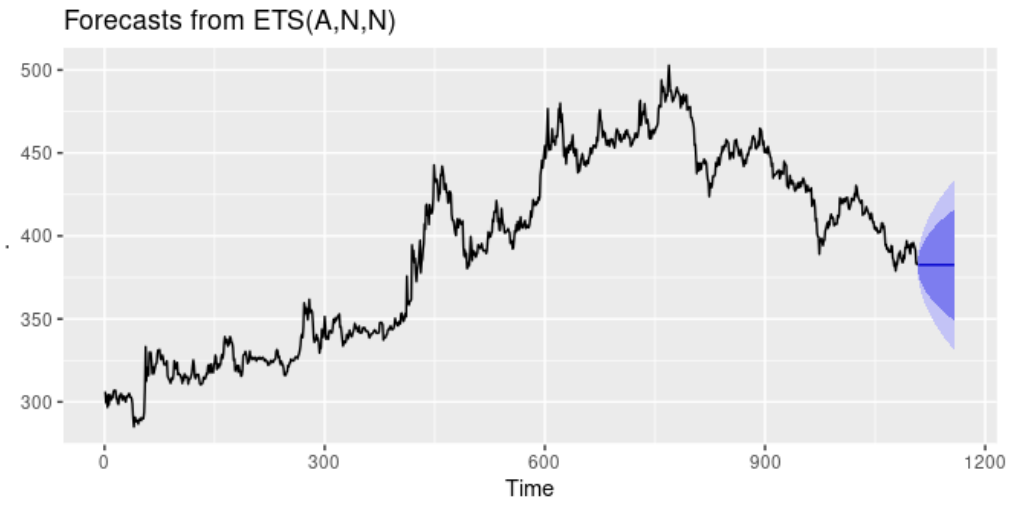
Vaikka nolla-arvot eivät usein aiheuta virheilmoituksia ennustemalleissa, niiden haittavaikutukset voivat pilata generoidun datan luotettavuuden. Nolla-arvojen olemassaolo painottaa ennustemallin keskiarvoa kohti nollaa ja ottaa nämä lukemat kausittaisena vaikutuksena. Esimerkiksi jos lähteenä käytettävästä historiadatasta puuttuu kuukauden mittainen otos, algoritmi huomioi tämän ennustaessaan samaa kuukautta ensi vuonna. (Hyndman & Athanasopoulos, 2021)

Nolla-arvojen ja tyhjien arvojen lisäksi datassa saattaa esiintyä kuvion 3 mukaisia outlier-arvoja, jotka poikkeavat muista lähdedatan arvoista mittaus- tai järjestelmävirheen vuoksi.



Kuvio 3. Outlier-lukema taulukossa (Hyndman & Athanasopoulos, 2021, https://otexts.com/fpp2/missing-outliers.html)

Jos outlier-lukema ei ole järjestelmävirhe, kehittäjän kannattaa selvittää sen alkuperä tarkasti ennen niiden käsittelyä. Outlier-lukemat voivat olla hyvä työkalu löytämään mittauslaitteiden ongelmia. Kun on todettu, että outlier ei johdu järjestelmän ongelmasta, se voidaan korvata tyhjällä rivillä tai viereisten rivien keskiarvolla. R-ohjelmistokielessä, jota käytetään ennustemallien generointiin, on myös komento tsclean(), jota voidaan käyttää outlier-lukemien korjaamiseen. Kuviossa 4 on korjattu versio kuvion 3 datasta.



Kuvio 4: Ennustemallin lähdedata, jossa Outlier-arvo on poistettu. (Hyndman & Athanasopoulos. 2021., https://otexts.com/fpp2/missing-outliers.html)

Ennustemalli käyttää lähteenään historiadataa, mutta sen määrä voi aiheuttaa ongelman asetettujen parametrien kanssa. Jos algoritmin parametrit asetetaan liian korkealle ja/tai algoritmille syötetään liikaa dataa, generointiaika kasvaa huomattavasti ja tulosrivit ovat liian kaavamaisia. Jos taas käytetään liian vähän historiadataa, lopputuloksena syntyy epäluotettava ennuste, joka generoi nolla- ja satunnaisrivejä. Oikea datamäärä löydetään kokeilemalla parametreja ja datamääriä tilanteen mukaan. (Hyndman & Athanasopoulos, 2021)

Suuri osa ajanjaksoihin jaetuista datasta voi aiheuttaa ongelmia ennustemalleille muuttujien vaihtelevuuden vuoksi. Esimerkiksi vuoden viikkomäärä vaihtelee vuoden mukaan ja jos halutaan ennustaa juhlapäiviä, niiden paikka datassa vaihtelee vuosittain. Näihin ongelmien ratkaisut kannattaa valita tilanteen mukaan. Viikkotasoisen datan hallintaan voidaan ottaa käyttöön avustavia STL tai TBATS-rakenteita, joilla voidaan käsitellä epätasaisia jaksoja. Juhlapäivien ratkaisuun kannattaa yleensä määritellä erillinen dummy-muuttuja, jolla tarkastellaan haluttua ominaisuutta datasta. Esimerkiksi juhlapäivään voidaan luoda dummy-muuttuja *isHoliday*, joka asetetaan arvoon 1 aina kun päivä vastaa juhlapäivän päivämäärää. Muuten se jätetään arvoon 0. (Mt.)

Algoritmilla ennustamisen heikkoutena on myös kehitetyn datan tilanneriippuvaisuus, jossa dataan vaikuttaa ulkoisia ja näennäisesti tilanteeseen riippumattomia vaikutuksia. Kuorikosken ja Reijulan (2020) artikkelissa mainitaan, että teoreettiset mallit pyrkivät kuvaamaan tarkasti vain jotakin tiettyä todelliseen ilmiöön vaikuttavaa mekanismia ja niitä ei tästä syystä ole mielekästä pyrkiä sovittamaan sellaisenaan havaintoaineistoon. Tähän tarvitaan myös kausaalimalli, joka perustuu muuttujien väliseen vaikutukseen eli mitkä ovat syitä ja mitkä seurauksia tilanteelle. (Kuorikoski & Reijula. 2020.)

3 Bislenz-verkkopalvelu

3.1 Käyttötapaukset

Bislenz-verkkopalvelu hakee sovellukseen tilastodataa yleisimpien sosiaalisen median kanavien ja tietopalveluiden rajapinnoista ja tuo ne käyttöliittymään. Käyttäjä löytää markkinointitilastot palvelusta ja tutkii lukemien kehittymistä ajan mukaan. Käyttäjä voi useimmissa näkymissä valita aikavälin, jonka aikaiset lukemat näytetään sivustolla.

Verkkopalvelun sisältö näyttää kaikille asiakkaille erilaiselta, koska Bislenz mukautuu käyttäjän tietolähteiden mukaan. Palvelun asiakaskunta ei ole vielä suuri, tietolähteet yhdistetään asiakkaiden kanssa käydyissä käyttöönottopalavereissa. Näissä palavereissa asiakkaat antavat tiedonhakuun tarvittavat hakutiedot, jotka lisätään järjestelmään ja määrittävät asiakkaalle näytettävät sivut. Lopuksi asiakkaalle lähetetään asiakastunnukset, joilla pääsee kirjautumaan palveluun.

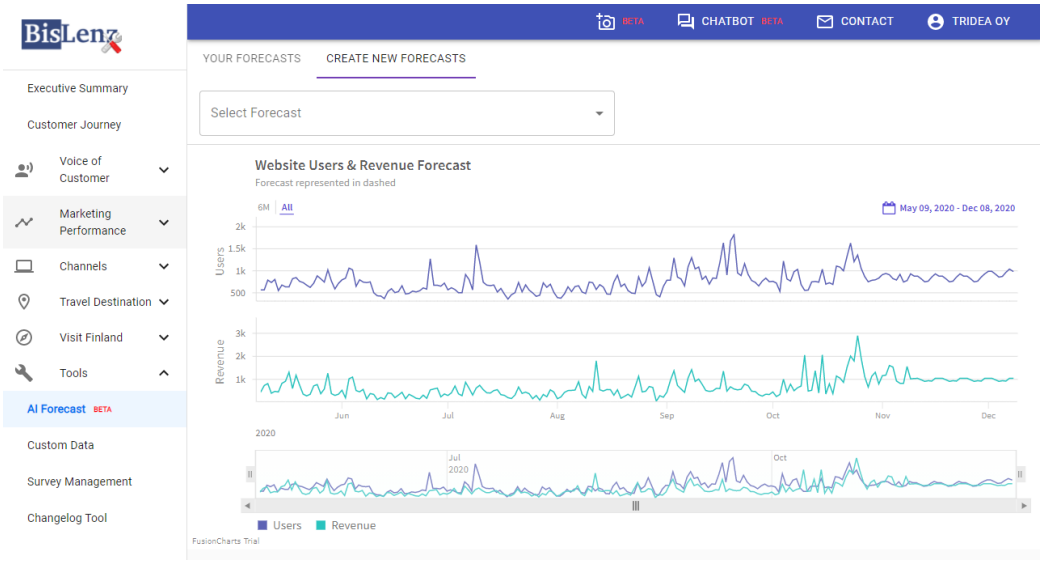
Bislenz-pystyy hakemaan dataa useista eri rajapinnoista, joista suurin osa tuodaan Supermetrics-tiedonhakupalvelun kautta, jotka esitellään taulukossa 2. Asiakaskäytössä ennustemallin pitäisi pystyä valitsemaan lähdedataa kaikista taulukon tiedonlähteistä, jotka täyttävät ennustemallin generoinnin käyttövaatimukset.

Taulukko 2. Bislenz-verkkopalvelussa käytettävät tietolähteet

Supermetrics-palvelusta haetut tiedonlähteet	Rajapinnoista haetut tiedonlähteet
• Facebook Insights • Facebook Ads • Facebook Public Data • Instagram Insights • Instagram Public Data • LinkedIn Ads • LinkedIn Company Pages • Google My Business • Google Ads • Google Search Console • Twitter insights • Twitter Ads • Youtube • Mailchimp	• Johku • Hotellinx • Shopify • Bokun • Winres • Google Trends • Tilastokeskus

Bislenz-palvelussa ennustemallia käytetään asiakaskäytön kontekstissa, jossa haetaan tietolähteitä yksi kerrallaan. Data on tallennettu päivittäin ja käytettävät lukemat ovat kokonais- tai desimaalilukuja. Generointiin käytettävää tietolähteiden historiadataa on hakukriteereistä riippuen aina kuukaudesta vuoteen asti. Käytössä ennustemallin täytyy siis toimia joko erikokoisten datalähteiden kanssa tai suodattaa pois datalähteet, jotka eivät sisällä tarpeeksi generointidataa. Asiakkaat tarvitsevat ennusteisiinsa mahdollisimman tuoreen datan, joten sitä on päivitettävä muutoksen yhteydessä päivittäiselle datalle vähintään viikoittain ja viikoittaiselle datalle vähintään kuukausittain.

Ennustemallin käyttöön on varattu Bislenz-palvelun Tools-kategoriasta löytyvä AI Forecast -näkymä. Näkymässä oli työn alkuvaiheessa kuviossa 5 esitelty ennustemallin testisivu, jossa näytettiin ennustemallin avulla generoitua testidataa kuvaajassa.



