Joni Toikkanen

TEKOÄLYAVUSTEINEN HAKUSANA-TUTKIMUSTYÖKALU HAKUKONEOP-TIMOINTIIN

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2025

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä/Tekijät	Joni Toikkanen
Työn nimi	Tekoälyavusteinen hakusanatutkimustyökalu hakukoneoptimointiin
Toimeksiantaja	Tmi JHK Toikkanen
Vuosi	2025
Sivut	41 sivua
Työn ohjaaja(t)	Miia Liukkonen

TIIVISTELMÄ

Opinnäytetyössä tutkitaan tekoälyn tehokkuutta hakusanatutkimuksessa verrattuna perinteisessä hakusanatutkimuksessa käytettäviin menetelmiin. Opinnäytetyön aikana kehitetään hakusanatutkimustyökalu, joka hyödyntää Google Ads-, Google Trends- ja OpenAI API -palveluita toiminnoissaan. Työkalu toteutetaan verkkosovelluksena ja teknisessä toteutuksessa keskitytään toimivien REST API -integraatioiden toteuttamiseen sovelluksen ja palveluiden välillä, tiedon tallentamiseen NoSQL-tietokantaan ja yksinkertaisen käyttöliittymän toteuttamiseen.

Opinnäytetyön toimeksiantajana toimii Tmi JHK Toikkanen, joka on opinnäytetyön tekijän oma yritys. Opinnäytetyön aikana syntynyttä sovellusta käytetään yrityksen sisäisenä työkaluna, jonka kehitystä jatketaan opinnäytetyön jälkeen. Sovellusta on tarkoitus käyttää verkkosivustojen hakukonesijoitusten parantamiseen ja hakukonemainontaan.

Opinnäytetyössä perehdytään alkuun syvällisemmin hakukoneoptimoinnin menetelmiin ja hakusanatutkimukseen, minkä jälkeen tutkitaan tekoälyn ja koneoppimisen hyödyntämistä hakusanatutkimuksessa. Sovelluksen teknisen toteutuksen osalta perehdytään tarkemmin REST API -arkkitehtuuriin, Googlen ja OpenAI:n tarjoamiin palveluihin sekä MongoDB -tietokannan ominaisuuksiin. Sovelluksen kehitystyössä lähdetään liikkeelle palvelinpuolen rakentamisesta ja edetään yksittäisten palvelumoduuleiden kehittämisellä kohti käyttöliittymän rakentamista. Käyttöliittymä rakentuu React.js-sovelluskehyksen ympärille. Sovelluksen API-rajapinnasta hakema trendidata visualisoidaan käyttöliittymässä viivakaavioiksi Recharts-kirjaston avulla. Käyttöliittymän yleisilme rakentuu Tailwind CSS-kehyksen ympärille.

Opinnäytteen lopputuloksena syntyy verkkosovellus, jolla käyttäjä voi hakea hakusanan trenditietoja Google Trends -palvelusta sekä hakusanaehdotuksia Google Ads API:n ja OpenAI API:n kautta. Loppupäätelmässä todetaan tekoälyn olevan tehokas työväline hakusanatutkimuksessa, joka täydentää hakusanojen tutkimisessa käytettävien menetelmien kirjoa tarjoamalla luovia hakusanaehdotuksia ja pitkän hännän hakusanoja. Tekoälyn tuottamien hakusanojen tehokkuuden varmentamiseen tarvitaan kuitenkin lisätutkimusta tilastoidun tiedon puuttuessa.

Asiasanat: hakukoneoptimointi, hakusanatutkimus, tekoäly, REST API

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Business Administration
Author (authors)	Joni Toikkanen
Thesis title	AI-assisted keyword research tool for search engine optimization
Commissioned by	Tmi JHK Toikkanen
Time	2025
Pages	41 pages
Supervisor	Miia Liukkonen

ABSTRACT

The thesis examines the effectiveness of AI in keyword research compared to traditional keyword research methods. The thesis develops a keyword research tool that leverages Google Ads, Google Trends and OpenAI APIs. The tool is implemented as a web application and the technical implementation focuses on implementing functional REST API integrations between the application and the services, storing data in a NoSQL database and implementing a simple user interface.

The thesis is commissioned by Tmi JHK Toikkanen, which is the author's own company. The application created during the thesis is used as an internal tool, and its development will be continued after the thesis. The application will be used to improve the search engine rankings of websites and for search engine advertising.

The thesis starts with an in-depth study of search engine optimization methods and keyword research, followed by an investigation of the use of artificial intelligence and machine learning in keyword research. The technical implementation of the application focuses on the REST API architecture, the services provided by Google and OpenAI, and the features of the MongoDB database. The development of the application started from the server side and progress through the development of individual service modules towards the development of the user interface. The user interface is built around the React.js application framework. The trend data retrieved from the So-vellus API interface is visualized in the UI as line graphs using the Recharts library. The overall look and feel of the interface are built around the Tailwind CSS framework.

The result of the thesis is a web application that allows the user to retrieve keyword trend data from the Google Trends service, as well as keyword suggestions via the Google Ads API and the OpenAI API. The conclusion states that artificial intelligence is a powerful tool for keyword research, complementing the range of methods used to explore keywords by providing creative keyword suggestions and long-tail keywords. However, further research is needed to verify the effectiveness of AI-generated keywords in the absence of statistical data.

Keywords: search engine optimization, search term research, artificial intelligence, REST API

SISÄLLYS

1 JOHDANTO

Opinnäytetyössä kehitetään tekoälyavusteinen hakusanatutkimustyökalu, jonka tarkoituksena on parantaa hakukoneoptimoinnin strategioita, joita käyttävät esimerkiksi sisällöntuottajat, markkinoijat ja verkkokauppiaat. Projektissa keskitytään useiden eri tietolähteiden integrointiin REST API -rajapinnan kautta. Hakusanatutkimustyökalussa hyödynnetään Google Trends -datan ja Google Ads API -rajapinnan lisäksi OpenAI:n luonnollisen kielen käsittelyä hakusanojen analysoimisessa ja laajentamisessa.

Tekoäly pystyy tuottamaan merkityksellisiä ja kontekstin huomioivia hakusanaehdotuksia muutamassa sekunnissa. Tekoälyominaisuuden tuominen osaksi opinnäytetyössä kehitettävää sovellusta on ajankohtaista, koska markkinoilla on nähtävissä selvä trendi kohti tekoälyavusteisia sovelluksia ja AI-ominaisuuksien puute sovelluksessa voi tulevaisuudessa olla heikkous markkinoilla. Tekoäly ei ole enää pelkkä lisäominaisuus, vaan siitä on tulossa lähinnä standardi moderneissa sovelluksissa. Tämän takia myös opinnäytetyössä kehitettävään hakusanatutkimustyökaluun integroidaan tekoäly, koska tavoite on tarjota käyttäjälle mahdollisimman tehokas tapa löytää hakusanoja, jotka parantavat verkkosivustojen, blogien ja verkkokauppojen hakukonenäkyvyyttä.

Hakukoneoptimointi koskettaa kaikkia yrityksiä ja järjestöjä, jotka käyttävät toiminnassaan verkkosivuja tai verkkopalveluita. Tämä aihe koskettaa myös oman yritykseni toimintaa, joten toteutettava työkalu kehitetään pääosin oman yritykseni tarpeisiin. Hakusanatyökalua voidaan käyttää orgaanisen hakukonesijoituksen parantamisen lisäksi hakukonemainonnassa, jolloin työkalun avulla voidaan etsiä klikkaushinnaltaan halvempia mutta silti suosittuja hakusanoja. Työkalua voidaan myös hyödyntää Amazonin ja Etsyn kaltaisilla markkina-alustoilla myynti-ilmoitusten toteuttamiseen.

Opinnäytetyön tavoitteena on pystyä kehittämään toimiva prototyyppi verkkopohjaisesta hakusanatutkimustyökalusta, jolla voidaan tekoälyn avulla laajentaa hakusanojen etsimistä ja analysointia. Tavoitteena on myös vertailla teko-

älyn tuottamia hakusanaehdotuksia perinteisten hakusanatutkimusmenetelmien tuottamiin hakusanoihin ja arvioida niiden tehokkuutta hakukoneoptimoinnissa.

Tutkimusongelmana on selvittää, kuinka tekoälyavusteinen hakusanatutkimustyökalu voi tehostaa hakusanojen analyysia hakukoneoptimoinnissa verrattuna perinteisiin menetelmiin. Tämän lisäksi työssä tutkitaan, kuinka tarkasti tekoäly pystyy tuottamaan käyttökelpoisia pitkiä avainfraaseja (engl. long-tail keyword) hakukoneoptimointiin.

2 HAKUSANATUTKIMUKSEN MERKITYS HAKUKONEOPTIMOINNISSA

Hakukoneoptimointi (SEO, Search Engine Optimization) on digitaalisen markkinoinnin strategia, jonka tavoitteena on parantaa verkkosivuston näkyvyyttä Googlen kaltaisissa hakukoneissa. Näkyvyyttä hakukoneissa voidaan parantaa optimoimalla verkkosivuston sisältöä, rakennetta ja teknisiä ominaisuuksia tavoilla, joilla hakukoneet ymmärtävät paremmin sivuston käyttäjälle tarjoaman sisällön. Keskeisenä osana hakukoneoptimointia toimii hakusanatutkimus, sisällön optimointi, linkkien rakentaminen sekä sivuston sisäinen ja ulkoinen optimointi. Hakusanatutkimus on prosessi, jossa selvitetään, mitä hakusanoja kohderyhmä käyttää hakiessaan tietoa palveluista ja tuotteista hakukoneissa. Tämän avulla sisältöä voidaan kohdentaa vastaamaan paremmin käyttäjien tarpeisiin. (Fano Oy 2023.)

2.1 Hakukoneoptimoinnin menetelmät

Hakukoneoptimoinnissa sisällön optimoinnilla tarkoitetaan tekstien, kuvien ja muiden elementtien muokkaamista muotoon, joka vastaa sekä käyttäjien että hakukoneiden odotuksia. Tähän kuuluu esimerkiksi hakusanojen luonnollinen käyttö tekstissä, laadukkaan ja informatiivisen sisällön tuottaminen sekä metatietojen, kuten otsikoiden ja kuvauksien optimointi. Onnistuminen hakukoneoptimoinnissa perustuu dataan ja analytiikkaan. Hakusanatutkimuksen avulla selvitetään, millä termeillä verkkosivusto tällä hetkellä löytyy, paljonko kyseisillä termeillä on kuukausittaisia hakuja ja kuinka kilpailtuja nämä hakutermit

ovat. Näillä tiedoilla voidaan kohdentaa hakukoneoptimoinnin strategiaa tehokkaammin ja saavuttaa parempia tuloksia. (Ilomäki 2023.)

Käyttäjäkokemuksen parantaminen on myös olennainen osa hakukoneoptimointia. Tämä sisältää verkkosivuston nopeuden optimoinnin, responsiivisen suunnittelun eri laitteille sekä helpon navigoinnin sivustolla. Tämän lisäksi on tärkeää, että verkkosivusto käyttää SSL-suojausprotokollaa ja sivuston rikkinäiset linkit on poistettu (Webbee Oy 2022). Googlen kaltaiset hakukoneet arvostavat sivustoja, jotka tarjoavat positiivisen käyttäjäkokemuksen, mikä voi johtaa parempiin sijoituksiin hakutuloksissa. Yhteenvetona hakukoneoptimointi on monivaiheinen prosessi, joka vaatii jatkuvaa seurantaa ja mukauttamista. Sen avulla pyritään parantamaan sivuston näkyvyyttä, lisäämään kävijämäärää ja lopulta saavuttamaan liiketoiminnallisia tavoitteita.

2.2 Hakusanatutkimus

Hakusanatutkimus, joka tunnetaan myös nimellä avainsanatutkimus, toimii hakukoneoptimoinnin perustana, jonka avulla tunnistetaan hakutermit, joita potentiaaliset asiakkaat käyttävät etsiessään tuotteita tai palveluita verkosta. Tämän prosessin tavoitteena on löytää verkkosivuston tavoitteiden kannalta olennaisia aiheita tai hakusanoja, joista kirjoittamalla ja tuottamalla muuta sisältöä on mahdollista saada verkkoliikennettä hakukoneista, kuten Googlesta tai Youtubesta. Hakusanojen lisäksi tutkimus tarjoaa lisätietoa siitä, kuinka paljon hakusanaan liittyviä aiheita haetaan ja kuinka kilpailtuja ne ovat. (Tyyskä 2023.)

Hakusanatutkimuksen avulla voidaan selvittää, mitkä liiketoimintaan tai myytäviin tuotteisiin tai palveluihin liittyvät aihealueet kiinnostavat potentiaalisia asiakkaita kaikkein eniten. Tämän tiedon avulla pystytään kohdentamaan sisältöä vastaamaan paremmin käyttäjien tarpeita ja parantamaan sivuston näkyvyyttä hakutuloksissa.

Hakusanatutkimus voidaan jakaa neljään vaiheeseen:

1. **Hakusanaideoiden etsiminen**: Tässä vaiheessa kerätään mahdollisimman paljon hakusanaideoita esimerkiksi Excel-taulukkoon. Listatuista hakusanoista muodostetaan myös yhdistelmiä ja lähellä olevia versioita. Ideointiin on mahdollista käyttää myös erilaisia työkaluja hakusanaideoiden löytämiseksi. Yksi tällainen työkalu on Google Keyword Planner, jota hyödynnetään myös toteutettavassa sovelluksessa.
2. **Hakusanojen kilpailun analysointi**: Arvioidaan, kuinka suuri kilpailu hakusanoilla on. Lyhyemmillä hakusanoilla on yleensä kovempi kilpailu ja yleisesti tunnetut brändit ja verkkosivustot näkyvät näiden hakusanojen kohdalla kärkijoukossa. Tuntemattomat verkkosivustot voivat kuitenkin hyödyntää niin sanottuja pitkän hännän hakusanoja, jotka saavat vähemmän hakukertoja, mutta kuvaavat tarkemmin haettavaa aihetta.
3. **Hakutarkoituksen varmentaminen**: Varmistetaan, mitä tarkoitusta varten haku on tehty. Hakujen syitä ovat tiedonhaku, siirtyminen, kaupallinen tutkimus ja ostaminen. Hakusanojen asettelusta voidaan päätellä mitä tarkoitusta varten käyttäjä hakee tietoa.
4. **Tärkeimpien avainsanojen valinta**: Keskitytään yritykselle tärkeimpiin avainsanoihin. Mikäli hakusanalla on korkea klikkauskohtainen hinta, viittaa tämä siihen, että hakusanalla tavoittaa maksavia asiakkaita. Google Keyword Plannerin avulla voidaan tutkia eri hakusanojen klikkauskohtaisia hintatietoja. (Kallio 2023.)

Onnistunut hakusanatutkimus perustuu dataan. Datan avulla pystytään selvittämään, millä termeillä sivusto tällä hetkellä löytyy, paljonko kyseisillä termeillä on kuukausittain hakuja ja kuinka kilpailtuja avainsanat ovat. Näiden tietojen avulla pystytään kohdentamaan hakukoneoptimoinnin strategiaa tehokkaasti ja saavuttamaan parempia tuloksia.

2.3 Tekoälyn ja koneoppimisen hyödyntäminen hakusanatutkimuksessa

Tekoälyn ja koneoppimisen kehittyminen on mullistanut monia liiketoiminnan osa-alueita. Näihin osa-alueisiin lukeutuu myös hakukoneoptimointi ja hakusanatutkimus. Tekoäly mahdollistaa suurien datamäärien analysoinnin, hakutot-

tumusten ennustamisen ja trendien tunnistamisen tehokkaammin kuin perinteisillä menetelmillä toteutettuna. Luonnollisen kielen käsittely, joka tunnetaan myös nimellä NLP (Natural Lanquaqe Processing), auttaa tunnistamaan semanttisesti samankaltaisia hakusanoja ja laajentamaan hakusanatutkimuksen käyttömahdollisuuksia (Numminen 2023).

Tekoälyn avulla voidaan myös optimoida verkkosivuston sisältöä paremmin hakukoneita varten. Esimerkiksi Googlen kehittämä BERT-algoritmi on muuttanut hakutulosten relevanssin arviointia, jolloin hakusanojen merkitys riippuu entistä enemmän niiden kontekstista (Devlin ym. 2019). Tekoälypohjaiset hakusanatyökalut, kuten SEMrush ja Ahrefs, tarjoavat arvokasta tietoa hakukoneoptimoijille käyttämällä koneoppimista analysoimaan kilpailua, hakusanojen hakuvolyymiä ja klikkausprosentteja.

Koneoppimismallit, kuten neuroverkot, voivat ennustaa, mitkä hakusanat tuottavat eniten liikennettä ja muuntautumista asiakkaiksi. Esimerkiksi regressiomallit voivat analysoida historiallista dataa ja ennustaa tulevia hakutrendejä. Mitä enemmän koneoppimisen mallit pääsevät hyödyntämään aiemmin kerättyä dataa, sitä paremmin ne pystyvät löytämään säännönmukaisuuksia ja nousevia trendejä (Haltu Oy 2024). Näin ollen tekoälyn hyödyntäminen hakusanatutkimuksessa auttaa yrityksiä optimoimaan sisältöään strategisemmin ja tehokkaammin.

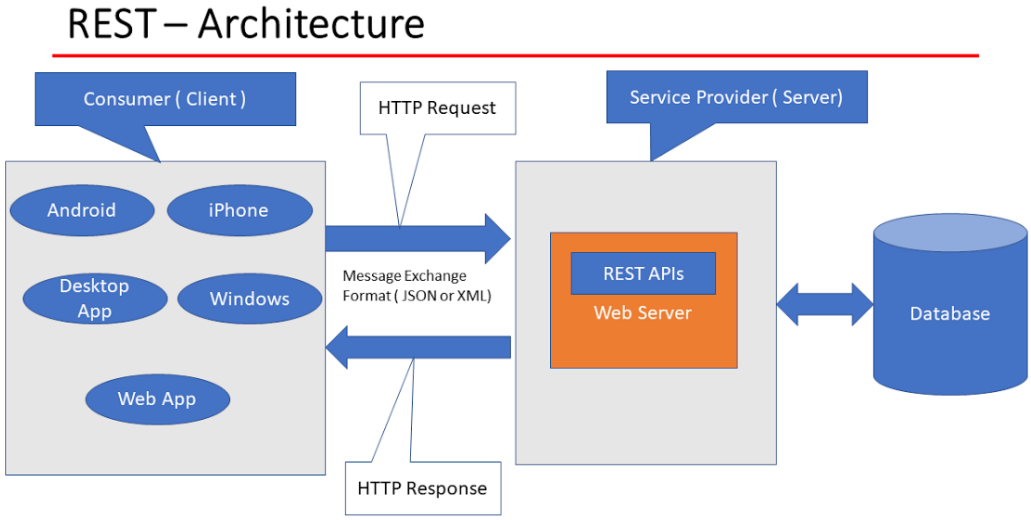
3 HAKUSANATUTKIMUSTYÖKALUN TEKNOLOGIA

Hakusanatutkimustyökalun kehittämiseen hyödynnetään useita eri teknologioita, joista käydään seuraavana läpi kaikkein oleellisimmat. Työkalussa käytettävä tieto tulee sovellukseen REST API -rajapinnan kautta, joten on syytä tutkia tätä teknologiaa hieman syvemmin. Lisäksi tämä luku avaa niitä syitä, joiden vuoksi sovellukseen on valittu Googlen tuottamat palvelut hakusanaanalyysia varten.

3.1 Rest-rajapinnan toimintaperiaate

Opinnäytetyössä toteutettavaan sovellukseen integroidaan useita ulkopuolisia palveluita REST API -rajapintaa hyödyntäen. Tämän takia on syytä ensin tut-

kia REST API -arkkitehtuurityylin toimintaperiaatetta. Fielding (2000, 76) esitteli väitöskirjassaan Representational State Transfer (REST) arkkitehtuurityylin. Tämä HTTP-protokollaan perustuva arkkitehtuurityyli on suunniteltu hajautettujen järjestelmien, kuten verkkopalveluiden toteuttamiseen.



Kuva 1. Kuvaus REST arkkitehtuurista (Gitonga 2021)

Kuvan 1 asiakas (engl. client) voi olla esimerkiksi mobiili-, verkko- tai työpöytäsovellus, joka lähettää HTTP-pyyntöjä palvelimelle (engl. server). Palvelin sisältää REST-rajapintoja, jotka sijaitsevat verkkopalvelimella. Palvelin ottaa vastaan asiakkaan pyynnön ja käsittelee sen. Kun palvelin on käsitellyt pyynnön, se lähettää HTTP-vastauksen asiakkaalle.

REST-arkkitehtuuri perustuu muutamaan keskeiseen periaatteiseen:

1. **Resurssit:** REST-arkkitehtuurissa tiedot esitetään resursseina, joilla on yksilöllinen URI (Uniform Resource Identifier). URI-rakenteen tulisi olisi olla mahdollisimman ymmärrettävä. Tällainen URI-rakenne on mahdollista saavuttaa rakentamalla hakemistorakenteen kaltaiset URI:t. Tämänkaltainen URI-rakenne on hierarkkinen, ja sen juurena on yksi polku, joka taas haarautuu alipolkuihin, joiden nimet paljastavat palvelun pääalueet. Esimerkkinä hierarkkisesta URI-rakenteesta voidaan käyttää seuraavaa rakennetta: https://esimerkki.com/users/{userId}. Esimerkissä URI:n juuren /users/ jälkeen voidaan hakea tiettyä käyttäjää id-numeron avulla.
2. **HTTP-menetelmät:** REST-rajapinnat käyttävät HTTP-protokollan menetelmiä resurssien käsittelyyn. Näillä menetelmillä REST-rajapinta suorittaa CRUD-operaatioita, joka on lyhennys sanoista create, read, update ja delete. Seuraavana listattuna HTTP-protokollan menetelmät, joilla CRUD-operaatioita suoritetaan REST-rajapinnassa:
    - POST-metodia käytetään uuden resurssin luomiseen

- GET-metodilla haetaan resurssi
- PUT-metodilla päivitetään olemassa oleva resurssi
- DELETE-metodilla poistetaan olemassa oleva resurssi.

3. **Tilaton:** Jokainen REST-pyyntö on tilaton, mikä tarkoittaa sitä, että palvelin ei tallenna mitään tietoja REST-verkkopalveluasiakkaan tilasta. Tämän takia jokaisessa pyynnössä on oltava kaikki tarvittavat tiedot.
4. **Resurssien esitykset:** Resurssit esitetään yleisimmin JSON- tai XML-muodossa, mutta näiden lisäksi resurssi voidaan esittää myös XHTML-muodossa. (Rodriguez 2015.)

HTTP-otsikot ovat olennainen osa API-pyyntöä ja -vastausta, sillä ne edustavat API-pyynnön ja -vastauksen metatietoja. Otsikot luokitellaan pyyntö- ja vastausotsikoiksi. Otsikot sisältävät tietoa asioista, kuten pyynnön ja vastauksen rungon sekä pyynnön valtuutuksen. Yleisimmät otsikot, jotka tulevat vastaan myös tämän opinnäytetyön API-pyynnöissä, ovat authorization, WWW-authenticate, accept-charset ja content-type. Authorization-otsikko on osa HTTP-pyyntöotsikoita, joita käytetään asiakkaan ja palvelimen välisessä viestinnässä. Tämän otsikon ensisijainen tehtävä on todentaa käyttäjä palvelimen kanssa. Yleensä käyttäjä välittää todennustiedot salasanan muodossa API-pyynnön yhteydessä. (Beeceptor s.a.) Tämä otsikko on olennaisen tärkeä verkkosovellusten turvatoimien toteuttamisessa. WWW-Authenticate on palvelimen lähettämä otsikko, jolla palvelin ilmoittaa tarvitsevansa todennuksen ennen kuin voi lähettää pyydetyn resurssin käyttäjälle. Tämä otsikko lähetetään yleensä 401 vastauskoodin kanssa, mikä tarkoittaa ei valtuutettua. Accept-Charset otsikolla ilmoitetaan palvelimelle merkistöt, jotka asiakas hyväksyy. Content-Type ilmaisee palvelimen asiakkaalle lähettämän vastauksen mediatyypin, joka auttaa asiakasta käsittelemään vastauksen runkoa oikein. (SmartBear Software Inc s.a.)

3.2 Google Trends, Google Ads ja OpenAI API hakusana-analyysissä

Google Trends ja Google Ads Keyword Planner ovat suosittuja työkaluja, joita käytetään hakusana-analyysiin hakukoneoptimoinnissa. Google Trends mahdollistaa hakutermien suosion seuraamisen ajassa, jolloin voidaan tunnistaa nousevia trendejä ja sesonkikohtaisia muutoksia. Google Ads Keyword Planner puolestaan tarjoaa tietoa hakusanojen hakumääristä, kilpailutilanteesta ja mainonnan klikkihinnoista. (Google LLC s.a.)