Kuvio 5. Bislenzin AI Forecast-näkymä työn aluvaiheessa

3.2 Sovellusrakenne

Bislenz-verkkopalvelu jaotellaan yksityiseen Bislenz-palvelimeen ja julkiseen Bislenz-sivustoon. Verkkopalvelun peruskäytössä nämä kokonaisuudet suorittavat tiedonhakuja keskenään. Osiot on erotettu toisistaan tietoturvasyistä, jotta asiakkaiden tietoja ei voida löytää manipuloimalla verkkosivustoa tai evästeitä.

Bislenz-palvelin sisältää sovelluksen tietokannan ja sen avulla suoritetaan kaikki asiakastietoja vaativat tietokantahaut, kirjautumistoiminnot, sähköpostilähetykset ja tietovaraston päivitykset. Bislenz-verkkosivusto sisältää verkkosivustolla käytettävät rakenteet ja tekee piilotettuja hakuja Backend-palvelimelle ajon aikana.

Bislenz-frontend sisältää Bislenzin käyttöliittymän, jossa käytetään sovelluksen komponentteja ja sivuja. Komponentit ovat sivuilla käytettäviä yleisiä elementtejä, jotka

on erotettu sivuista. Niitä voidaan käyttää kaikilla sivuilla, myös ennustemallisivulla tarvittaessa. Jos Bislenz-palvelussa aiotaan luoda uusia sivuja, on syytä käyttää komponenttikirjaston komponentteja omien komponenttirakenteiden sijaan, jotta ladattavan sivuston koko ei kasvaisi kohtuuttomuuksiin. Bislenz-sivustolla käytettävät komponentit sisältävät esimerkiksi:

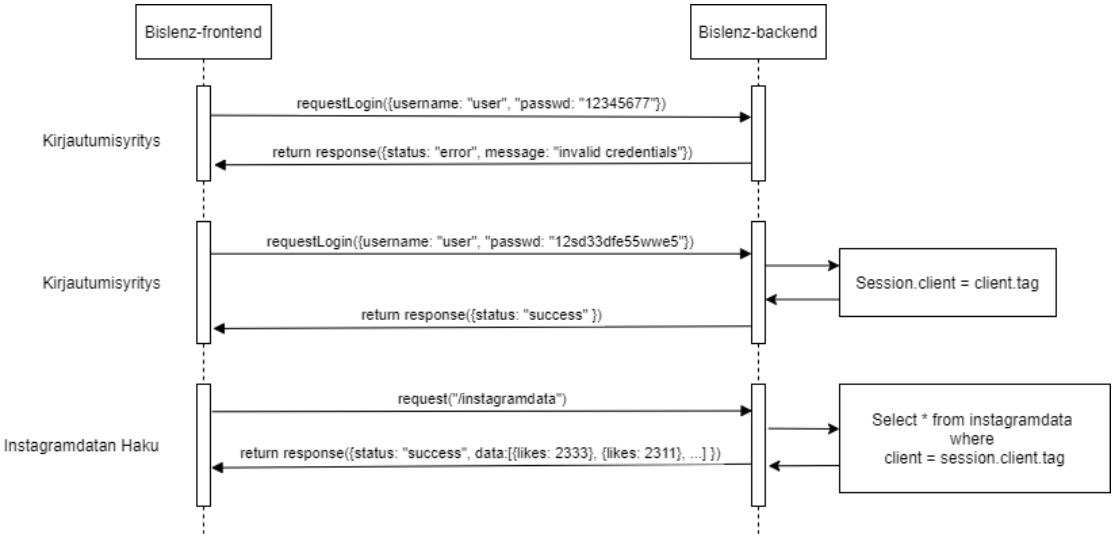
- Sivun reunan navigaatiovalikko, jolla vaihdetaan sivun näkymää
- Aikavalitsimen sivustolla haetun datan suodattamiseen
- Välilehtivalikon, jos näkymään tarvitaan
- Kuvaajia ja karttoja datan visualisointiin

Bislenz-verkkopalvelun näkymät ovat sovelluksessa näytettäviä sivuja, jotka käyttävät komponenttikansion elementtejä luodakseen sisällön. Näkymiä hallitaan privateRoutes.js-tiedostossa, jossa voidaan asettaa osoite jokaiselle näkymälle. Bislenz-palvelussa voi olla vain yksi näkymä kerrallaan.

Sivulla käytettävät tiedot haetaan backend-palvelimesta tyypillisesti sivun latauksen yhteydessä. Datan haut voidaan tehdä muillakin tavoilla, jos käyttäjän tarvitsee esimerkiksi valita tietolähteitä ennen datahakua. Sivuston latauksen yhteydessä käyttäjät eivät joudu odottelemaan kesken sivun käytön, joka parantaa käyttökokemusta.

Käyttöönotossa syötetyt datalähteet tallennetaan Bislenz-verkkopalvelun tietokantaan. Datalähteiden tietojen avulla Bislenz-palvelin tekee niiden mukaisen haun tietolähteen API-osoitteeseen ja tallentaa saadut tulosrivit tietokantaan. Tulosrivien tallentaminen tietokantaan on pakollinen välivaihe, koska datan hakeminen suoraan verkkosivulle API-osoitteesta vie käyttäjän näkökulmasta liikaa aikaa ja ruuhkauttaa tietolähteiden rajapinnat käyttäjien tekemistä kutsuista. Datahaut vaikuttaisivat myös käyttäjäkokemukseen kielteisesti lisäämällä lataustaukoja keskelle sivuja tai aiheuttamalla yhteysongelmia rajapinnan kanssa, jos yksittäinen haku epäonnistuu huonon yhteyden vuoksi.

Bislenz-verkkopalvelun käyttäjät kirjautuvat järjestelmään Login-näkymästä, jossa asiakas syöttää käyttäjätunnuksen ja salasanan. Onnistuneessa kirjautumisessa palvelin tallentaa palvelimen sessiomuuttujaan asiakastunnisteen, jonka avulla sivusto tunnistaa kirjautuneen käyttäjän sivulla. Tämä prosessi näkyy kuviossa 6. Hyödyntämällä tätä tunnistetta sovelluksen sivut hakevat käyttäjälle tärkeitä tietoja tietokannasta sivulle.



Kuvio 6. Bislenz-verkkopalvelun ja palvelimen vuorovaikutus

Bislenz - verkkopalvelu tekee käyttäjätunnuksen tarkastuksen aina sivun vaihtuessa. Jos käyttäjätunnusta ei löydetä, sovellus ohjaa käyttäjän takaisin kirjautumisnäkymään. Jos palvelun osoiteriviltä löydetään polku väärään näkymään, käyttäjä ohjataan määriteltyyn 404-virhesivulle, jossa käyttäjälle kerrotaan url-osoitteen olevan väärä.

Bislenz-verkkopalvelun sivulla olevassa päävalikossa sivut Executive Summary, Marketing Performance, Customer Journey, Voice of Customer ja Tools ovat kirjoittamishetkellä kaikille käyttäjille yhteisiä välilehtiä. Muut välilehdet ja niiden sisältö vaihtelevat asiakkaan määrittelemien tietolähteiden mukaan.

4 Tekniikoiden esittely

4.1 Frontend-tekniikat

4.1.1 JavaScript

Verkkosivut ovat kehittyneet valtavasti viimeisen kolmen vuosikymmenen aikana ensimmäisten verkkosivujen julkaisemisen jälkeen. Verkossa toimiva sisältö kykenee nykyisin työpöytäsovellusten tasoiseen toiminnallisuuteen. Samalla verkkosivustojen tekemisestä on tullut entistä monimutkaisempi prosessi, jossa kehittäjän on implementoitava moninkertainen määrä toiminnallisuutta.

JavaScript on monipuolinen ohjelmointikieli, jota käytetään tyypillisesti verkkosivujen toiminnallisuuden ohjelmointiin. StackOverFlow-verkkosivuston vuonna 2020 tehdyssä kyselyssä, johon osallistui 65 000 sovelluskehittäjää maailmanlaajuisesti, 67 % äänesti JavaScriptin maailman suosituimmaksi ohjelmointikieleksi. Tässä kyselyssä JavaScript on voittanut kyseisen kategorian nyt kahdeksan kertaa peräkkäin. (StackOverFlow Insights, 2020, Technology)

Mozilla Oy:n kehittäjädokumentaatiossa JavaScriptin suosio selitetään sen dynaamisuudella, jossa se päivittää koodin määrittelemiä arvoja ajon aikana. Tämä ominaisuus on tärkeää HTML-koodissa, jossa käyttäjät antavat sivulle käskyjä lataamisen jälkeen, joita JavaScriptin pitää kuunnella. Jos esimerkiksi verkkosivulla on painike (button), joka hakee sivustolta elementin ja päivittää sen sisältöä klikkaamisen yhteydessä, napin toiminta on määriteltävä JavaScript-ohjelmakoodilla. (Mozilla, 2021)

JavaScript on toiminnaltaan modulaarinen ja se soveltuu jaettavien ominaisuuskirjastojen tekemiseen. Uusi kirjasto määritellään export-komennolla, jonka avulla kirjoi-

tettua ohjelmakoodia voidaan kutsua muissa JavaScript-tiedostoissa import-komennolla. JavaScript-moduulien kanssa on oltava kuitenkin tarkkana. Moduulien huolimaton yhdisteleminen luo yhdistämiskonflikteja, jotka vaikuttavat haitallisesti ohjelmiston turvallisuuteen ja selaimen suorituskykyyn.

Suurimmassa osassa käyttöjärjestelmiä JavaScript päätteellisiä .js-tiedostoja tuetaan oletuksena, joten sen käyttöä varten ei tarvitse asentaa erillisiä kehitysympäristöjä. Niiden sijaan JavaScriptin kanssa käytetään runtime-kirjastoja, joilla verkkosivujen kehittäminen helpottuu.

Hyödyistään huolimatta JavaScriptissä on heikkouksia, jotka vaikeuttavat kielen käyttöä. Suurin osa näistä johtuu kielen dynaamisuudesta ja niiden selvittäminen ei onnistu muuten kuin niistä tietämällä etukäteen.