Google Trends on hyödyllinen työkalu pitkän aikavälin hakukäyttäytymisen analysoinnissa. Se auttaa ymmärtämään, mitkä aiheet ja hakutermit ovat kasvussa ja mitkä menettävät suosiotaan. Tämä tieto auttaa hakukoneoptimoinnissa ja sisältömarkkinoinnissa. Verkkosivustot voivat kohdistaa sisältöä hakusanalle, jonka suosio on nousussa ennen kuin kilpailu kasvaa liian kovaksi. Google ei kuitenkaan tarjoa omaa REST API -rajapintaa Google Trends -palvelulle, joten opinnäytetyössä käytetään toista palveluntarjoajaa Google Trends -datan tuomiseksi sovellukseen, joka on SerpAPI.

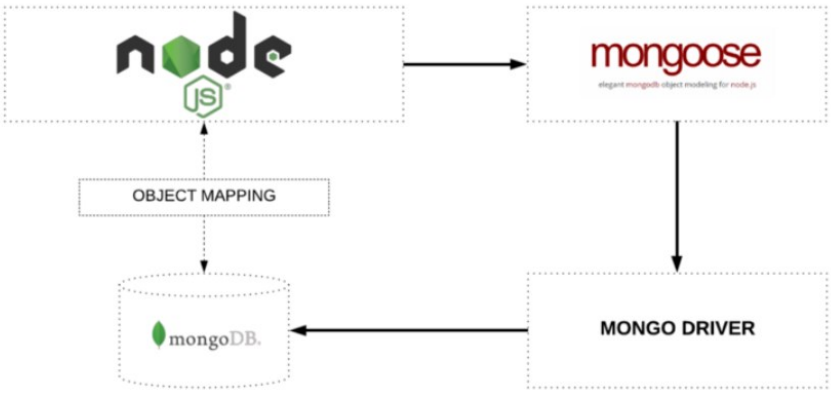
Google Ads Keyword Planner on hakusanojen suunnitteluun tarkoitettu ilmainen työkalu, joka on suunniteltu maksullisten hakusanamainosten suunnitteluun Google Ads -alustalla (Helpotkotisivut.fi MS Oy 2018). Google Ads Keyword Planner tarjoaa tarkempia tietoja hakusanojen hakuvolyymeista ja kilpailusta. Sitä hyödynnetään yleensä maksullisessa hakukonemainonnassa, mutta sitä voidaan myös hyödyntää orgaanisessa hakukoneoptimoinnissa tunnistamalla hakusanoja, joilla on korkea hakumäärä, mutta matala kilpailu. Google Trends ja Google Ads Keyword Planner tulevat olemaan keskeisiä työkaluja toteutettavassa sovelluksessa. Niiden avulla käyttäjä pystyy ymmärtämään markkinatrendejä, kohdentamaan sisältöä ja optimoimaan verkkosivuston näkyvyyttä hakukoneissa.

Käyttäjät voivat integroida OpenAI:n tehokkaita koneoppimismalleja omiin projekteihin OpenAI:n tarjoaman API-rajapinnan kautta. OpenAI tarjoaa useamman esikoulutetun koneoppimismallin, joista tuorein julkaisu on GPT-4o, joka pystyy ymmärtämään ja tuottamaan sekä tekstiä että koodia. OpenAI tarjoaa myös koneoppimismalleja kuvien ja äänien generointiin. Koneoppimismalleja on myös mahdollista mukauttaa käyttäjän omiin tarpeisiin. Tässä tapauksessa käyttäjä hienosäätää valmiiksi opetettua koneoppimismallia oman harjoitusdatan avulla. Tällainen hienosäätö mahdollistaa alhaisemman viiveen pyynnöissä ja tuo käyttäjälle kustannussäästöjä. (Thevapalan 2023.) Opinnäytetyössä hyödynnetään OpenAI:n API-rajapintaa tuomalla tekoälyn tuottamia hakusanaehdotuksia sovellukseen.

3.3 MongoDB ja Mongoose tietokantaratkaisuina

MongoDB on MongoDB Inc -yhtiön kehittämä NoSQL-tietokanta, joka on suunniteltu skaalautuvaksi ja suorituskykyiseksi, kun käsitellään suurta määrää dataa. MongoDB poikkeaa perinteisestä relaatiotietokannasta sillä, että se käyttää dokumenttipohjaista tallennusmallia, jossa tiedot tallennetaan JSON-tiedostotyypin kaltaisiin BSON-dokumentteihin. (MongoDB Inc s.a.) MongoDB mahdollistaa dynaamisen skeeman, joka soveltuu erityisen hyvin sovelluksiin, jotka käsittelevät vaihtelevaa ja suurikokoista dataa.

Mongoose on MongoDB:lle kehitetty objektimallinnuskirjasto (ODM), joka sisältää työkalut tietokannan hallintaan ja validointiin Node.js-sovelluksissa. Mongoosen avulla pystytään määrittelemään skeemoja, validoida tietoa ja määritellä yhteyksiä, jolloin MongoDB:n käytöstä tulee kehittäjäystävällisempää. Sillä voidaan esimerkiksi määritellä, että tietokannan dokumenteissa tulee olla tietyt kentät ja niiden tietotyyppien tulee olla oikeita.



Kuva 2. Kuvakaappaus objektien kartoituksesta Noden ja MongoDB:n välillä hyödyntämällä mongoosea (Karnik 2018)

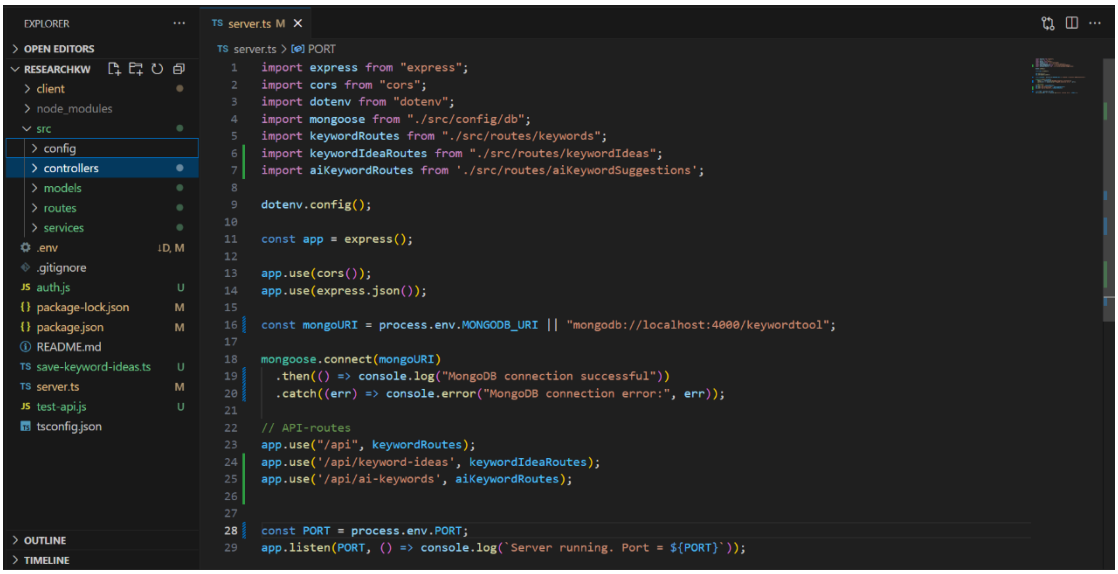
MongoDB ja Mongoose muodostavat tehokkaan ratkaisun verkkosovelluksille, jotka vaativat joustoa tietomallinnuksessa. Siksi ne soveltuvat tilanteisiin, joissa joudutaan käsittelemään suuria määriä puolistrukturoitua dataa, kuten hakusanatutkimuksessa käytettäviä hakutietoja.

4 PALVELINPUOLEN TOTEUTUS

Hakusanatutkimustyökalun kehitys aloitetaan palvelinpuolen rakentamisesta. Tässä vaiheessa asennetaan tarvittavat työkalut projektille ja rakennetaan yhteys tietokantaan. Tämän jälkeen työkaluun tuotavat ominaisuudet rakennetaan itsenäisiksi moduuleiksi yksi kerrallaan.

4.1 Projektin aloitus

Aloitin kehitysprojektin luomalla sovellukselle repositorion Github -palveluun. Kloonasin luodun repositorion omalle tietokoneelle ja alustin sovelluksen hakemistoon uuden Node.js -projektin. Seuraavaksi asensin projektiin Express.js-, dotenv-, axios-, cors- ja mongoose-paketit terminaalin kautta node package managerilla. Projektissa tullaan käyttämään TypeScriptiä ohjelmointikielenä, joten projektiin asennetaan TypeScript -kääntäjä, ts-node-paketti sekä tyypitystuet Express.js-, Node.js-, cors- ja mongoose-paketeille. Lopuksi alustetaan TypeScript -projekti, jonka yhteydessä luodaan tsconfig.json-tiedosto. Lisäsin projektikansioon myös .env-tiedoston, johon tallennetaan serveripuolen ympäristömuuttujat. Tietokantaa varten loin MongoDB Cloud -palvelussa sovellukselle oman klusterin, johon sovelluksen hakemat tiedot tallennetaan.



Kuva 3. Kuvakaappaus server.ts-tiedostosta

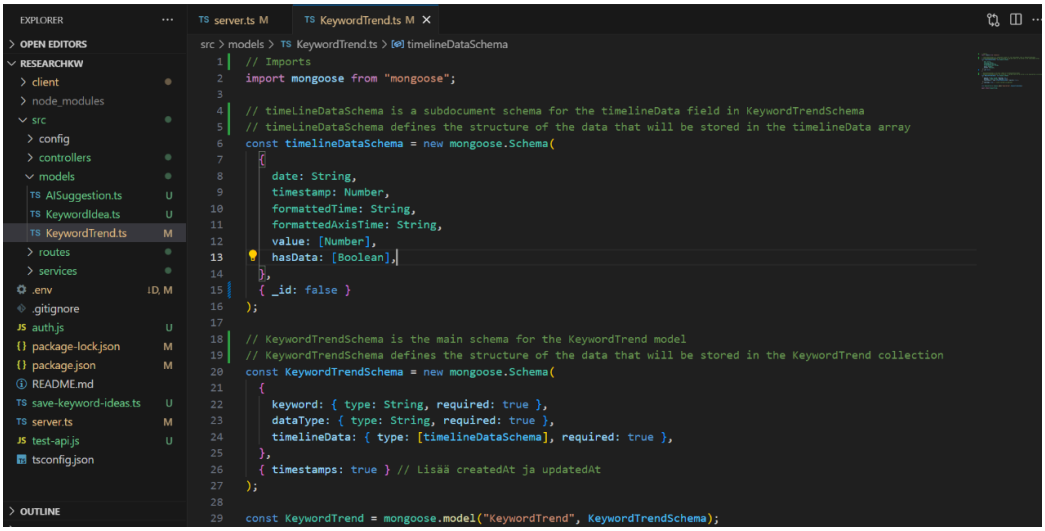
Ensimmäisenä loin projektiin serveripuolen tiedostorakenteen ja yhteyden MongoDB -tietokantaan. Kuvan 3 server.ts-tiedostossa, joka sijaitsee projektin juurikansiossa, luodaan yhteys MongoDB -tietokantaan mongoosen avulla. Samaan tiedostoon luodaan myös Express -applikaatio, jolla voidaan määritellä mitä funktioita kutsutaan eri HTTP-metodien ja URL-polkujen yhteydessä. Tiedostoon on määritelty API-reitit ja reititinfunktiot sovelluksen eri ominaisuuksille. Nämä reitit ja funktiot määritetty app.use()-funktioiden sisään, joissa annetaan ensin parametrinä URL-polku ja tämän jälkeen reititinfunktio. Cors-paketin käyttöönotto antaa käyttöliittymäpuolelle mahdollisuuden tehdä pyyntöjä serveripuolen Express -palvelimelle, koska käyttöliittymä toimii paikallisella palvelimella eri portissa kuin sovelluksen serveripuoli.

Kehitysvaiheessa sovelluksen serveripuoli on määritelty toimimaan paikallisen palvelimen portissa numero 4000, jonka URL-osoite on http://localhost:4000. Portin numero ei käy ilmi server.ts-tiedoston lähdekoodista, koska portin numero annetaan PORT-nimisenä ympäristömuuttujana app-oliolle, joka app.listen()-metodilla kuuntelee määritetyn portin yhteyksiä. Myös MongoDB -tietokannan yhteyden luomiseen tarvittava MongoDB URI on piilotettu ympäristömuuttujaan nimeltä MONGODB_URI, koska se sisältää mm. käyttäjätunnuksen ja salasanan MongoDB -palvelussa luomaani klusteriin. Lähdekoodissa käytetään ympäristömuuttujia tietoturvasyistä, koska API-pyyntöihin tulee monesti liittää mukaan arkaluontoisia kirjautumistietoja, joiden ei haluta näkyvän lähdekoodissa esimerkiksi Github -palvelussa. Ympäristömuuttujia varten projektiin asennettiin dotenv-paketti, joka lukee .env-tiedoston dotenv.config()-funktiokutsulla, jolloin ympäristömuuttujat ovat käytettävissä sovelluksen lähdekoodissa.

4.2 Google Trends- ominaisuuden kehittäminen

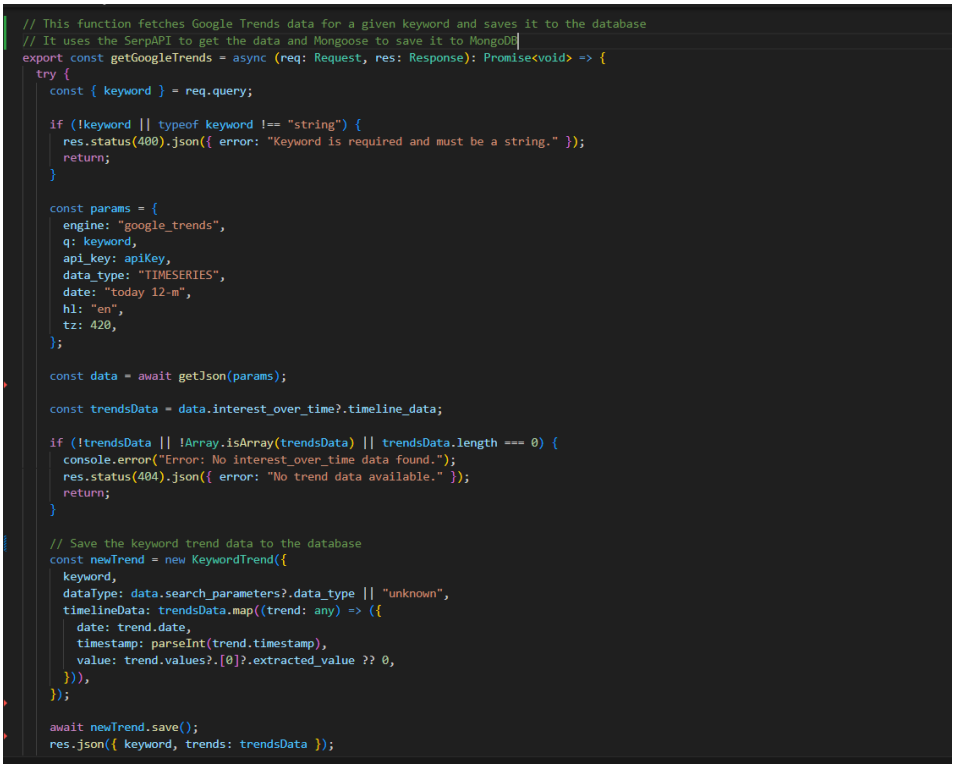
Google Trends oli ensimmäinen hakusanatyökaluun kehitetty toiminnallinen moduuli. Se hakee reaaliaikaisia trenditietoja käyttäjän syöttämistä hakusanoista SerpAPI:n avulla ja tallentaa ne MongoDB -tietokantaan visualisointia varten. Moduulin tavoitteena on antaa käyttäjälle mahdollisuus etsiä hakusanaa ja tarkastella sen suosiota ajan mittaan. Näin saadaan tietoa siitä, että onko hakusanan suosio nousussa, laskussa vai pysyykö se vakaana. Moduulin toimintaperiaate voidaan listata seuraaviin vaiheisiin:

1. käyttäjä syöttää hakusana hakukenttään
2. hakusana lähetetään palvelinpuolelle
3. palvelin lähettää get-pyynnön SerpAPI -palveluun hakusanan trenditietojen hakemista varten
4. trenditiedot tallennetaan tietokantaan
5. tiedot visualisoidaan viivakaavion avulla käyttöliittymässä.



Kuva 4. Kuvakaappaus KeywordTrend.ts-tiedostosta

Kuvan 4 KeywordTrend.ts-tiedosto sisältää tietomallit SerpAPI:n rajapinnasta haettavaa trendidataa varten. Tähän tiedostoon on luotu skeemat, jotka määrittävät MongoDB -tietokantaan tallennettavien dokumenttien tyypit ja niiden sisältämät kentät. Pääasiallinen dokumenttimalli on nimeltään KeywordTrendSchema ja se ottaa yhtenä tietotyyppinä vastaan edellä määritetyn alidokumentin nimeltään timelineDataSchema, joka sisältää listan hakusanan historiatiedoista. Näitä historiatietoja käytetään datan visualisointiin sovelluksen käyttöliittymässä.



```
// This function fetches Google Trends data for a given keyword and saves it to the database
// It uses the SerpAPI to get the data and Mongoose to save it to MongoDB
export const getGoogleTrends = async (req: Request, res: Response): Promise<void> => {
  try {
    const { keyword } = req.query;

    if (!keyword || typeof keyword !== "string") {
      res.status(400).json({ error: "Keyword is required and must be a string." });
      return;
    }

    const params = {
      engine: "google_trends",
      q: keyword,
      api_key: apiKey,
      data_type: "TIMESERIES",
      date: "today 12-m",
      hl: "en",
      tz: 420,
    };

    const data = await getJson(params);

    const trendsData = data.interest_over_time?.timeline_data;

    if (!trendsData || !Array.isArray(trendsData) || trendsData.length === 0) {
      console.error("Error: No interest_over_time data found.");
      res.status(404).json({ error: "No trend data available." });
      return;
    }

    // Save the keyword trend data to the database
    const newTrend = new KeywordTrend({
      keyword,
      dataType: data.search_parameters?.data_type || "unknown",
      timelineData: trendsData.map((trend: any) => ({
        date: trend.date,
        timestamp: parseInt(trend.timestamp),
        value: trend.values?.[0]?.extracted_value ?? 0,
      })),
    });

    await newTrend.save();
    res.json({ keyword, trends: trendsData });
```

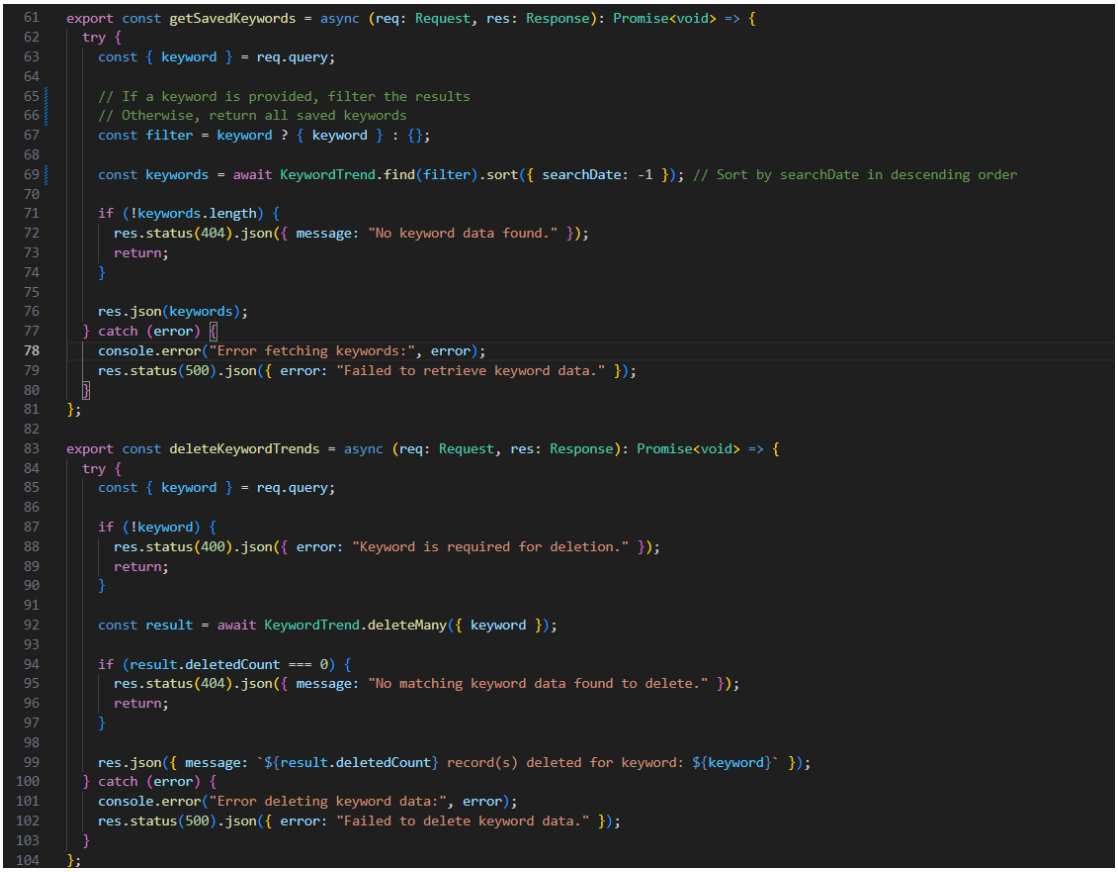
Kuva 5. Kuvakaappaus KeywordController.ts-tiedostosta

Keywords.ts-tiedosto sisältää reitit ja funktiokutsut trenditietojen hakemiseen ja tallentamiseen SerpAPI:n rajapinnasta, tallennettujen tietojen hakemiseen tietokannasta sekä trenditietojen poistamiseen tietokannasta. Näiden reittien kutsumat funktiot ovat erillisessä keywordController.ts-tiedostossa, joka näkyy kuvassa 5. Eniten toimintoja sisältävä funktio on nimeltään getGoogleTrends-funktio, jossa käyttäjän antamalla hakusanalla haetaan SerpAPI:n rajapinnasta kyseisen hakusanan trenditiedot. Palvelinpuolelle on asennettu SerpAPI:n node-paketti, joka tarjoaa sisäänrakennetut funktiot API-pyyntöjen lähettämiseen. Rajapintaan lähetettävä pyyntö sisältää seuraavat parametrit:

- *engine*, jolla ilmaistaan haluttu tietolähde palveluntarjoajan valikoimasta
- *q*, joka sisältää haettavan hakusanan
- *api_key*, joka sisältää palveluntarjoajalta hankitun API-avaimen
- *data-type*, jolla ilmaistaan haluttu tietotyyppi
- *date*, joka määrittää halutun ajanjakson
- *hl*, joka määrittää halutun kielen
- *tz*, jolla määritetään haluttu aikavyöhyke

Näistä parametreista *engine*, *g*, ja *api_key* ovat välttämättömiä pyynnön onnistumisen suhteen. Sovelluksen lähettämässä pyynnössä on määritelty trenditietojen aikajaksoksi viimeisen 12 kuukauden historiatiedot, jolloin vastaus sisältää viikoittaiset trenditiedot vuoden ajalta. Pyyntö lähetetään parametrei-

neen SerpAPI:n getJson-funktiota käyttäen palvelimelle, minkä jälkeen palvelin lähettää takaisin vastauksen, mikäli pyyntö sisältää palveluntarjoajan vaatimat parametrit. SerpAPI:n lähettämästä vastauksesta muodostetaan KeywordTrend-tietotyypin mukainen olio, johon tallennetaan hakusana, datatyyppi sekä historiatiedoista iteroidaan lista, joka sisältää päivämäärän, aikaleiman ja trendiarvon kyseiseltä viikolta. Tämä olio tallennetaan MongoDB -tietokantaan dokumentiksi.



```
export const getSavedKeywords = async (req: Request, res: Response): Promise<void> => {
  try {
    const { keyword } = req.query;

    // If a keyword is provided, filter the results
    // Otherwise, return all saved keywords
    const filter = keyword ? { keyword } : {};

    const keywords = await KeywordTrend.find(filter).sort({ searchDate: -1 }); // Sort by searchDate in descending order

    if (!keywords.length) {
      res.status(404).json({ message: "No keyword data found." });
      return;
    }

    res.json(keywords);
  } catch (error) {
    console.error("Error fetching keywords:", error);
    res.status(500).json({ error: "Failed to retrieve keyword data." });
  }
};

export const deleteKeywordTrends = async (req: Request, res: Response): Promise<void> => {
  try {
    const { keyword } = req.query;

    if (!keyword) {
      res.status(400).json({ error: "Keyword is required for deletion." });
      return;
    }

    const result = await KeywordTrend.deleteMany({ keyword });

    if (result.deletedCount === 0) {
      res.status(404).json({ message: "No matching keyword data found to delete." });
      return;
    }

    res.json({ message: `${result.deletedCount} record(s) deleted for keyword: ${keyword}` });
  } catch (error) {
    console.error("Error deleting keyword data:", error);
    res.status(500).json({ error: "Failed to delete keyword data." });
  }
};
```

Kuva 6. Kuvakaappaus getSavedKeywords- ja deleteKeywordTrends-funktioista

Kuvassa 6 nähdään getSavedKeywords-funktio, jolla suoritetaan tallennettujen trenditietojen haku tietokannasta. Funktio hakee pääsääntöisesti kaikki tallennetut tiedot, mutta voi myös ottaa vastaan käyttäjän antaman hakusanan, jolloin hakutulokset suodatetaan annetun hakusanan mukaan. Hakutulokset järjestetään laskevaan järjestykseen, jolloin uusin hakutulos näkyy listan kärjessä. Hakutietojen poistamista varten luotu deleteKeywordTrends-funktio ottaa vastaan käyttäjän syöttämän hakusanan, jonka mukaan tietokannasta poistetaan samanniminen trenditieto, mikäli tällainen tietokannasta löytyy.