JavaScriptissä ei ole yhteen tietotyyppiin rajattuja muuttujia (kuten kokonaisluku, liukuluku, merkkijono tai taulukko) vaan muuttujien tietotyyppiä voidaan muuttaa ajon aikana, mikä tapahtuu kuviossa 7.

```
let x; // alustetaan muuttuja x
x = 5;
console.log(x); // palauttaa: 5
x = "hello";
console.log(x); // palauttaa: hello
x = true;
console.log(x); // palauttaa: true
```

Kuvio 7. JavaScript esimerkki: muuttujan arvon muuttuminen

JavaScriptin muuttujissa ei siis ole manuaalista arvon hallintaa, kuten esimerkiksi C-ohjelmointikielissä. JavaScript pyrkii sovittelemaan tämän aiheuttamia virheitä käsittelemällä arvoja ajon aikana. Esimerkiksi, jos ehtolauseessa odotetaan Boolean-arvoa

ja annetaan jokin muu arvo, esimerkiksi teksti ”merkkijono” (Kuvio 8), JavaScript käsittelee sen edelleenkin boolean-arvomuodossa ehdon sisällä.

```
// molemmat ehtolauseet toteutuvat

const x = true;
if (x) {          // true
  console.log(x); // true
}

const y = "merkkijono";
if (y) {          // true
  console.log(y); // merkkijono
}
```

Kuvio 8. JavaScript-esimerkki: Ei-boolanisen muuttujan käsittely

Arvonkäsittelyn säännöissä on kuitenkin muutama poikkeustapaus, joissa muuttuja käsitellään poikkeuksellisesti arvolla epätosi. Nämä ovat:

- false
- 0
- ””, tyhjä merkkijono
- null, asetettu puuttuva arvo
- undefined, asettamaton puuttuva arvo
- NaN (Not a number), arvo kun yritetään laskea ei-numerolla

JavaScript ei myöskään suorita sovellusrivejään järjestyksessä. Jos jokin ohjelmakoodin arvoista palauttaa arvonsa hitaasti, ohjelmakoodi ei odota suorittamisen lopussa palautettavaa arvoa. Tällöin hitaan rivin arvo on suorittamisen jälkeen ei-määritetty eli *undefined*, vaikka kyseinen rivi suoritettiin. Tämä voi aiheuttaa ongelmia mm. tietokantahakujen kanssa, mutta se voidaan kiertää käyttämällä Promise-rakennetta tai async/await -rakenteita, jotka pakottavat JavaScriptin odottamaan funktion arvoa.

Esimerkiksi kuviossa 9 käytetään Promise-rakennetta ja async/await-rakennetta samaan lopputuloksen aikaansaamiseen eli palautetaan viiveellä palautettava lopputulos.

```
new Promise((resolve) => {
  setTimeout(() => {
    resolve("6 seconds have passed");
  }, 6000);
}).then((resolved_value) => {
  console.log(resolved_value);
})
```

```
async function return_after_6_seconds (){
  setTimeout(() => {
    return "6 seconds have passed";
  }, 6000)
}

async function wait_for_slow_value (){
  const x = await return_after_6_seconds()
  return x;
}
```

Kuvio 9. Promise ja async/await rakenteen esimerkki: Odota 6 sekuntia ja palauta

4.1.2 ReactJS

StackOverFlow-verkkosivuston kyselyssä v. 2020 Facebook-konsernin kehittämä React.js on tällä hetkellä yksi maailman johtavista web-käyttöliittymän sovelluskehyksistä. Kyselyssä 36 % vastanneista tietotekniikan ammattilaisista käytti React.js-sovelluskehystä ja tämän lukeman odotetaan kasvavan jatkossa. (StackOverFlow, 2020)

React.js on JavaScript-kirjasto, jolla suunnitellaan ja toteutetaan käyttöliittymiä. Sen tarjoamat komponentit ja komponentin state-tilan hallinnan osaava käyttäjä voi tuottaa monimutkaisia sovelluksia nopeammin ja selkeämmin.

React-komponentit ovat Reactin luomia rakenteita, jotka toimivat React-kirjaston oman lifecycle-metodien mukaan. Komponentit voivat tallentaa efektejä (effect) ja tilamuuttujia (state), joilla ne osaavat mukautua niille tehtäviin muutoksiin. Toisin kuin JavaScript-muuttujat, React-komponentit määritellään isoilla alkukirjaimilla (kuvio 10).

```
const SummaryPage = (props) => {
  const theme = useTheme();
  const classes = useStyles(theme);
  const [data, setData] = useState([]);
  useEffect(() => {
    axios.get(props.API + "/publicData").then((fetchedData) => {
      setData(fetchedData.data);
    });
  }, [props]);
  return (
    <div className={classes.root}>
      <Box padding={2} display="flex" justifyContent="center">
```

Kuvio 10. React-komponentti, tilamuuttuja ja efekti

Tilamuuttujat ovat komponentille asetettuja arvoja, jota päivitetään ajon aikana. Niitä muutetaan komponentissa setState-funktiolla, joka asettaa sille parametrinä annetun arvon ajettaessa. React-sovelluksessa tilamuuttujia tarvitaan komponenteissa aina, jos komponentin pitää muuttaa omia arvojaan ajon aikana. Muuttumattomat muuttujat kannattaa määrittää komponenttiin JavaScript-muuttujiin.

React.js tunnetaan ajon aikaisesta JavaScript-XML-sovelluspohjaan (JSX) pohjautuvasta toiminnallisuudestaan, jolla voidaan ohittaa JavaScriptin esiprosessorin ja AJAX-komentojen käyttäminen sivulla. Näin sivulle voidaan tehdä muutoksia ilman sivun uudelleenlataamista. JSX mahdollistaa sovelluksen muuttujien käyttämisen suoraan HTML-rakenteen kanssa.

React-sovellukset ovat ottaneet viime vuosina uuden, funktionaalisen syntaksin käyttöön, jota ennen komponentit määriteltiin luokkarakenteella. Luokka-komponenteissa efektit korvataan komponentissa määritellyillä lifecycle-metodeilla, joilla säädetään komponentin reaktioita eri tilanteissa. Funktionaalisissa komponenteissa niiden vastaa efekti eli useEffect. Yleisimmät lifecyclet ovat componentDidMount, componentDidUpdate ja componentWillUnmount, jotka aktivoituvat nimensä mukaisissa tilantessa.

- ComponentDidMount aktivoituu komponentin latauksen lopussa.
- CompoentDidUpdate aktivoituu aina komponentin arvojen muuttuessa.
- ComponentWillUnmount aktivoituu komponentin poistuessa näkyvistä.

Kirjoittamishetkellä React tukee sekä funktionaalisia- että luokkakomponentteja ja molemmilla komponenttityypeillä voidaan tuottaa yhtä toimivia ja monipuolisia React-sovelluksia. Molemmat ovat laajasti käytettyjä, joten kehittäjän kannattaa opetella käyttämään molempia tarvittaessa.

React.js-kirjasto ei ole täysin ongelmaton. React-komponentit päivittyvät aina muuttujan arvojen vaihtuessa, mikä aiheuttaa sivulle jatkuvia uudelleenlatauksia. Nämä uudelleenlataukset voivat johtaa yhteentörmäyksiin toisten ohjelmointikirjastojen kanssa.

4.1.3 Node.js-ohjelmakirjastot

JavaScriptin kanssa käytetään nykyisin usein Node.js ajoympäristöä (Web Framework), joka hallinnoi käyttäjien lisäämiä sovelluspaketteja. Sillä voidaan myös määritellä JavaScriptillä määriteltyjä verkkopalvelimia. StackOverFlow-verkkosivun vuonna 2020 tekemän kyselyn vastaajista yli puolet käyttävät Node.js- sovelluskehystä projekteissaan. (StackOverFlow, 2020, Technology)

Node.js-ympäristössä serverinhallinta tehdään myös JavaScript-ohjelmointikielellä, jolloin vältytään muiden kielien vaatimuksilta (esim. PHP). Node.js on myös asynkroninen, joten sille tehdyt haut eivät vaadi palvelimelta muissa palvelimissa esiintyvää odottamista.

Bislenz-verkkosovelluksessa Node.js-serverinhallintaa käytetään sekä sovelluksen että palvelimen puolella. Käyttöliittymän puolella Node-järjestelmää käytetään npm-pakettien asentamiseen sekä sovelluksen rakentamiseen tallennuksen yhteydessä paikallisella localhost-palvelimella. Näin ohjelmakoodille tehdyt muutokset nähdään suoraan omalta selaimelta, mikä nopeuttaa kehitystä.

Kuten kaikki Runtime-ohjelmistot, Node-järjestelmän käynnistämät palvelimet toimivat paikallisesti käyttöjärjestelmässä. Tämän vuoksi se on asennettava koneelle ennen käyttöä, jotta paikallisten palvelimien käynnistäminen onnistuu. Uusin versio Node.js-runtimesta löytyy osoitteesta nodejs.org. Kirjoittamishetkellä Node.js vaatii järjestelmän admin-oikeuksia lisätäkseen tiedostosijaintinsa käyttöjärjestelmän ympäristömuuttujiin.

Node.js-ohjelmiston NPM- ja NPX-komentoja voidaan ajaa järjestelmän komentopaneelista asennuksen jälkeen. NPM-komennoilla voidaan käynnistää ja sulkea paikallinen verkkopalvelin, rakentaa sovellus verkkosivuksi julkaisua varten tai asentaa npm-kirjastopaketteja. Toimiakseen NPM tarvitsee paikallisesti tallennetun package.json-taulukon ja node-modules-kansion, joka sisältää kaikki node-kirjastot. NPX-komennoilla voidaan ajaa node-komentoja ilman npm:n tarvitsemia paketteja. Node-järjestelmässä on monia komentoja, joista yleisimmät komennot löytyvät taulukosta 3.

Taulukko 3: Node-ohjelmiston yleisimmät komennot

Komento	**Komennon Kuvaus**
node	Käynnistä node-terminaali komentolinjalle.
npm start	Käynnistä nykyisessä kansiossa oleva node-sovellus paikallisessa localhost-palvelimessa
npm run build	Rakenna sivu build-kansioon, joka sisältää julkaistavan verkkosivun ilman node-järjestelmän työkaluja.
npm install	Asenna kaikki package.json-tiedostossa merkatut tiedostot
npm install kirjasto	Asenna kirjasto ja lisää se package.json-kansioon.
npx create-react-app sovellus	Luo kansio ”sovellus” ja sisällytä siihen kaikki React-sovellukseen vaadittavat paketit, package.json sekä html ja css-tiedostot.

Node ohjelmistossa käytetään Node-kirjastoja, joiden avulla sivustolle on kerätty julkisesta lähteistä kerättyjä toimintoja,kuten:

- Kuvaajia ja malleja (FusionCharts ja Recharts)
- Sivulla käytettäviä graafisia elementtejä, kuten nappeja, listoja, taulukoita ja layoutin apuvälineitä (Material UI)
- Tietokannan hakumenetelmiä (axios)
- Päivämäärien laskemisen, vertailun ja muotoilun apuvälineitä (moment)
- Aikavälien valitsemiseen käytettäviä komponentti (Datepicker)
- Kuvankaappauksien kerääminen (screenshot)

Sovelluksen komponenteista axios, moment sekä Material UI ovat käytetyimpiä sovelluksessa. Sovelluksen jokaisella sivulla on axiosin toimintaa vaativia datahakuja ja moment on käytössä lähes aina päiväyksien muotoilemisessa ja vertaamisessa.Näkymien käyttöliittymässä käytetään Material UI-kirjaston valmiita elementtejä, joilla on valmiit määritykset teeman, kontrastin ja animaation osalta.

Node-kirjaston avulla ladatut paketit nopeuttavat kehitystä, mutta niiden lisääminen kasvattaa nodella rakennetun sivuston kokoa. Jos sivustolla luodun paketin koko kasvaa tarpeeksi suureksi, sen rakentaminen ”npm build” -komennolla vaikeutuu. Tämä johtuu Node-ympäristölle määritettyjen muistirajoitusten ylittyessä. Esimerkiksi React-sovellukset paketoidaan Node-järjestelmässä vain yhteen valtavaan tiedostoon, joka ei jaottele sitä oletuksena alempiin sovelluspaketteihin. Tämä ongelma voidaan ohittaa tai ratkaista joko:

1. Poistamalla tarpeettomia kirjastoja package.json-tiedostosta.
2. Kasvattamalla node-järjestelmän muistirajoja ennen sivuston rakentamista esim. komennolla: NODE_OPTIONS=--max_old_space_size=4096.
3. Jakamalla kokonaisuus erikseen ladattaviin kokonaisuuksiin käyttämällä kehityskirjastolle ominaisia työkaluja.

4.2 Backend-tekniikat

4.2.1 Palvelin

Bislenz-verkkopalvelun palvelimena käytetään Linux-virtuaalikonetta, jolle on asennettu Node.js-palvelin sekä MySQL-tietokanta. Palvelimen Node-järjestelmä seuraa samaa perusrakennetta kuin sivustolla. Eroavaisuutena palvelimen node-järjestelmää käytetään palvelimella julkisesti internetissä, josta sille voidaan tehdä hakuja käyttämälle sille määritettyä osoitetta. Palvelimella käytetään myös eri node-kirjastoja, kuten pm2, cron ja express -kirjastoja

Pääosa palvelimesta koostui ajettavista JavaScript-tiedostoja, joilla suoritettiin tietohakuja kohteisiin ja tämä haettu data käytettiin tiedostossa. Nämä tiedostot olivat joko DataUpdate-tiedostoja, joilla haettua dataa tallennettiin tietokantaan tai route-tiedostoja, joilla määriteltiin uusia tiedonhakuja sivuston puolelle. Nämä tiedostot ajettiin joko app.js-tiedostosta cron-kirjastolla ajastettuna, manuaalisesti test-reitin avulla tai verkkosivun käyttöliittymän kutsusta hakemalla nimetyn reitin API-osoitetta.

Palvelimelle lisätyllä Express-kirjastolla määritettiin osoitteeseen lisättäviä polkuja. Näiden hakupolkujen (routes) avulla sovelluksessa haettiin tietoa palvelimelta.

Hakupolussa määriteltiin uniikki merkkijono, joka lisätään sovelluksessa käytettyyn API-osoitteeseen. Tällä osoitteella tehty haku käynnistää palvelimessa määritellyn ohjelmakoodin, jossa voidaan käyttää sessiomuuttujia, käyttäjän client-tietoja sekä muuta arkaluontoista dataa. Jos haun aikana tarvitaan sivustosta mukana lähetettäviä lisätietoja, nämä voidaan liittää haun mukana header, tai params-muuttujina. Pal-

velimen hakupolkujen määrää ei ole rajoitettu, mutta jokaisella täytyy olla uniikit nimet, ettei tapahdu nimien yhteentörmäystä. Tässä tapauksessa järjestelmä antaa virhekoodin ja hylkää haun.

Jos esimerkiksi näkymässä lähetetään haku osoitteeseen ”*backendin_osoite/apin_nimi/ReviewData*”, osoitteeseen lisätään params-objektiin haettavan tiedon aloitus- ja lopetuspäivämäärät startDate ja endDate. Kun haku on suoritettu, sen tulokset palautetaan selaimelle then-osioon, kuten kuviossa 11.

```
axios.get(props.API + "/ReviewData", {
  params: {
    startDate: moment(reviewdates.startDate).format("YYYY-MM-DD"),
    endDate: moment(reviewdates.endDate).format("YYYY-MM-DD"),
  },
})
.then((reviews)=>{
  setReviewData(
    reviews.data.filter((e) => e.datasource === "googlemybusiness")
  );
})
```

Kuvio 11. Sovelluksen ReviewData-tiedonhakulauseke palvelimelle

Palvelimen express-tunnistaa haun ja ajaa palvelimelta osoitetta vastaavan reitin, joka näkyy kuviossa 12. Reitissä voidaan käyttää sessiomuuttujia (req.session) ja haun params-kohdan muuttujia (req.query). Tulokset palautetaan haun tekijälle palauttamalla JSON- vastauslähetys (res.json).

```
router.get("/ReviewData", cors(tila), (req, res) => {
    const sql =
        req.query.startDate && req.query.endDate
            ? `SELECT * FROM reviews where client_id = ? AND date >= ? AND date <= ?`
            : `SELECT * FROM reviews where client_id = ?`;
    db.query(
        sql,
        req.query.startDate && req.query.endDate
            ? [req.session.client_id, req.query.startDate, req.query.endDate]
            : [req.session.client_id],
        (err, result) => {
            if (err) throw err;
            res.json(result);
        }
    );
});
```

Kuvio 12. Tiedonhakulausekkeen reitti palvelimella

Palvelimella suoritetaan myös Bislenz-järjestelmän tietovarastojen päivittäminen dataupdate-nimisten JavaScript-tiedostojen avulla, joissa haetaan tiedoston nimen mukainen data tietolähteen omasta rajapinnasta. Esimerkiksi kuviossa 13 on Twitter-datan haku SME-tietokantataulusta. Tämä data syötetään Bislenz-tietokantaan uusina datariveinä. Tiedosto kirjoitetaan node-moduulien muotoon, jossa suoritettava ohjelmakoodi asetetaan module.exports -rakenteen sisään. Tällä tavalla määritelty moduuli tuodaan toiseen tiedostoon require-lausekkeella.

```
// app.js
const TwitterMonthlyFollowers = require("./dataupdates-uusi/twitter_public_data/monthly_followers");
// call insert & update data on the last day of month 23:00h
cron.schedule("0 23 1 * *", () => {
  TwitterMonthlyFollowers();
});

// twittermonthlyfollowers.js
const axios = require("axios");
const db = require("../../config/database");
const moment = require("moment");
module.exports = () => {
  const somelista = [];
  db.query(
    "SELECT * FROM `SME_feeds` WHERE `client_id` IS NOT NULL AND `source` LIKE 'twitter'",
    (err, result) => {···
    }
  );
};
```

Kuvio 13. Twitter-datahaku ja sen ajoittaminen cron-kirjastolla

Kun tietokannan päivitys määritellään App.js-tiedostossa, käytetään Cron-kirjastoa, jolla voidaan aikatauluttaa sovelluksessa tehtäviä datapäivityksiä. Ajastukselle voidaan määrittää mikä vain haluttu ajanjakso. Esimerkiksi kuviossa 14 datarivit päivitetään joka yö klo 4:00 paikallista suomen aikaa (+2:00). Bislenz- verkkopalvelun datapäivitykset käynnistetään yleensä joka yö samaan aikaan, kerran viikossa tai kerran kuukaudessa, mutta sovellukseen voidaan lisätä tarvittaessa muitakin ajanjaksoja.

```
const cron = require("node-cron");
// call insert & update data on 4:00h on local finnish time
cron.schedule(
  "0 4 * * *",
  () => {
    activeForecast();
  },
  {
    scheduled: true,
    timezone: "Europe/Helsinki",
  }
);
```

Kuvio 14. Funktion käynnistäminen Bislenz-palvelimella node-cron-kirjastoa käyttäen.

Bislenz-palvelimelle tehtävät muutokset suoritetaan ottamalla etäyhteys palvelimeen ja muokkaamalla palvelimella olevia tiedostoja paikallisessa editorissa. Palvelimelle on määritetty asetukset, joilla tallentaminen ei käynnistä palvelinta uudestaan, joten se pitää tehdä manuaalisesti muutoksien tallentamisen jälkeen. Uudelleenkäynnistys tapahtuu käyttämällä komentoa: pm2 start 0.

4.2.2 MySQL

MySQL on vuonna 1996 tehty relaatiotietokantajärjestelmä, jota käytetään valtaosassa websovelluksia ja monet uudet tietokantajärjestelmät pohjatuvat siihen tavalla tai toisella. MySQL:n dokumentaatiosivulla mm. Google, Facebook ja Zappos listataan MySQL:ää käyttäviksi sovelluksiksi. (Oracle, 2021)

StackOverFlow-verkkosivuston vuoden 2020 tutkimuksessa 55.6 % vastasi MySQL:n olevan yleisin tietokantajärjestelmä verkkosovelluksille. Tutkimukseen otti osaa 65 000 ammatillista sovelluskehittäjää. (StackOverFlow, 2020)

MySQL:ssä tietokanta (database) jaotellaan tietokantatauluihin (table) ja niiden sisältämiin riveihin (row). Taulukon sarakkeet (column) määrittävät riveille tietyntyyppisiä soluja (cell), jotka rivien kuuluu täyttää sarakkeessa merkityillä arvoilla. Tietokantaan kuuluvien taulujen ja rivien määrää ei ole rajoitettu, mutta ne pyritään minimissä tallennustilan vuoksi. MySQL-tauluihin voidaan asettaa rajoituksia tietyn sarakkeiden

arvoihin. Jokaisessa taulussa on yleensä uniikki taulu, jonka arvo on oltava erilainen kaikista taulun riveissä.

MySQL-tietokannassa voidaan hakea taulukon tietoja, syöttää rivejä ja määrittää yhteyksiä taulukoiden välille. Sovelluskäytössä tyypillisimmät komennot ovat tietokantarivien rivien hakeminen (kuviossa 15) ja syöttäminen (kuviossa 16). Bislenz-palvelimella suoritetut haut suoritetaan käyttämällä erillistä tietokantaobjektia, jossa määritellään tietokannan vaatimat käyttäjätiedot ja osoitteet.

```
db.query(
    "SELECT * FROM `SME_feeds` WHERE `source` = 'facebook' AND `groupName` IS NOT NULL",
    (err, result) => {
        if (err) throw err;
        groupData(result).forEach((page) => {
            somelista.push([page.name, page.fb]);
        });
```

Kuvio 15. MySQL-tietokantahaku Bislenz- verkkopalvelussa

```
        const FBdata = array[i];
        const FBsql =
            "INSERT INTO social_media_listing SET ? ON DUPLICATE KEY UPDATE id=id, likes = " +
            FBdata.likes +
            ", posts = " +
            FBdata.posts +
            ", comments = " +
            FBdata.comments +
            ", shares = " +
            FBdata.shares;
        db.query(FBsql, FBdata, (err, result) => {
            if (err) {
                console.error(err);
            } else {
                console.log(result);
            }
        });
```

Kuvio 16. MySQL-tietokantarivien syöttäminen Bislenz- verkkopalvelussa

4.2.3 R-ohjelmointikieli

R-ohjelmointikieli on S-ohjelmointikielien sukuun kuuluva asynkroninen ohjelmointikieli, jota hyödynnetään tyypillisesti datan analysointiin ja generointiin sen mate-

maattisen laskentatehon vuoksi. R-ohjelmakoodia ylläpitävän R-Project-sivun mukaan yksi R:n vahvuuksista on helppous tuottaa julkaisuvalmiita kuvaajia ja datakäyriä sekä tarvittaessa niiden matemaattisia symboleita ja yhtälöitä. Käyrien hallinnassa oletusasetukset on hyvin suunniteltu ja käyttäjällä on täysi hallinta. (R-Project, 2021)

R-ohjelmakoodia ei tueta oletuksena käyttöjärjestelmissä, joten sen suorittamiseen tarvitaan R-kehitysympäristö. Tämä löytyy R:n verkkosivuilta, jonka jälkeen asennus suoritetaan ohjeiden mukaan. Asennuksen yhteydessä määriteltävät järjestelmämuuttujat kannattaa tarkistaa, jotta R-tiedostot tunnistetaan projektikansiossa.