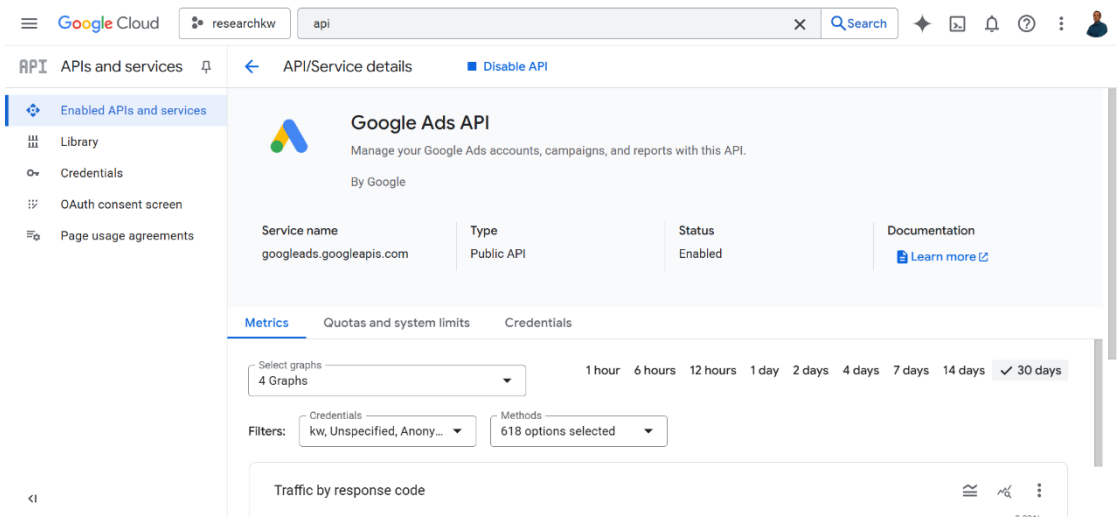
4.3 Google Ads Keyword Planner -ominaisuuden kehittäminen

Google Ads Keyword Planner -ominaisuus integroi virallisen Google Ads API:n hakemaan käyttäjän antamaan syötteeseen liittyviä hakusanoja ja niihin liittyviä mittareita, kuten hakumäärää ja kilpailua. Tämä ominaisuus tuo sovellukseen tietoon perustuvat avainsanaehdotukset, joita tukevat todelliset mainostilastot. Tavoitteena on tarjota käyttäjälle hakusanaehdotuksia, jotka perustuvat käyttäjän syöttämään hakusanaan. Hakusanaehdotukset sisältävät myös kuukausittaiset keskiarvot hakumääristä ja kilpailun tasosta, jotka on jaoteltu karkeasti low-, medium- ja high-luokkiin. Nämä tiedot auttavat käyttäjää priorisoimaan hakusanoja trendin lisäksi myös todellisen haku- ja mainostiedon perusteella.

Google Ads API:n asennus ja autentikointi

Google Ads API -rajapinnan integraatio sovellukseen on huomattavasti työläämpi verrattuna SerpAPI:n tai OpenAI API:n integraatioon. Google Ads API:n käyttäminen edellyttää seuraavien avainten hankkimista:

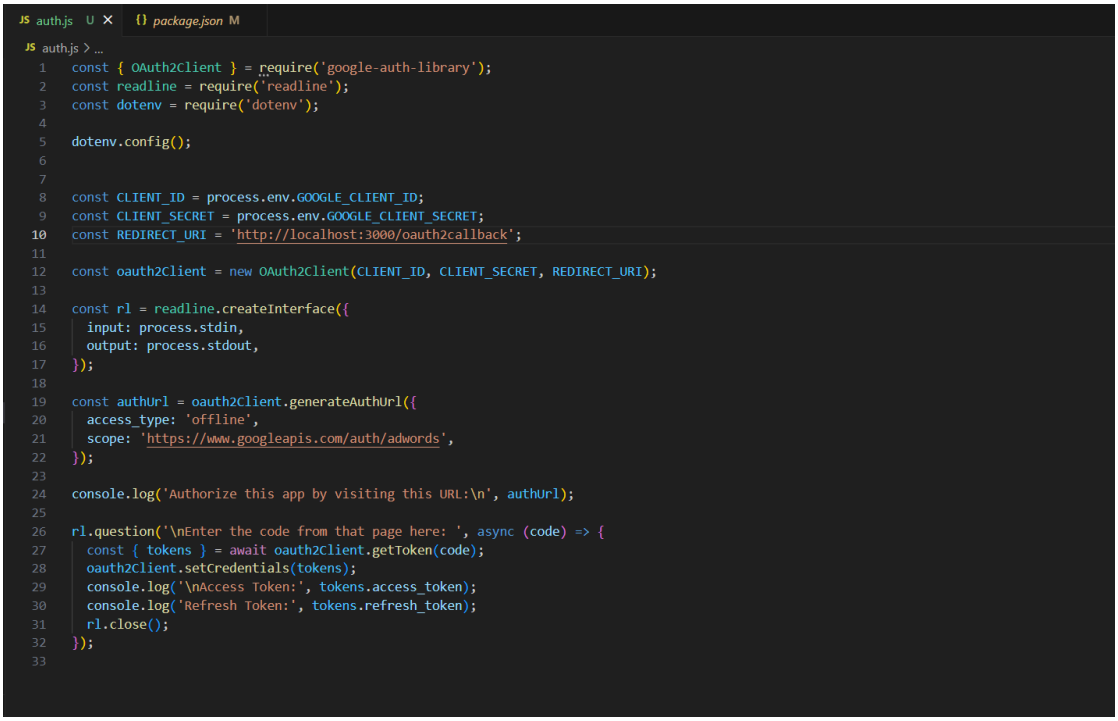
- Google Ads Developer Token
- Login Customer ID (manageritili)
- Customer ID (testi- tai asiakastili)
- OAuth2 standardin mukaisen autentikoinnin ja access tokenin.



Kuva 7. Kuvakaappaus Googlen API-kirjastosta

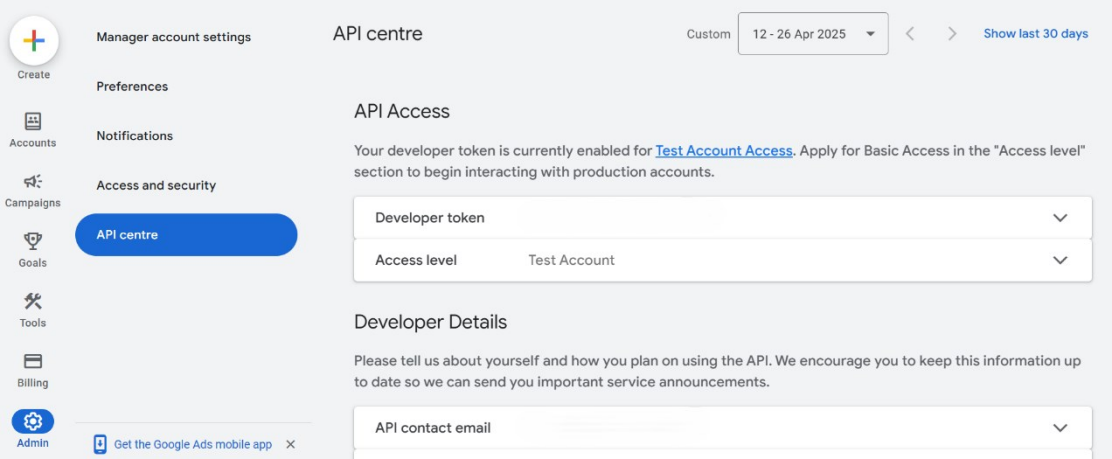
Ensimmäisenä kirjauduin Google Cloud -palveluun, jossa loin sovellukselle oman projektin researchkw-nimellä. Siirryttyäni juuri luotuun projektiin, etsin

Googlen API-kirjastosta kuvan 7 mukaisen Google Ads API:n ja otin sen käyttöön projektissa. Seuraava vaihe oli luoda valtuutus (engl. credentials) Google Ads API:n käyttöön. Valtuutuksen myötä sain Client ID- ja Client secret-avaimet, joita tarvitaan OAuth2-autentikoinnissa.



Kuva 8. Kuvakaappaus auth.js-tiedostosta

Autentikointia varten asensin google-auth-library-kirjaston ja loin kuvan 8 mukaisen auth.js-tiedoston, johon lisäsin Google Cloud -palvelusta saamani Client ID- ja Client secret -avaimet ympäristömuuttujina. Suoritin auth.js-tiedoston koodieditorini komentorivillä, jolloin komentorivi palautti URL-osoitteen, joka ohjaa valtuutussivustolle. Annoin sovelluksellenI valtuuden käyttää Google Ads API -palvelua Google-tilini kautta, jolloin sain komentoriviini näkyviin Access Token- ja Refresh Token -avaimet, jotka tallensin myös ympäristömuuttujina .env-tiedostoon.



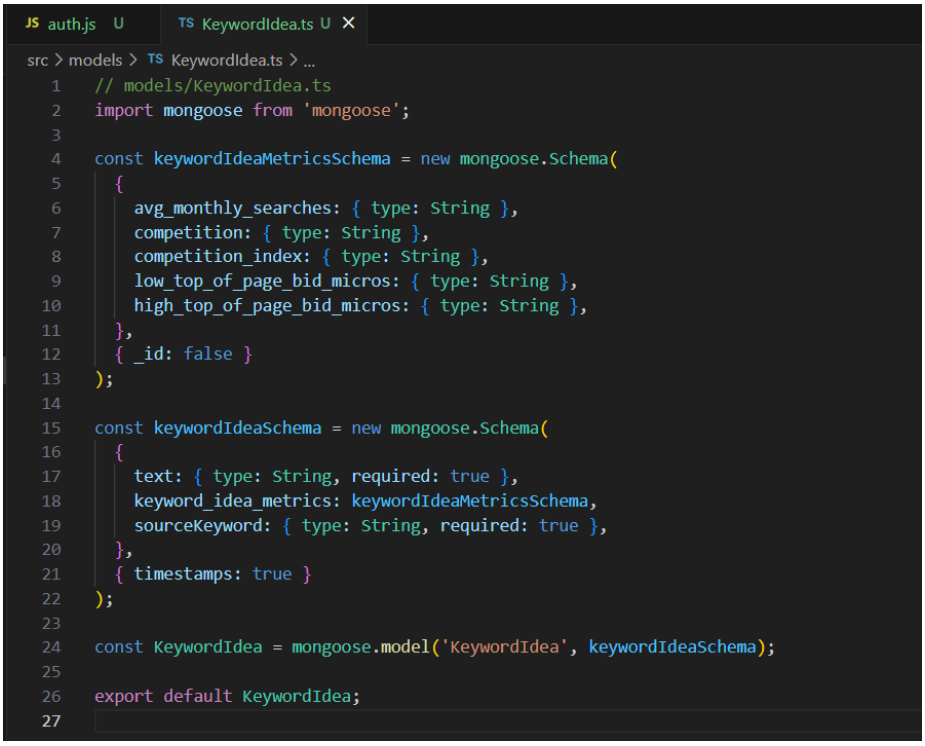
Kuva 9. Kuvakaappaus API centre -sivusta

Integraatioon tarvittavaa Developer Token -avainta varten tulee olla käytössä Google Ads -palvelun manageritili, jonka Admin -valikosta löytyy kuvan 9 mukainen API centre -sivu. Kyseiseltä sivulta kopioin 22-merkkisen Developer Token -avaimen, jonka liitin yhdeksi ympäristömuuttujaksi muiden joukkoon .env-tiedostoon. Ennen kehitystyön jatkamista tarvitsin myös testitilit Google Ads -palveluun sekä manageritasolla että asiakastasolla. Testitilien luonnin jälkeen kopioin manageritilin tilinumeron, jonka tallensin LOGIN_CUSTOMER_ID-ympäristömuuttujaksi. Asiakastilin tilinumeron tallensin CUSTOMER_ID-ympäristömuuttujaksi. Tämän jälkeen sovelluksellani on käytössä kaikki tarvittavat avaimet Google Ads API:n käyttämiseen.