R-kehitysympäristössä käytetään CRON-ohjelmakirjastoja, jotka sisältävät mallinnusta ja laskentaa helpottavia funktiorakenteita. Kirjastot ovat vapaasti ladattavissa install.packages-komennolla, jonka jälkeen niitä voidaan käyttää kaikkialla generoinnin aikana. Ennustemallille tärkeimmät kirjastot ovat taulukkojen muokkaamiseen, aikajaksojen määrittelyyn sekä automaattiseen ARIMA-ennustemalliin käytettävät funktiot. (Mt.)

R-ohjelmakoodi tallennetaan ”.R”-tiedostoihin, joita käytetään R-Script-sovittamiskirjaston avulla Bislenzin verkkopalvelimella ennustemallin generoimisessa lähdedatasta. Kirjasto ottaa verkkosovelluksesta annetun datan ja vie sen parametrina R-tiedostoon, jonka tekemien muutosten jälkeen data palautetaan takaisin sovellukseen (kuvio 17). R-ohjelmakoodia käytetään tavallisesti prosessoritehokkaissa koneissa, joten palvelinkäytössä sen käyttö kannattaa minimoida mahdollisimman pitkälle.

```
R("./forecast/daily/Daily/main.R")
    .data(generationSourceData, settings)
    .call((err, result) => {
        if (err) throw err;
        const generatedForecast = result.data;
```

Kuvio 17. R-ohjelmakoodin ajaminen Node-palvelimella käyttäen R-Script-kirjastoa

5 Ennustemallin toteuttaminen

5.1 Testidatan ajaminen ja mallin konfigurointi

Ennustemallin toteutusta varten kerättiin työn alkuvaiheesta tietoa sen käytöstä ja ajamisesta, jotta sen käyttöönottoon voitaisiin määritellä oikea toimintatapa.

Ennustemalli oli työn alkuvaiheessa ajettu palvelimella onnistuneesti, joten ensimmäisenä selvitettiin sen generointinopeus testidatan avulla. Testit suoritettiin Bislenz-palvelimella, jotta saadut lukemat vastaavat sovelluksen generointilukemia, sillä paikallisesti ajettuna ennustemallin ajaminen käy paljon nopeammin kuin palvelimella.

Palvelimen express-reittejä muokattiin, jotta ajaminen onnistui manuaalisesti testien aikana. Palvelimelle määritetyt forecast- ja forecastSource-tiedostot lisättiin palvelimen test-hakupolkuun (kuviossa 18), joka ajetaan menemällä sen määrittämään osoitteeseen www.tietokannan_nimi.fi/test2. Route-reitin sisällä voidaan kommentoida kutsu //-merkillä, jolloin sitä riviä ei ajeta.

```
//Data forecasts
const forecast = require("./forecast");
const forecastSource = require("./dataupdates-uusi/forecast/forecast_source");

//localhost:5000/test to run functions manually
app.get("/test", (req, res) => {
  // forecastSource();
  forecast();
});
```

Kuvio 18. Forecast-tiedoston ajaminen palvelimen "/test"-hakureitissä

Forecast-source-tiedosto luo ennustemallin generointiin tarvittavan testidatan. Se hakee testirivejä tietokannasta ja tallentaa sen sisällön Bislenz-tietokannan ”forecast-source”-MySQL-tauluun. Haetut tiedot ovat toimivaa testidataa, jotka eivät kuulu asiakkaaille.

Forecast-tiedostossa forecast-source- tietokantataulun rivit haetaan tiedostona ja syötetään ARIMA-ennustemallille, joka generoi ennusteen datariveinä. Tämä data tallennetaan ”forecast-result”-tietokantatauluun, josta sitä voidaan käyttää verkkosivulla.

Ennustemallille määritellään tiedostossa generointiparametrit p, d ja q, jotka säätelevät autoregressiota, liikkuvaa keskiarvoa sekä differointia. Näiden arvojen minimiarvo on 0, ovat kokonaislukuja ja niiden arvoja kasvatettiin tasaisesti testauksen aikana.

Jotta parametrien mukaisia generointiaikoja voitiin vertailla, forecast-tiedostoon asetettin ajastimet. Ensin asetettiin console.time-funktio ennen ennustemallin generointia ja sen jälkeen console.timeEnd-funktio. Console.time käynnistää palvelun käyttämän sekuntikellon ja console.timeEnd pysäyttää sen. Tätä rakennetta käyttämällä saatiin ennustemallin generointiaika millisekunteina, jonka mitattiin eri parametriarvoilla.

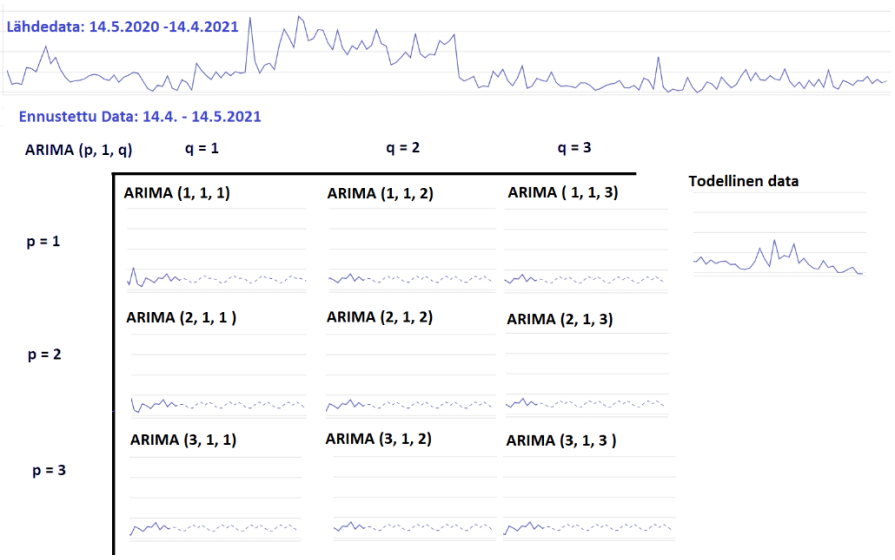
Generointiajat näytetään taulukossa 4. Palvelimen suoritustehon vuoksi parametreja ei asetettu arvoa 3 suuremmaksi. Integrointilukemaa ei muutettu mittauksen aikana, koska syötetty data ei missään vaiheessa vaadi moninkertaista differointia.

Taulukko 4. Työssä käytetyn ARIMA(p, d, q)-ennustemallin suoritusajat sekunteina

Ajettava malli ja parametrit	Generointiaika: Lyhyin	Generointiaika: Pisin
ARIMA (1,1,1)	190 000 ms	436 596 ms
ARIMA (2,1,2)	339 007 ms	442 256 ms
ARIMA (3,1,3)	416 324 ms	522 089 ms

Vertailutulosten perusteella ARIMA-mallin suoritusaika todettiin liian hitaaksi selaimen käyttöön, sillä useamman minuutin latausaika oli liikaa verkkosovelluksen asiakaspalvelustandardeille. Generointikoodin muuttamista pohdittiin, mutta lopulta tämä todettiin turhan suureksi tehtäväksi projektin aikana, joten datan generointi siirrettiin sivuston suorittamisesta palvelimelle tietovaraston päivityksen yhteyteen.

Vertailutulosten nopeuden lisäksi työssä selvitettiin generoitujen rivien sopivuus. Tämä selvitettiin generoimalla tulosdata jokaisella parametriyhdistelmällä ja vertailemalla tuloksia. Kaikissa generoiduissa datassa käytettiin samaa vuoden takaista lähdedataa, josta jätettiin pois viimeiset 30 päivää. Näiden päivien puuttuvat arvot generoitiin ennustemallin avulla ja asetettiin kuvaajaan katkoviivalla. Tulokset generoidusta datasta näkyvät kuviossa 19.



Kuvio 19. Ennustemallin generoima data eri lähtöparametreilla

Kuvion tulosten mukaan ARIMA-ennustemallin generoimat datarivit tuottivat lukemia, jotka vaihtelivat vähän. Tämä johtui R-ohjelmakoodissa käytettävästä auto-arima-metodista, joka tasapainottaa arvoja datalle tehtyjen automaattisilla testeillä. Testien avulla ennustemallin aikana outlier-arvot, eli poikkeusarvot sekä null, eli

puuttuvat arvot huomioidaan automaattisesti algoritmissa. Nämä testit voidaan poistaa käytöstä muuttamalla ennustemallin parametrejä, jolloin ennusteiden vaihteluväli kasvaa, mutta silloin outlier- ja null-arvot joudutaan käsittelemään manuaalisesti. Tämän vuoksi testit jätettiin päälle, jotta rajatilanteita ei tarvitse käsitellä erikseen.

Generoitu data ei näyttänyt merkittävää muutosta ARIMA (2,1,2) - mallia suuremmilla arvoilla. Jos otimme huomioon taulukon 1 generointiajat ja kuvion 18 tulosdatan tarkkuuden, ennustemalli toimii parhaiten parametreilla ARIMA (2, 1, 2). Kaikki palvelimella toteutetut ennusteet generoitiin jatkossa käyttämällä ennustemallissa näitä parametreja.

5.2 Datan käsittely ja tallentaminen

Ennustemallin käyttöä varten palvelimelle tehtiin muutoksia ennustemallin tietojen keräämiseen ja tallentamiseen. Ensiksi suoritettiin tarvittavat muutokset tietokantataululle, jotta käyttäjien valitsemat ennustemallin määritelmät saadaan tallennettua. Näiden mukaan ennustemalli voitiin generoida näiden määritelmien mukaan, josta saatu data tallennettiin tietokantaan asiakkaan käytettäväksi.

Ennustemallin käyttämä data on oltava säännöllistä. Se on generoitava jaksoittain, esimerkiksi päivittäin tai viikoittain tilanteesta riippuen. Bislenz-sovelluksessa ennustemalli on suunniteltu niin, että se voi ottaa minkä tahansa siihen soveltuvan datalähteen käsittelyyn. Tämän ominaisuuden toteuttamiseen tietokantatauluun piti tehdä muutoksia.

Ennustemallin generointiin tarvittava data haettiin Bislenz-tietokannasta, jonka tietolähteenä käytettiin API-rajapintapalveluista haettua dataa. Tämä data päivitettiin säännöllisesti palvelimella olevilla dataupdate-tiedostoilla.

Työn alkuvaiheessa ennustemallia varten oli määritelty kaksi MySQL-tietokantataulua Bislenz-Tietokantaan:

- Ennusteen lähdedata (forecast-source), josta ennustemallin generoima data haettiin. Taulukko päivitettiin datapäivitys-tiedostolla, joka ajettiin testin aikana manuaalisesti.
- Ennusteen tulokset, (forecast-results), jonne ennustemallin generoimat ennusterivit tallennettiin.

Ennustemalliin lisättiin kolmas tietokantataulu toteutusta varten: forecast-queue. Tähän taulukkoon tallennetaan käyttäjän määrittelemät ennusteet datariveinä, jotka sisältävät kaikki tarvittavat tiedot ennustemallin generointiin. Rivit sisältävät datalähteen, käyttäjän tunnisteen, haettavan aikavälin ja ennusterivien määrän. Palvelimella toteutettavassa tiedostossa ennustemalli generoidaan jokaiselle tämän tietokantataulun riville ja aikaansaadut rivit tallennetaan forecast-results-tauluun. Tämä prosessi suoritetaan säännöllisin aikavälein, jotta ennuste pysyy ajankohtaisena.

Datapäivityksen aikana palvelin hidastuu, joka vaikuttaa asiakaskokemukseen. Tämän vuoksi päivitys ajastettiin yölle klo 2:00, jolloin verkkopalvelun ruuhka oli pienimmillään. Ennusteiden päivittämiseen varattu forecastUpdate-tiedosto haetaan App.js-tiedostossa require-funktiolla ja asetetaan cron-ajastuksen sisään. Tämä prosessi kuviossa 20.

```
const forecastUpdate = require("./dataupdates-uusi/forecast/forecast_update");
// call insert & update data on 2:00h
cron.schedule("0 2 * * *", () => {
  forecastUpdate()
});
```

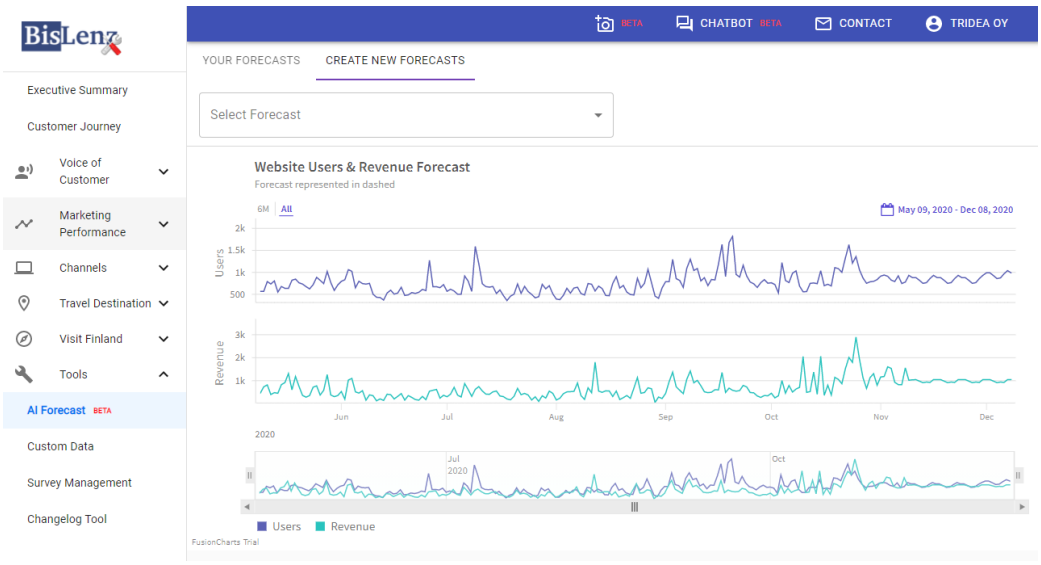
Kuvio 20. Forecast-update-funktion ajastus Bislenzin palvelimella käyttämällä cron-kirjastoa.

Kun tietokantaan tehdyt muutokset saatiin valmiiksi, keskityttiin itse datan käynnistävään tiedostoon. Bislenz-palvelimelle määriteltiin forecastUpdate-tiedosto forecast-kansioon ja sen sisällä suoritetaan seuraavat operaatiot järjestyksessä:

1. Haetaan forecast-queries-tietokantataulun kaikki rivit, jotka tallennetaan JSON-taulukkoon.
2. Jokaisella saadulla objektilla luodaan tietokantahaku Bislenz-tietokantaan, josta saadaan lähdedatat jokaiselle ennustemallille.
3. Tähän dataan lisätään objektin mukaisesti datarivejä, jotka ovat tyhjiä. Tämä data syötetään ennustemallille, joka generoi ennusteen.
4. Tästä datasta poistetaan lähdedata, jotta niillä ei ruuhkauteta tietokantaa. Ennustetut datarivit tallennetaan forecast-results-tietokantataulun riveiksi.

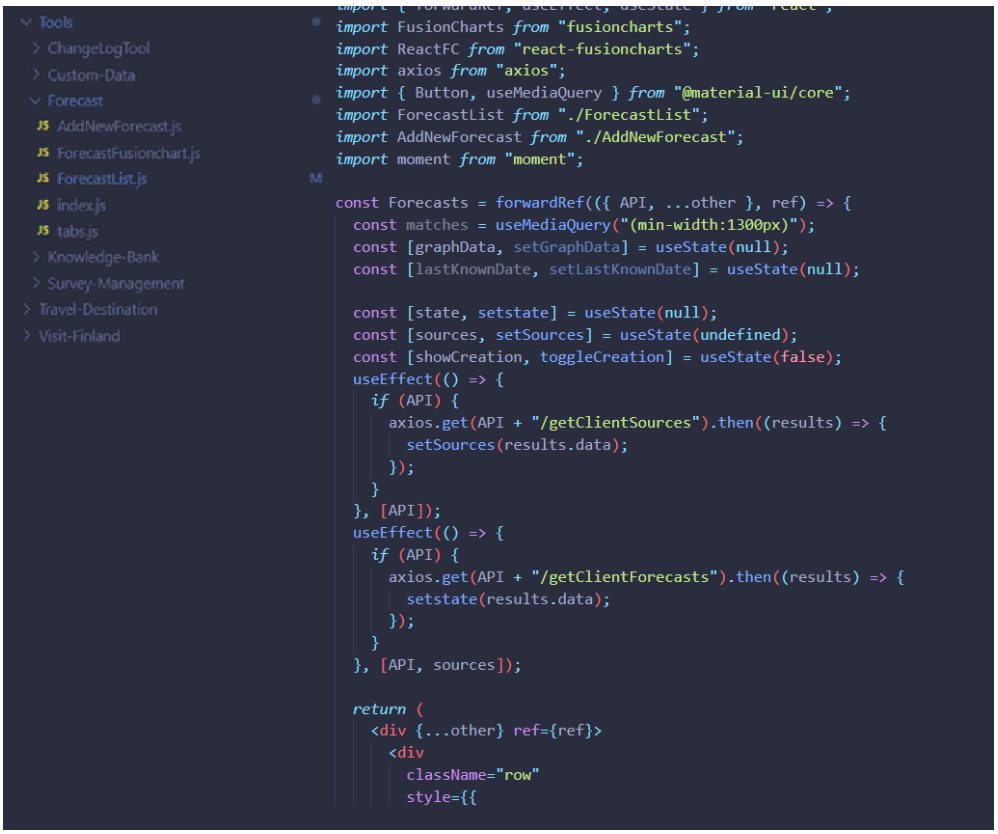
5.3 Tulosdatan visualisointi

Tutkimuksen alussa määriteltiin, että sivuston on toimittava määritellyssä Bislenz-palvelussa, joten verkkopalvelun valmista AI-Forecast-näkymää muokattiin sopivaksi ennustemallin käyttöön. Tässä näkymässä oli alkuvaiheessa yhteenveto luoduista ennusteista kuvion 21 mukaisesti, mutta ennustemallien luomiselle ei ollut vielä omaa näkymää sivulla. Ennustemallin generointi toteutettiin manuaalisesti sivuston ulkopuolella.



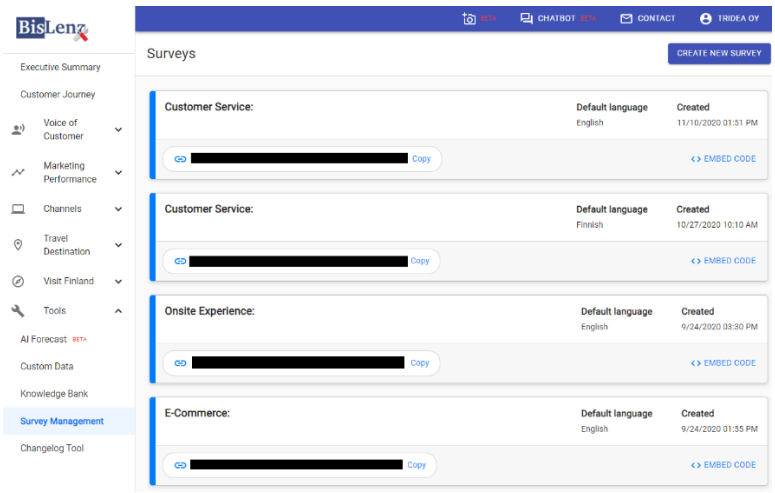
Kuvio 21. Bislenz-sivuston AI Forecast-näkymä

Sivuston käyttöliittymää muokattiin ensimmäiseksi. Ennusteelle varatulle Forecast-kansioon lisättiin uudet JavaScript-tiedostot sivustolla käytettäviin kokonaisuuksiin, jotka toteutettiin React-komponenteilla. Näkymän pääsivu on index.js-tiedoston Forecasts-komponentti (kuviossa 22).

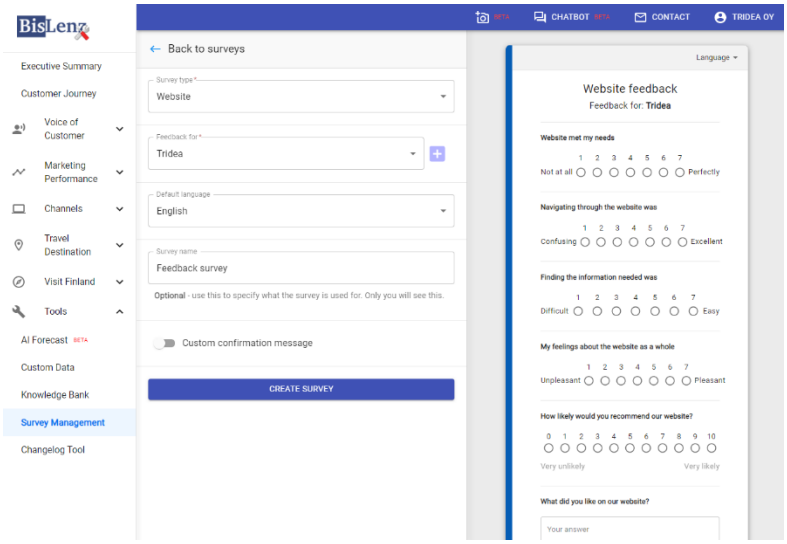


Kuvio 22. Forecastkansion rakenne ja siihen luotu index.js – tiedosto, jossa on Forecasts-niminen React-komponentti

Muokatussa näkymässä listataan kaikki sen hetken aktiiviset ennusteet, oli niissä dataa tai ei. Sivun käyttöliittymän pohjana käytetään Bislenz-palvelun Survey Management -näkymää (kuvio 23), jossa toteutettiin samanlainen kokonaisuus ja sen kyselylomaketta (kuvio 24), jolla asiakas määritteli uuden kyselyn.