Palvelinpuolen kehitys Google Ads API- integraatiolle

Aiemman SerpAPI -integraation tapaan myös Google Ads API -integraatio aloitettiin luomalla mongoosella hakusanaehdotuksille ja hakusanaehdotuksen mittareille omat tietomallit. Tietomalli nimeltään keywordIdeaMetricsSchema on alidokumentti, johon tallennetaan hakusanaehdotuksen mittareita, kuten:

- *avg_monthly_searches*, joka kertoo keskimääräisen kuukausittaisen hakumäärän
- *competition*, joka ilmoittaa kilpailun tason karkealla asteikolla LOW, MEDIUM ja HIGH
- *competition_index*, joka kertoo kilpailuntason lukuarvona asteikolla 0–100
- *low_top_of_page_bid_micros*, joka kertoo mainostajien tarjoaman hinnan alarajan hakusanamainonnassa
- *high_top_of_page_bid_micros*, joka kertoo mainostajien tarjoaman hinnan ylärajan hakusanamainonnassa.



Kuva 10. Kuvakaappaus KeywordIdea.ts-tiedostosta

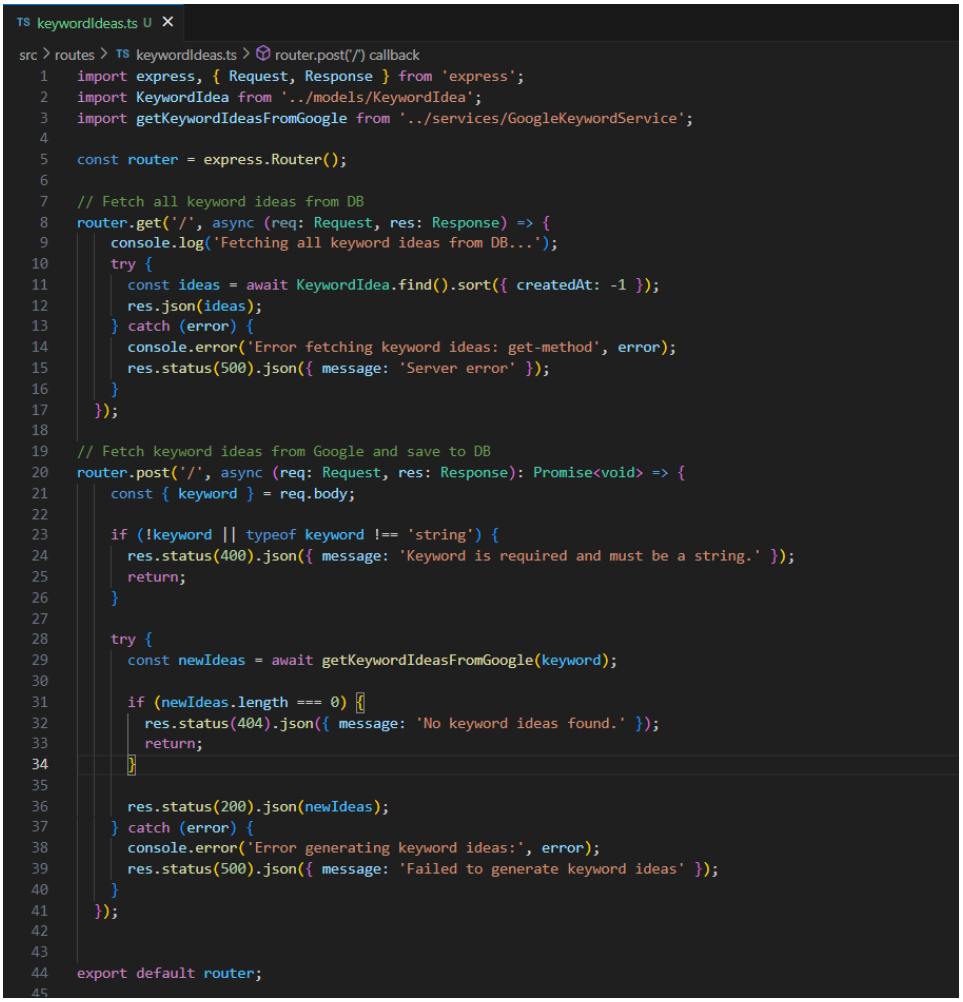
Hintatiedot ilmoitetaan mikroina, joista saadaan mainostiliin määritetyn valuutan mukainen hinta jakamalla mikroina ilmoitettu arvo 1 000 000. Hakusanaehdotuksissa mittareiden saatavuus vaihtelee hakusanaehdotuksen suosion mukaan. Kuvan 10 keywordIdeaSchema-tietomalli toimii varsinaisena päädokumenttina ja se sisältää keywordIdeaMetricsScheman lisäksi *text*-kentän, joka sisältää itse hakusanaehdotuksen sekä *source_keyword*-kentän, joka sisältää ehdotuksen lähteenä toimineen hakusanan.

Google Ads API -rajapintaan lähetettäviä pyyntöjä ja tietokantaan tallennusta varten loin palvelinpuolen services-kansioon tiedoston nimeltään GoogleKeywordService.ts. API-pyyntöjä varten asensin node-pakettina google-ads-api-kirjaston, joka tarjoaa työkalut API-pyyntöjen lähettämiseen Google Ads API -rajapintaan. Tässä vaiheessa Google Ads API:n asennuksessa ja autentikoinnissa hankitut avaimet tulevat tarpeeseen.

```
TS GoogleKeywordService.ts U ×
src > services > TS GoogleKeywordService.ts > getKeywordIdeasFromGoogle
1  import { GoogleAdsApi } from 'google-ads-api';
2  import KeywordIdea from '../models/KeywordIdea';
3  import dotenv from 'dotenv';
4  dotenv.config();
5
6  const client = new GoogleAdsApi({
7    client_id: process.env.GOOGLE_CLIENT_ID!,
8    client_secret: process.env.GOOGLE_CLIENT_SECRET!,
9    developer_token: process.env.GOOGLE_DEVELOPER_TOKEN!,
10 });
11
12 const customer = client.Customer({
13   customer_id: process.env.CUSTOMER_ID!,
14   login_customer_id: process.env.LOGIN_CUSTOMER_ID!,
15   refresh_token: process.env.GOOGLE_REFRESH_TOKEN!,
16 });
17
```

Kuva 11. Kuvakaappaus GoogleKeywordService.ts-tiedostosta

Kuvassa 11 luodaan client-objekti, joka sisältää Google Cloud -palvelusta kopioidut Client ID- ja Client secret -avaimet. Lisäksi mukaan liitetään Google Ads -manageritilin Developer Token. Customer-objektiin liitetään testimainostilien tilinumerot sekä autentikoinnista saatu GOOGLE_REFRESH_TOKEN.



```
TS keywordIdeas.ts U ×
src > routes > TS keywordIdeas.ts > router.post('/') callback
1  import express, { Request, Response } from 'express';
2  import KeywordIdea from '../models/KeywordIdea';
3  import getKeywordIdeasFromGoogle from '../services/GoogleKeywordService';
4
5  const router = express.Router();
6
7  // Fetch all keyword ideas from DB
8  router.get('/', async (req: Request, res: Response) => {
9      console.log('Fetching all keyword ideas from DB...');
10     try {
11       const ideas = await KeywordIdea.find().sort({ createdAt: -1 });
12       res.json(ideas);
13     } catch (error) {
14       console.error('Error fetching keyword ideas: get-method', error);
15       res.status(500).json({ message: 'Server error' });
16     }
17   });
18
19 // Fetch keyword ideas from Google and save to DB
20 router.post('/', async (req: Request, res: Response): Promise<void> => {
21     const { keyword } = req.body;
22
23     if (!keyword || typeof keyword !== 'string') {
24       res.status(400).json({ message: 'Keyword is required and must be a string.' });
25       return;
26     }
27
28     try {
29       const newIdeas = await getKeywordIdeasFromGoogle(keyword);
30
31       if (newIdeas.length === 0) {
32         res.status(404).json({ message: 'No keyword ideas found.' });
33         return;
34       }
35
36       res.status(200).json(newIdeas);
37     } catch (error) {
38       console.error('Error generating keyword ideas:', error);
39       res.status(500).json({ message: 'Failed to generate keyword ideas' });
40     }
41   });
42
43
44 export default router;
45
```

Kuva 12. Kuvakaappaus keywordIdeas.ts-tiedostosta

Projektin routes-kansioon luotiin tiedosto nimeltään keywordIdeas.ts, joka näkyy kuvassa 12. Tämä tiedosto sisältää reitit hakusanaehdotuksia varten. Tiedoston GET-reitti hakee tallennetut hakusanaehdotukset tietokannasta ja POST-reitti ottaa vastaan käyttäjän syöttämän hakusanan ja kutsuu getKeywordIdeasFromGoogle-funktiota, joka saa parametrinä käyttäjän antaman hakusanan.

```
17
18  async function getKeywordIdeasFromGoogle(seedKeyword: string): Promise<any[]> {
19    const rawIdeas: any = await customer.keywordPlanIdeas.generateKeywordIdeas({
20      customer_id: process.env.CUSTOMER_ID!,
21      keyword_seed: { keywords: [seedKeyword] },
22      geo_target_constants: ['geoTargetConstants/2840'], // United States
23      language: 'languageConstants/1000', // English
24      keyword_plan_network: 'GOOGLE_SEARCH_AND_PARTNERS',
25      include_adult_keywords: false,
26      page_size: 10,
27      page_token: '',
28      keyword_annotation: [],
29      toJSON: () => ({}),
30    });
31
32    const ideas = Array.isArray(rawIdeas) ? rawIdeas : [];
33
34    // Format the ideas and prepare them for saving in MongoDB
35    const formattedIdeas = ideas.map((idea: any) => ({
36      text: idea.text,
37      keyword_idea_metrics: idea.keyword_idea_metrics,
38      sourceKeyword: seedKeyword,
39    }));
40
41    // Save the formatted ideas to MongoDB and return the saved entries
42    const savedIdeas = await KeywordIdea.insertMany(formattedIdeas);
43    return savedIdeas;
44  }
45
46  export default getKeywordIdeasFromGoogle;
47
```

Kuva 13. Kuvakaappaus getKeywordIdeasFromGoogle-funktiosta

Kuvassa 13 esiintyvä getKeywordIdeasFromGoogle-funktion sisällä kutsutaan google-ads-api-kirjaston generateKeywordIdeas-funktiota, jolle annetaan oleellisimpina parametreinä asiakkaan mainostilin tilinumero, hakusivujen määrä ja käyttäjän antama lähteenä toimiva hakusana. Parametriin *geo_target_constants* määritetään Google Ads -palvelun käyttämä maakoodi, jos haku kohdentaa tiettyyn maahan. Tässä tapauksessa haku kohdistuu Yhdysvaltoihin ja kielivalinta on englanti. Parametriin *keyword_plan_network* määritetään verkostot, joihin hakusanasuunnitelma kohdistuu. Rajoittamattomalla haulla Google Ads API saattaa palauttaa niin suuren määrän dataa, että koko listaus ei mahdu yhteen API-pyynnön vastaukseen. Tässä tapauksessa vastaus jaetaan useampaan osaan ja parametri *page_token* toimii sivumerkkinä seuraavaa palautusta varten. Parametri *keyword_annotation* antaa mahdollisuuden liittää hakuun merkintöjä, jotka auttavat ymmärtämään haun kontekstin. Merkinnät voivat auttaa ymmärtämään, jos haku liittyy johonkin tiettyyn

tuoteryhmään tai toimialaan. Lopuksi vastaus muunnetaan JSON-muotoon. Funktio getKeywordIdeasFromGoogle suorittaa myös vastauksen iteroinnin KeywordIdea-tietomallin mukaiseen muotoon ja tallentaa sen tietokannan keywords-kokoelmaan.