Kuvio 23. Active Forecasts-sivun mallina käytettävä Survey management-sivu



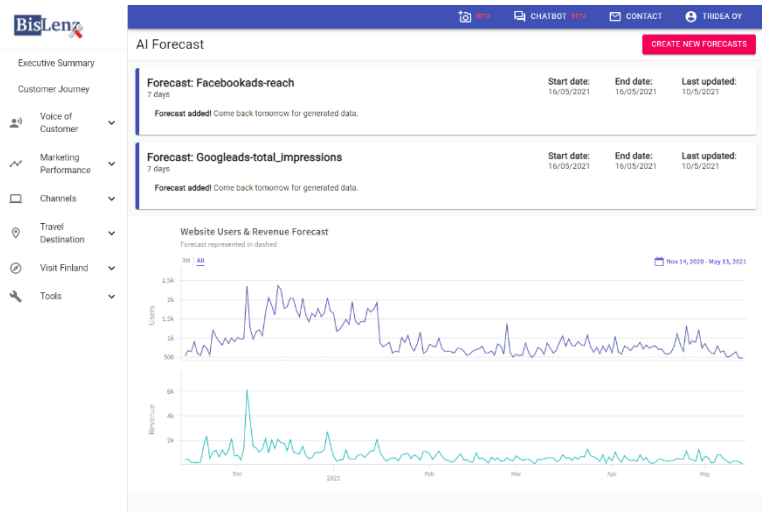
Kuvio 24. Create New Forecast -näkymän mallina käytettävä Create New Survey-näkymä

Verkkosivun käyttöliittymä suunniteltiin pääosassa komponenttikansion elementeillä. Tarvittavat komponentit olivat:

- Gridpage, jolla luotiin näytön leveyden mukaan muuttuva layout, jotta sivusto toimii kaiken kokoisilla näytöillä
- LineApexChart, jolla luodaan sivulle tarvittavat kuvaajat
- ModalDatepicker, jolla lähdedatan alku- ja lopetuspäivät valitaan kalenterin mukaan.

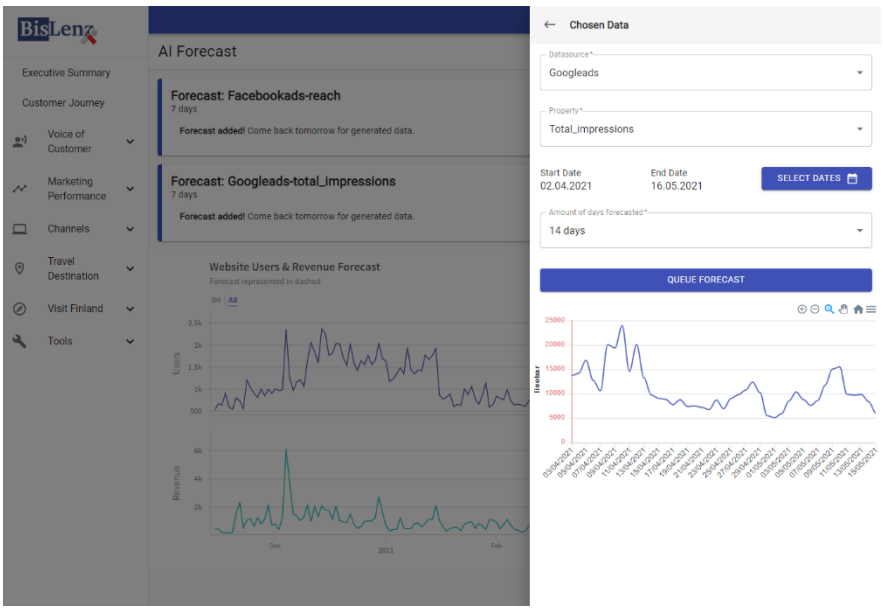
Ulkoasussa käytettiin Bislenz-sovelluksessa paljon käytettyä Material UI -kirjastoa, jolla sivustolle saatiin yhtenäinen ulkonäkö muiden näkymien kanssa. Material UI -kirjastoa käytetään painikkeiden ja tekstivalikkojen tekemiseen.

Näkymässä listataan asiakkaan ennusteet vaakasuuntaisina segmentteinä, joissa listataan tekstinä ennusteen tiedot ja ennustedatan mukaan tehty viivadiagrammi kuvion 25 mukaan. AI Forecast -otsikon oikealla puolella esitellään Create New Forecasts -painike, jota painamalla käyttäjä pääsee luomaan uuden ennusteen.

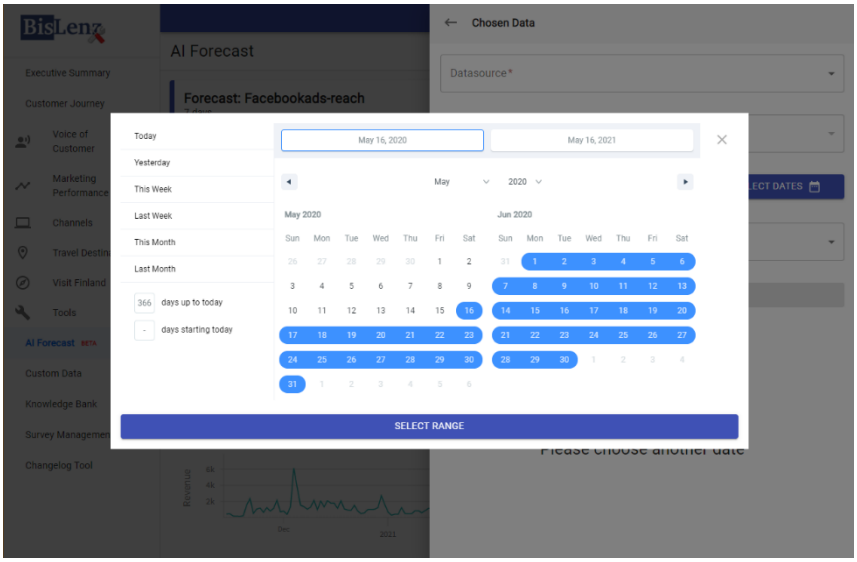


Kuvio 25. Työn aikana toteutettu Ai Forecast -näkymä

Nappia painaessa tietolomake liukuu sivun reunasta Material UI-kirjaston Drawer-elementin avulla päänäkymän päälle. Lomakkeessa käyttäjä pystyy valitsemaan ennusteessa käytettävän tietolähteen ennustettavien rivien määrän kuvion 26 mukaisesti sekä ennusteen generointiin käytettävän ajanjakson kuvion 27 mukaisesti. Datalähdettä vaihtaessa valikossa näkyvä viivadiagrammi muuttuu valintojen mukaisesti.

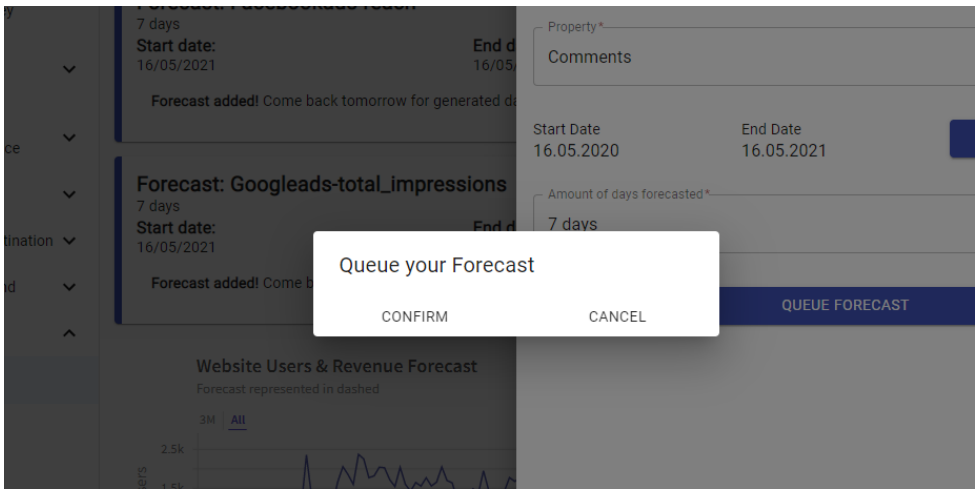


Kuvio 26. Uuden ennusteen luominen uudessa Ai Forecast-näkymässä



Kuvio 27. Päivämäärävalinta uuden ennusteen määrittämisessä

Valikon lopussa oleva Queue forecast -painike tulee aktiivisesti vasta kun kaikki pakolliset kentät on täytetty. Painikkeesta avataan erillinen hyväksymisikkuna, josta käyttäjä hyväksyy tekemänsä ennusteen (kuvio 27).



Kuvio 28. Määritetyn ennusteen hyväksymisvalikko

Kun sivuston ulkonäkö saatiin toimimaan, keskityttiin tarkemmin sivuston datasisällön hakemiseen. Ennusteet ja datalähteet haetaan näkymän latautuessa sivulle käyttämällä React-sovelluksen efektejä.

Sivuston tiedonhakulauseet asetetaan React-efektirakenteiden (useEffect) sisään, koska ne aktivoidaan sivun latauksen latauksessa ja muuttujien päivityksen aikana. Näin sivusto voi suorittaa uuden haun käyttäjän muuttaessa valintojaan.

Efektin sisällä itse haku suoritetaan käyttämällä axios-kirjastoa, jolla suoritetaan haku tietokantapalvelimen osoitteeseen. Sille annetaan osoite, jonka avulla sivusto suorittaa haun palvelimelle ja palauttaa sen tulokset käyttöliittymään. Tällä tavalla tietokannasta voidaan hakea tietoja ilman tietokantatunnusten antamista selaimelle. Jos hakuun tarvitaan arvoja käyttöliittymästä, ne lisätään axios-hakuun params-objektina.

Tietokantapalvelimella ei alkuvaiheessa ollut sivulla tarvittavia hakupolkuja, joten ne lisättiin palvelimelle toteutuksen aikana. Työn aikana toteutettiin taulukon 5 mukaiset käyttöliittymässä käytettävät hakupolut. Hakujen lopuksi data asetettiin sivun state-muuttujiin, jotka muuttivat sivun nykyisen mukaiseksi.

Taulukko 5. AI forecast -sivua varten tehdyt hakupolut

Hakupolku (lisätään API-osoitteeseen)	**Hakuun liitettävät parametrit**	**Palautettu data / tulos**
/getClientSources	-	Kirjautuneen käyttäjän kaikki datalähteet.
/getClientForecasts	-	Kirjautuneen käyttäjän kaikki luodut ennusteet.
/fetchDataBySource	Datalähteen nimi (datasource) Sarakkeen nimi (fetched_ property) Päivämääräsarakkeen nimi (dateprop)	Lähteestä haettu data. Sisältää päiväyksen ja halutun sarakkeen lukemat.
/createNewForecast	Create new forecast -lomakkeeseen syötetyt kentät objektissa: Datalähteen nimi (datasource) Sarakkeen nimi (fetched_ property) Päivämääräsarakkeen nimi (dateprop) Valittavan datan aloituspäivä (start_date) ja lopetuspäivä (end_date) Ennustettavien rivien määrä (row_amount_generated)	Lisää annettujen tietojen mukaan uuden ennusteen tietokantaan.

6 Tulokset ja arviointi

6.1 Tavoitteiden toteutuminen

Työn tavoitteena oli muokata Bislenz-verkkopalvelun toteutettua ennustemallialgoritmia ja sille varattua AI-Forecast-näkymää, jotta se voidaan ottaa asiakaskäyttöön. Tutkimus keskitettiin seuraavien tutkimuskysymysten vastaamiseen.

1. Kuinka ennustemallialgoritmi toimii?
2. Miten ARIMA-ennustemallialgoritmi otetaan käyttöön verkkopalvelussa?
3. Soveltuuko ennustemalli verkkopalvelun tarpeisiin?