4.4 OpenAI API -integraatio

Hakusanatutkimustyökalun tekoälymoduuli integroidaan OpenAI:n GPT-4o-tekoälymalli tuottamaan luovia hakusanaehdotuksia ja pitkän hännän hakusanoja Google Trends- ja Google Ads Keyword Planner -työkalujen rinnalle. GPT-4o on tämän opinnäytetyön julkaisuhetkellä OpenAI:n uusin kielimalli ja pystyy tunnistamaan eri vivahteita puhutussa ja kirjoitetussa kielessä. Tämän takia kielimalli pystyy tarjoamaan hakusanaehdotuksiin tuoreita keskustelufraaseja ja termejä, jotka eivät välttämättä vielä näy hakuvolyymitilastoissa.

TS AISuggestion.ts U

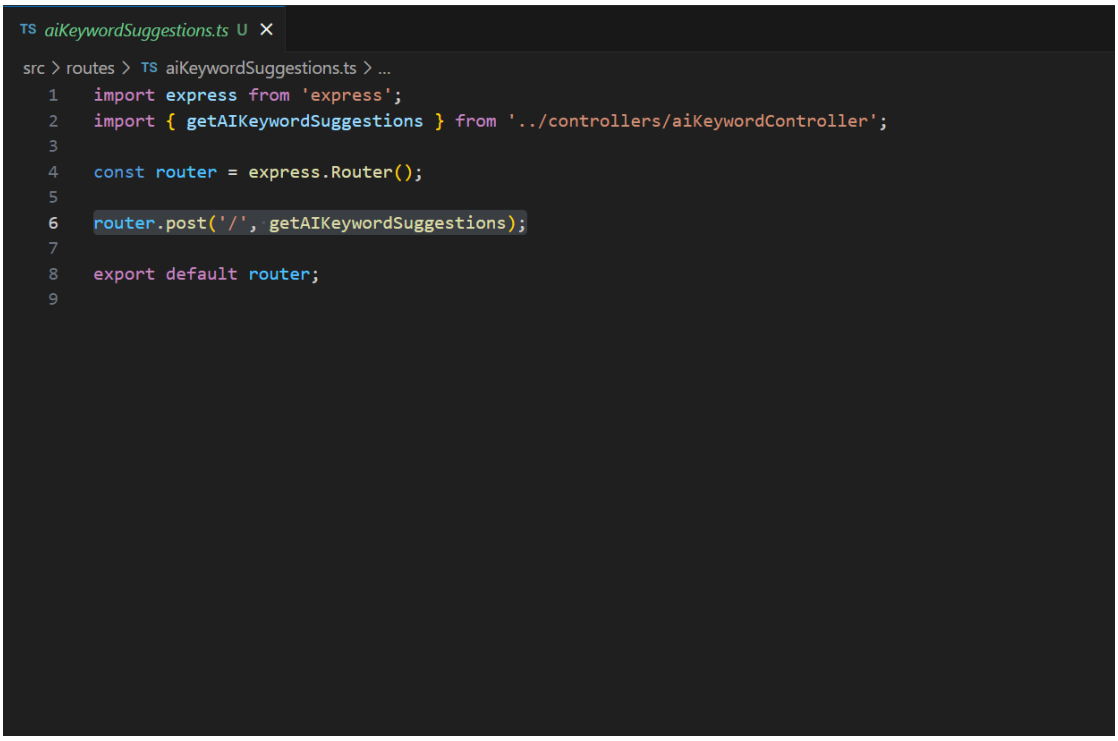
src > models > TS AISuggestion.ts > ...

```ts
import mongoose, { Schema, Document } from 'mongoose';

interface AISuggestion extends Document {
  keyword: string;
  suggestions: string[];
  createdAt: Date;
}

const AISuggestionSchema: Schema = new Schema({
  keyword: { type: String, required: true },
  suggestions: { type: [String], required: true },
  createdAt: { type: Date, default: Date.now }
});

export default mongoose.model<AISuggestion>('AISuggestion', AISuggestionSchema);

```

Kuva 14. Kuvakaappaus AISuggestions.ts-tiedostosta

Kehitystyö aloitettiin asentamalla OpenAI:n SDK-paketti projektiin. Lisäsin myös OpenAI:n sivustolta hankitun API-avaimen ympäristömuuttujaksi .env-tiedostoon. Seuraavaksi tekoälyn ehdotuksille luotiin oma skeema models-kansion AISuggestions.ts-tiedostoon, joka näkyy kuvassa 14. AISuggestionSchema-tietomalli sisältää *keyword*-, *suggestions*- ja *createdAt*-kentät. Tietomallissa *keyword*-kenttään tallennetaan käyttäjän antama hakusana ja *sug-*

gestions-kenttään tekoälyn tuottamat hakusanaehdotukset. Päivämäärä tallennetaan *createdAt*-kenttään, johon määritetään oletuksena dokumentin luontihetki.



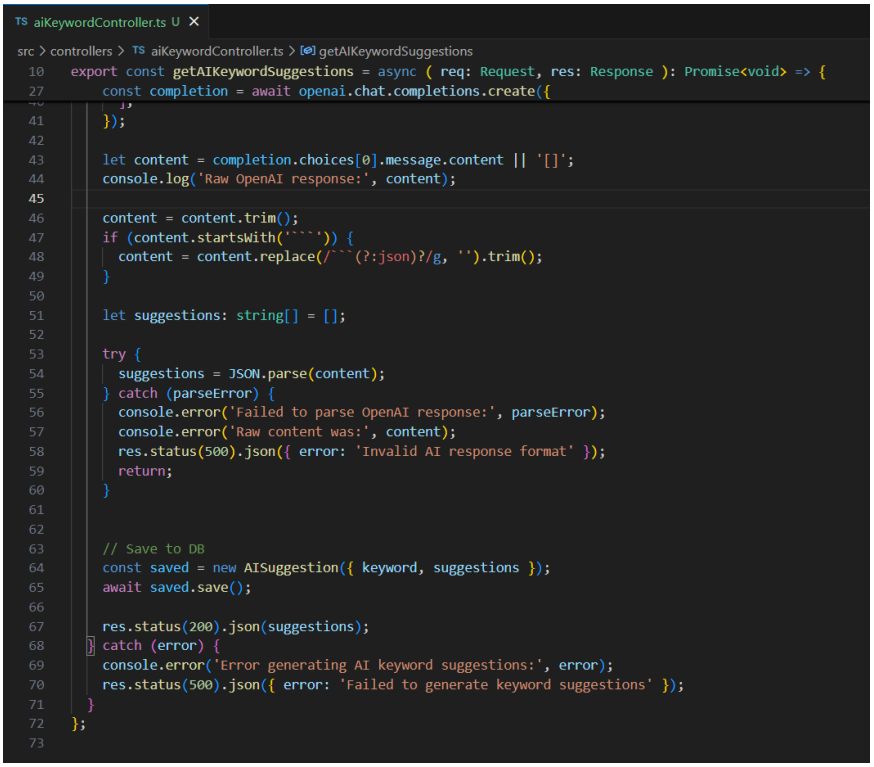
Kuva 15. Kuvakaappaus aiKeywordSuggestions.ts-tiedostosta

Tekoälymoduulille on rakennettu yksi POST-reitti routes-kansion aiKeywordSuggestions.ts-tiedostoon, joka näkyy kuvassa 15. Reitti kutsuu getAIKeywordSuggestions-funktiota, joka löytyy controllers-kansion aiKeywordController.ts-tiedostosta.

```
TS aiKeywordController.ts U ×
src > controllers > TS aiKeywordController.ts > getAIKeywordSuggestions
1   import { Request, Response } from 'express';
2   import { OpenAI } from 'openai';
3   import dotenv from 'dotenv';
4   import AISuggestion from '../models/AISuggestion';
5
6   dotenv.config();
7
8   const openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });
9
10  export const getAIKeywordSuggestions = async ( req: Request, res: Response ): Promise<void> => {
11    const { keyword } = req.body;
12
13    if (!keyword || typeof keyword !== 'string') {
14      res.status(400).json({ error: 'Keyword is required and must be a string' });
15      return;
16    }
17
18    try {
19      // Check if suggestions already exist
20      const existing = await AISuggestion.findOne({ keyword });
21      if (existing) {
22        res.status(200).json(existing.suggestions);
23        return;
24      }
25
26      // Call OpenAI
27      const completion = await openai.chat.completions.create({
28        model: 'gpt-4o',
29        messages: [
30          {
31            role: 'system',
32            content:
33              'You are an SEO expert who helps generate keyword and long-tail keyword suggestions.',
34          },
35          {
36            role: 'user',
37            content: `Return ONLY a valid JSON array of 10 keyword or long-tail keyword suggestions related to:
38            "${keyword}".`,
39          },
40        ],
41      });
42
43      let content = completion.choices[0].message.content || '[]';
44      console.log('Raw OpenAI response:', content);
```

Kuva 16. Kuvakaappaus aiKeywordController.ts-tiedostosta

Kuvan 16 aiKeywordController.ts-tiedostossa luodaan ensin openai-olio, jolle annetaan parametrina OpenAI:n API-avain. Funktion sisällä tehdään ensin haku tietokantaan ja tarkistetaan, jos käyttäjän antamalle hakusanalle löytyy jo valmiit hakusanaehdotukset. Mikäli hakusanaehdotuksia ei löydy, suoritetaan API-pyyntö OpenAI:lle hakusanaehdotuksia varten. Kuvassa 16 näkyvä completion-muuttuja muodostaa API-pyyntöön ns. keskustelun, jossa GPT-4o kielimallille lähetetään viesti, jossa on määritelty kielimallin rooli ja käyttäjän kehote. Kielimallin rooli on toimia SEO-asiantuntijana ja tuottaa hakusanaehdotuksia ja pitkän hännän hakusanoja käyttäjän antamasta hakusanasta. Käyttäjän kehotteessa määritellään kielimalli tuottamaan 10 hakusanaehdotusta tai pitkän hännän hakusanaa ja palauttamaan lista ainoastaan validina JSON-tyypin listana.



Kuva 17. Kuvakaappaus OpenAI:n vastauksen esikäsittelystä ja tallennuksesta

OpenAI:n palauttama vastaus ei kuitenkaan välttämättä ole pyynnössä esitetty puhdas merkkijonojen JSON-listaus, vaan siinä saattaa esiintyä takaviivoja ja Markdown-muotoilua, joten funktioon on lisätty esikäsittelyvaihe näiden poistamiseksi, joka näkyy kuvassa 17. Esikäsittelyn jälkeen hakusanaehdotukset tallennetaan tietokantaan.

5 KÄYTTÖLIITTYMÄN KEHITYS

Palvelinpuolen toimintojen ja moduulien kehityksen jälkeen loin React-sovelluksen käyttöliittymälle. Konsolissa suoritettu komento *npx create-react-app client --template typescript* loi käyttöliittymää varten React-sovelluksen client-kansioon, joka käyttää ohjelmointikielenä TypeScriptiä. Datan visualisointia varten asensin Recharts-kirjaston, jolla voi luoda näyttäviä kaavioita käyttöliittymään. Käyttöliittymän tyylittelyä varten asennettiin Tailwind CSS-kehys sekä Axios-kirjasto pyyntöjen lähettämiseen ulkoisille palvelimille. Hakusanatutkimustyökaluun luodaan erillinen komponentti jokaista toimintoa varten, joten loin niitä varten components-kansion ja serverille lähetettäviä pyyntöjä varten api-kansion.

5.1 Trendien hakeminen ja visualisointi

Ensimmäinen kehitettävä komponentti sisältää hakukentän trenditietojen hakemiseen ja tulosten visualisoimisen kaaviolla. Tätä moduulia varten on kaksi itsenäistä komponenttia, joista toinen sisältää hakupalkin haettaville trenditiedoille ja toinen sisältää alasvetovalikon ja viivakaavion tallennetuille hakusanoille.