Tutkimuksen aikana selvitettiin sekä ARIMA-ennustemallin että Bislenz-verkkopalvelun toiminta ja käsiteltiin molempien käyttämiseen tarvittava teoria. Lopputuloksena ARIMA-mallin generointiarvoja vertailtiin nopeuden ja tarkkuuden mukaan, mistä saatujen tuloksien mukaan ennustemallin käyttöönottoon kehitettiin toimintasuunnitelma. Loput projektista keskittyi Bislenzin AI Forecast-näkymän kehittämiseen tämän toimintasuunnitelman mukaan.

Ennustemallille voidaan jo luoda ennusteita, jotka tallentuvat tietokantaan käyttöliittymästä. Ennustemallin generoinnin parametrit saatiin määritettyä palvelimelle, joiden mukaan ennusteen generointi on tarpeeksi nopeaa. Työn lopputuloksena saatiin valmiiksi kaikki ennustemallin toteuttamisen vaiheet, paitsi ennustemallin automatisointi, jonka toteuttamisessa ilmaantui ongelmia tietokantasarakkeiden automaattisessa määrittelyssä. Tämän ominaisuuden toteuttaminen osoittautui aikaa vieväksi ominaisuudeksi R-ohjelmointikielellä ja määritellyt ennusteet eivät päivity automaattisesti palvelimen puolelta, mikä jättää toteutuksen keskeneräiseksi. Jäljellä olevaa työmäärää arvioiden puuttuvat toiminnot voidaan toteuttaa 1–2 viikon täyspäiväisellä työpanoksella, jos automatisoinnin ongelmat ratkeavat suunnitelmien mukaan.

Tavoitteissa määritelty ennustemallialgoritmin toiminta käsiteltiin tarkasti teorian osalta, mutta käytännön sen toimintaa selitettiin vähemmän. Tältä osin toteutuksesta keskitettiin ennustemallin konfigurointiin, jossa valmiin ohjelman nopeutta ja tuloksia vertailtiin. Lopputuloksena saatiin kyllä toimivat ajoparametrit, mutta ennustemallin toimintaperiaatetta olisi voinut määrittää lisää sovelluksen puolella. Esimerkiksi mallille syötettyjä lähdedatoja ei muutettu, joiden avulla voitaisiin kartoittaa poikkeustilanteita eri datamäärillä ja tulosten heitoilla. Ennustemallille ei myöskään määritetty useamman sarakkeen generointia, mikä saattaa tapahtua erilaisella lähdedatalla.

Vaikka lopputulos jäikin keskeneräiseksi, sen toteutuksessa otettiin verkkopalvelun rajoitteet huomioon. Ennustemallin ajamisessa suosittiin vähemmän kuormittavia ratkaisuja, jotka

6.2 Jatkokehittämissuunnitelma

Lopputulokseksi suunniteltu kokonaisuus sisälsi datan generoimisen, tiedon tallentamisen tietokantaan, automaattisen päivittämisen Bislenz-verkkopalveluun sekä näkymän käyttöliittymän muokkaamisen. Itse toteutuksesta jäi kuitenkin ennustemallin optimoiminen sekä R-koodilla toimiva ARIMA-generaattorin toteuttaminen, sillä toteutuksessa tämä annettiin projektin alussa. Näiden aiheiden poistaminen työstä jätti tutkimukseen aukon, josta löytyisi ainesta jopa useaan uuteen tutkimukseen.

Ennustemalli voidaan toteuttaa palvelimen kautta generoimisen sijaan toisella tavalla: Generoiminen suoraan selaimen kautta. Tällä menetelmällä tietokantaan ei tarvita erillisiä tietokantatauluja tai generoitujen rivien tallennusta, mutta se kasvattaa käyttäjien sivun lataamisaikaa. Tämän vuoksi vaihtoehto vaatii generointilogiikan säätämistä, jotta ajamisnopeus saadaan laskettua hyväksyttävälle tasolle. Toteutuksen aikana tätä nopeutta ei saavutettu aikarajan puitteissa. Jos ennustemallia saadaan jatkokehitettyä, tätä menetelmää saatetaan käyttää Bislenz-verkkopalvelussa myöhemmin.

Ennustamisesta löytyy paljon tutkittavaa tutkimuksen aiheena. Datageneroinnissa ja ennustamisessa on pitkä historia, mutta aihe kehittyy jatkuvasti uusien menetelmien ja laskutapojen ilmaantuessa. Ennustemalleja on käsitelty harvoin, jos lainkaan tietotekniikan opinnäytetöissä ja tutkimuksessa käytetty ARIMA on vain yksi monista ennustemalleista.

Aikasarjan mukaan ennustamisen sijaan voidaan hyödyntää seuraavia ennustamis-menetelmiä:

- Tilanteeseen liittyvien muiden muuttujien avulla (Judgemental Forecasting),
- Trendeillä tai painotetuilla keskiarvoilla (Weighted averages),
- Neuroverkoilla ennustaminen (Neural Network-models)
- ennustemallin yhdistäminen tekoälyyn (Arificial Intelligence).

(Hyndman & Athanasopoulos, 2021)

7 Pohdinta

Työn teon aikana aiheesta ei ole tehty aikaisempaa tutkimusta Web-palvelunkehittämisen näkökulmasta, sillä se kuuluu aiheeltaan enemmän analytiikan ja matematiikan puolelle. Aiheen tarkasteleminen uudesta näkökulmasta hyödyttää kuitenkin niitä, jotka työskentelevät vastaavanlaisissa projekteissa tai vertailevat ennustustapoja toteutuksissaan. Toteutuksen avulla voidaan tutkia myös Bislenz-verkkopalvelua itseään ennakkotapauksena toimivasta markkinoinnin verkkopalvelusta, sillä siitä esitellään käytetyt toimintamallit ja tekniikat.

Työssä keskityttiin Bislenz-palvelun uuden ominaisuuden kehittämiseen. Määriteltyjen tutkimuskysymysten mukaiset tavoitteet saavutettiin ja dokumentissa noudatettiin valittua tutkimusmenetelmää, sisältäen tilanteeseen tarvittavat teoriat, tekniikat ja menetelmät. Dokumentti on havainnollistava, informatiivinen ja noudattaa asiatyyliä. Lopputulos noudattaa määrittelyssä luvattua tietosuojaa ja vastaa tehtyihin tutkimuskysymyksiin. Toteutuksen aikana kerätty teoriapohja on hyvin visuaalinen ja sisältää paljon esimerkkejä. Suurin osa kuvioista on koodiesimerkkejä, jotka ovat tulleet sovelluksesta itsestään, ei ulkoisista lähteistä. Hyvä suunnittelu oli toteutuksessa tärkeässä roolissa ja toteutus tullaan viemään jatkossa loppuun suunnitelmien mukaisesti.

Tekninen toteutus jäi osittain puutteelliseksi. Ennustemalli on suunniteltu ja dokumentoitu, mutta toteutuksen keskeneräiset osuudet jättivät avoimia kysymyksiä tutkimukseen, kuten ennustemallin automatisointiin, tietokannan tilarajojen, sekä palvelimen suorituskapasiteettiin liittyviä ongelmia. Toimeksiannossa määritellyt tarpeet ovat kuitenkin otettu huomioon työn lopputuloksessa.

Kokonaisuudessaan työn aikana saadut johtopäätökset ovat luotettavia ja työ etenee johdonmukaisesti. Dokumentin syy/seuraus -suhteet on esitelty ja ne seuraavat toisiaan. Tutkimustyössä käytetyt lähteet ovat kattavia ja aiheeseen nähden tuoreita, mutta yleistettävyyden parantamiseksi niitä olisi voinut olla enemmän

Tutkimuksen aikaiset testit vastasivat kysymyksiin, mutta ne suunniteltiin puutteellisesti aloittamaan eri lähtötilanteista. Esimerkiksi nopeustestissä ennustemallille ei määritetty datan pituutta, joten testien välissä käytettiin eri määrä historiadataa. Tämä muuttaa generointinopeuksia ja painottaa tuloksia vääriin lukemiin, joka haittaa lopputuloksen luotettavuutta.

Toteutettu ratkaisu on luonteeltaan erittäin tapauskohtainen. Lopputulos on kehitetty ennakkotapauksen mukaan ja käyttää ennakkotapauksessa käytettäviä tekniikoita. Työssä lopputulos vaikutti jo alkuvaiheessa selvältä, vaikka aiheet antoivatkin mahdollisuuden muunlaiseen johtopäätökseen. Toteutettavuudesta oli jo tarpeeksi näyttöä ennustemallin ohjelmistokoodin antaneiden asiantuntijoiden lausunnosta. Tavoitteen kannalta tärkein kysymys oli ennemmin: Miten toteutus kannattaa ja miten se halutaan suorittaa?

Lähteet

2020 Developer Survey, 2020., Tietotekniikan-kehittäjäkysely, Viitattu 9.5.2020, https://insights.stackoverflow.com/survey/2020

Athanasopoulos, G. & Hyndman. R. J., 2018, Forecasting: Principles and Practice, 2 p., Melbourne: OTexts, Oppaan verkkojulkaisu, Viitattu 9.5.2020, https://otexts.com/fpp2/

Bister, T., 2019, Tietojenkäsittelyn opinnäytetyö: viittoja ja karttoja tutkimisen ja kehittämisen teille, Jyväskylä: Jyväskylän ammattikorkeakoulu, Viitattu 25.4.2021, ISBN-9789518305432

Contributed Packages, 2021, Luettelo käytettävissä olevista CRAN-kirjastoista, Viitattu 15.5.2021, https://cran.r-project.org/web/packages/

Kananen, H. & Puoltaival, H., 2019, Tekoäly: Bisneksen uudet työkalut, tekoälyn ja datageneraation opas, Liettua: Balto-print

Ketä olemme, 2021, Tridea OY:n yrityksen mission, arvojen ja jäsenten esittelysivu, Viitattu 20.4.2021, https://tridea.fi/fi/keta-olemme/

Kuorikoski & Reijula, 2020. Laskennalliset mallit voivat lisätä julkisen päätöksenteon avoimuutta. Artikkeli Tampereen yliopiston verkkojulkaisussa, Viitattu 9.5.2021, https://www.tuni.fi/alustalehti/2020/05/26/laskennalliset-mallit-voivat-lisata-julkisen-paatoksenteon-avoimuutta

Palvelumme, 2021, Bislenz-verkkopalvelun ominaisuuksien esittelysivusto, Viitattu 21.4.2021, https://bislenz.com/palvelumme/

Palma W. 2016. Time Series Analysis., Chile: John Wiley & Sons, Aikasarjan analyysin oppaan verkkopainos, Viitattu 9.5.2020, https://ebookcentral-proquest-com.ezproxy.jamk.fi:2443/lib/jypoly-ebooks/reader.action?docID=4517503

What is JavaScript, 2021, JavaScript-ohjelmointikielen kuvaus Mozillan kehitysdokumenteissa, Viitattu 6.5.2021, https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

What is R, 2021, R-ohjelmistokirjaston esittely, Viitattu 5.5.2021, https://www.r-project.org/about.html

Why mySQL, 2021, mysql-tietokantaohjelmiston dokumentaatio Oraclen nettisivulla, Viitattu 3.5.2021, https://www.mysql.com/why-mysql/