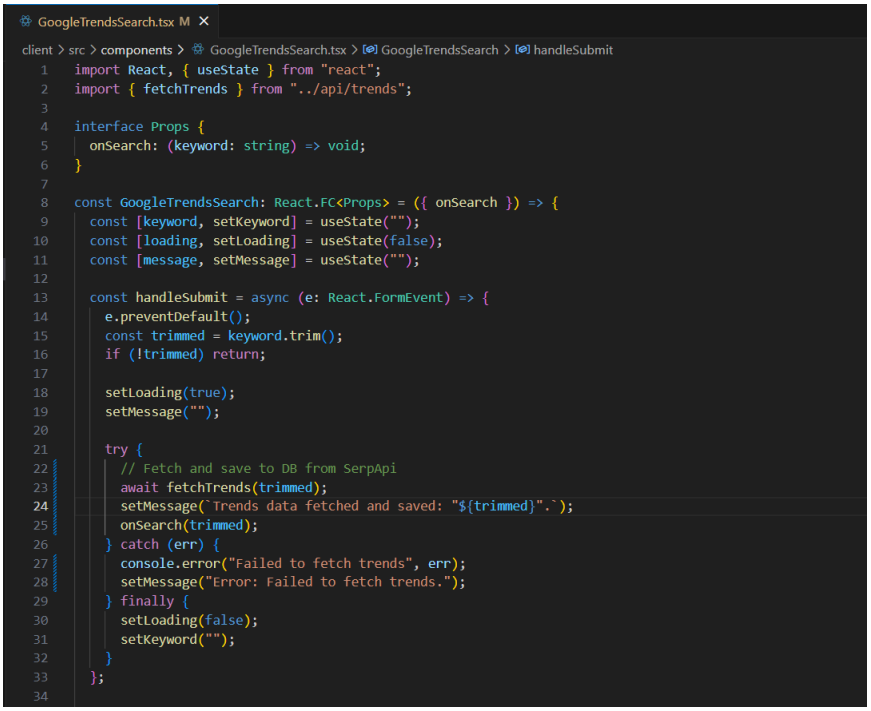
KeywordTrendsChart.tsx M trends.ts M

client > src > api > trends.ts > ...

```ts
import axios from 'axios';

const BASE_URL = process.env.REACT_APP_API_URL;

export const fetchTrends = async (keyword: string) => {
  const response = await axios.get(`${BASE_URL}/keywords/trends`, {
    params: { keyword },
  });
  return response.data;
};

export const fetchSavedKeywords = async () => {
  const response = await axios.get(`${BASE_URL}/keywords`);
  return response.data;
};

export const deleteKeyword = async (keyword: string) => {
  const response = await axios.delete(`${BASE_URL}/keywords/trends`, {
    params: { keyword },
  });
  return response.data;
};

export {};
```

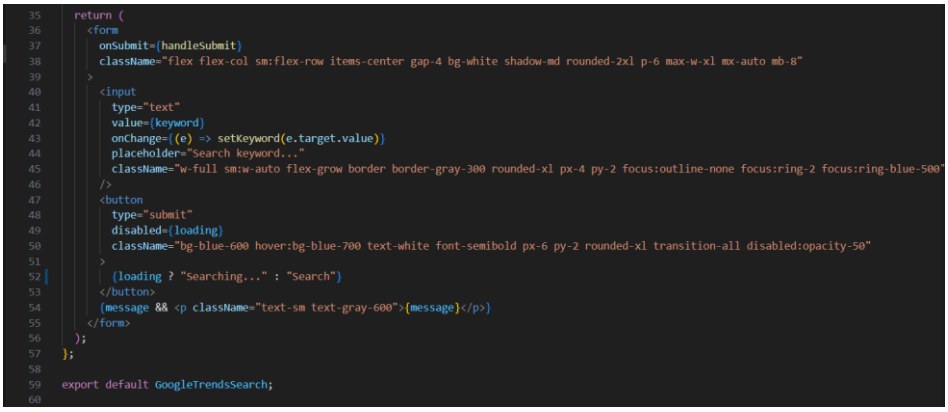
Kuva 18. Kuvakaappaus trends.ts-tiedostosta

Kuvan 18 trends.ts-tiedosto sisältää trenditietojen API-pyyntöjen reitit. BASE_URL-muuttuja sisältää palvelinpuolen URL-osoitteen ympäristömuuttujana. Funktio fetchTrends hakee hakusanalla SerpAPI:n kautta trenditiedot. Funktio fetchSavedKeywords hakee tallennetut hakusanat tietokannasta ja deleteKeyword-funktio poistaa valitun hakusanan.



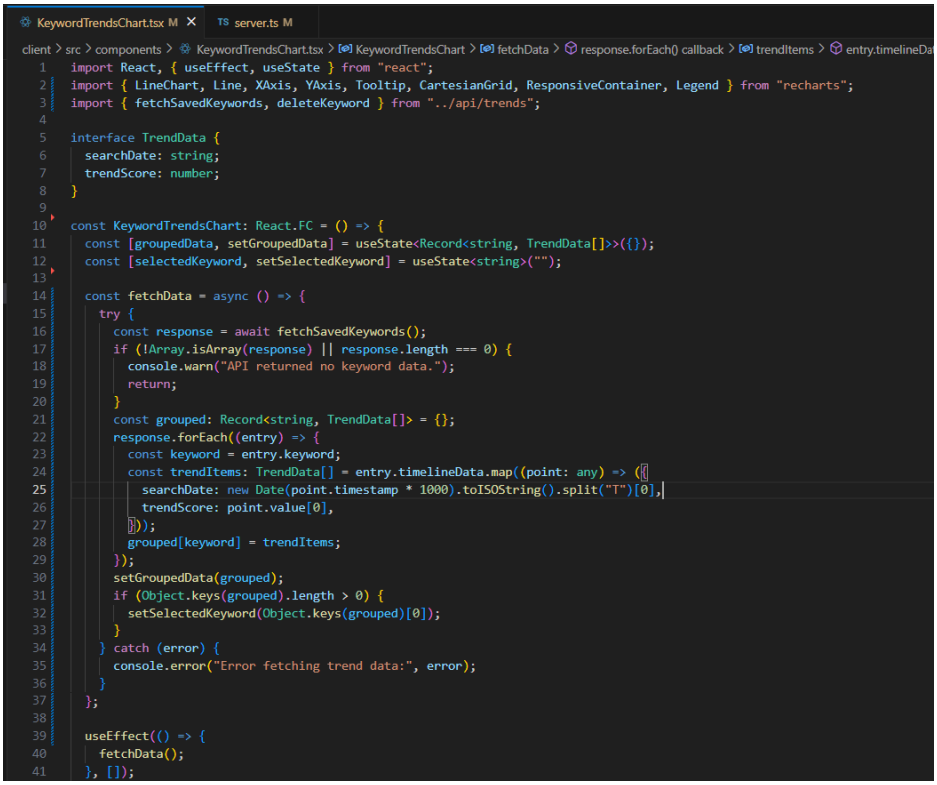
Kuva 19. Kuvakaappaus GoogleTrendSearch.tsx-tiedostosta ja handleSubmit-funktiosta

Käyttöliittymän ensimmäinen komponentti on hakulomake, johon käyttäjä kirjoittaa hakusanan trenditietojen hakemista varten. Käyttäjän painaessa search-painiketta suoritetaan kuvan 19 handleSubmit-funktio, jonka sisällä kutsutaan aiemmin esiteltyä fetchTrends-funktiota.



Kuva 20. Kuvakaappaus trenditietojen hakulomakkeesta

Kuvan 20 hakulomake on tyylitelty Tailwind CSS:n avulla. Sama tyyliä sovelletaan myös muihin käyttöliittymän komponentteihin. Hakulomake sisältää tekstikentän ja painikkeen.

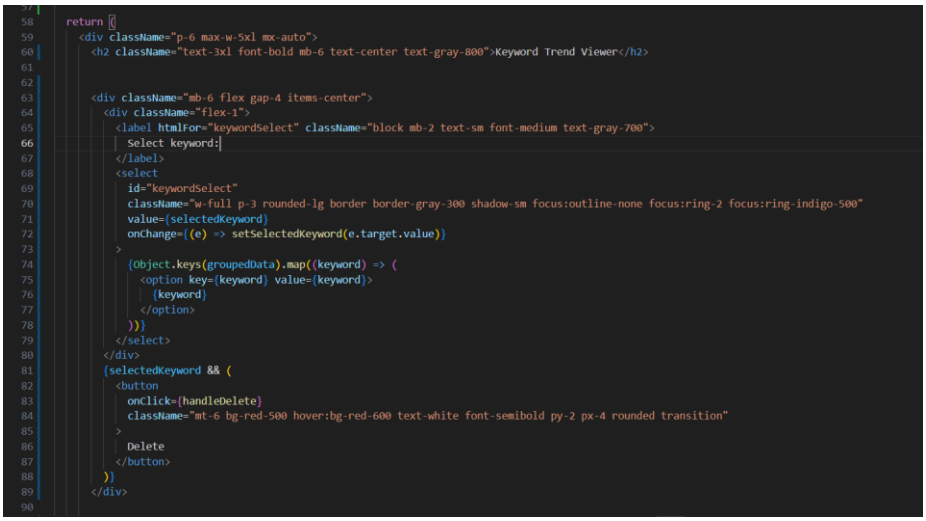


Kuva 21. Kuvakaappaus KeywordTrendsChart.tsx-tiedostosta

Kuvassa 21 on KeywordTrendsChart.tsx-tiedostossa oleva fetchData-funktio, joka hakee tallennetut hakusanat trenditietoineen tietokannasta kutsumalla fetchSavedKeywords-funktiota. Tietokannasta haettujen hakusanojen trenditiedot iteroidaan TrendData-tyypin mukaiseen muotoon. Esikäsittelyn jälkeen tietokannasta haetut hakusanat trenditietoineen sijoitetaan groupedData-tilamuuttujaan. Tallennetujen hakusanojen hakeminen tietokannasta tapahtuu automaattisesti käyttöliittymän avaamisen tai päivittämisen yhteydessä, koska fetchData-funktio on sijoitettu Reactin useEffect-funktion sisään.

Kuva 22. Kuvakaappaus handleDelete-funktiosta

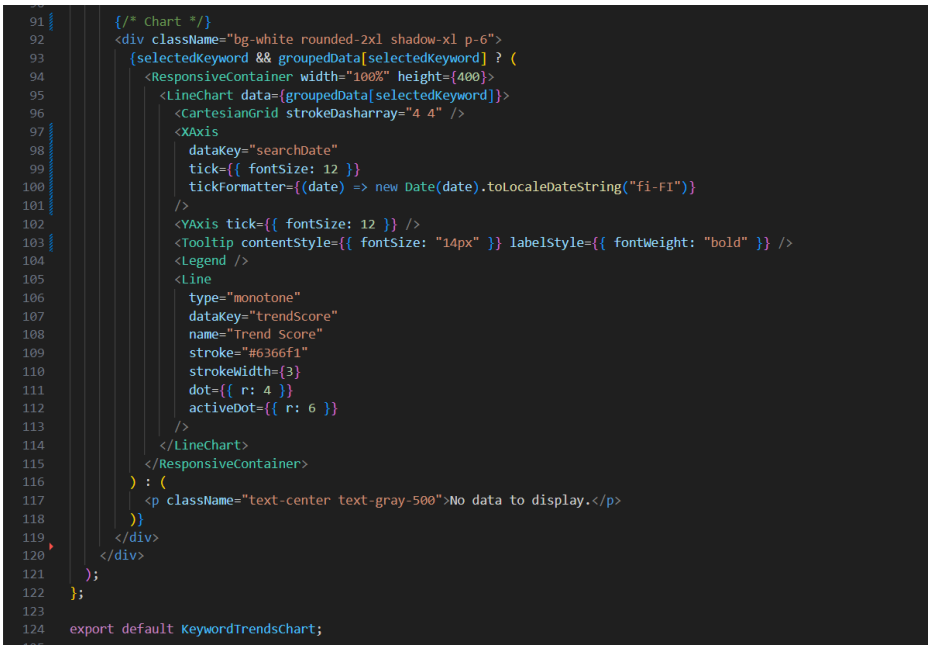
Kuvan 22 handleDelete-funktio poistaa käyttöliittymän alasvetovalikossa valittuna olevan hakusanan tietokannasta kutsumalla deleteKeyword-funktiota.



```
return (
  <div className="p-6 max-w-5xl mx-auto">
    <h2 className="text-3xl font-bold mb-6 text-center text-gray-800">Keyword Trend Viewer</h2>


    <div className="mb-6 flex gap-4 items-center">
      <div className="flex-1">
        <label htmlFor="keywordSelect" className="block mb-2 text-sm font-medium text-gray-700">
          Select keyword:
        </label>
        <select
          id="keywordSelect"
          className="w-full p-3 rounded-lg border border-gray-300 shadow-sm focus:outline-none focus:ring-2 focus:ring-indigo-500"
          value={selectedKeyword}
          onChange={(e) => setSelectedKeyword(e.target.value)}
        >
          {Object.keys(groupedData).map((keyword) => (
            <option key={keyword} value={keyword}>
              {keyword}
            </option>
          ))}
        </select>
      </div>
      {selectedKeyword && (
        <button
          onClick={handleDelete}
          className="mt-6 bg-red-500 hover:bg-red-600 text-white font-semibold py-2 px-4 rounded transition"
        >
          Delete
        </button>
      )}
    </div>
```

Kuva 23. Kuvakaappaus alasvetovalikon rakenteesta

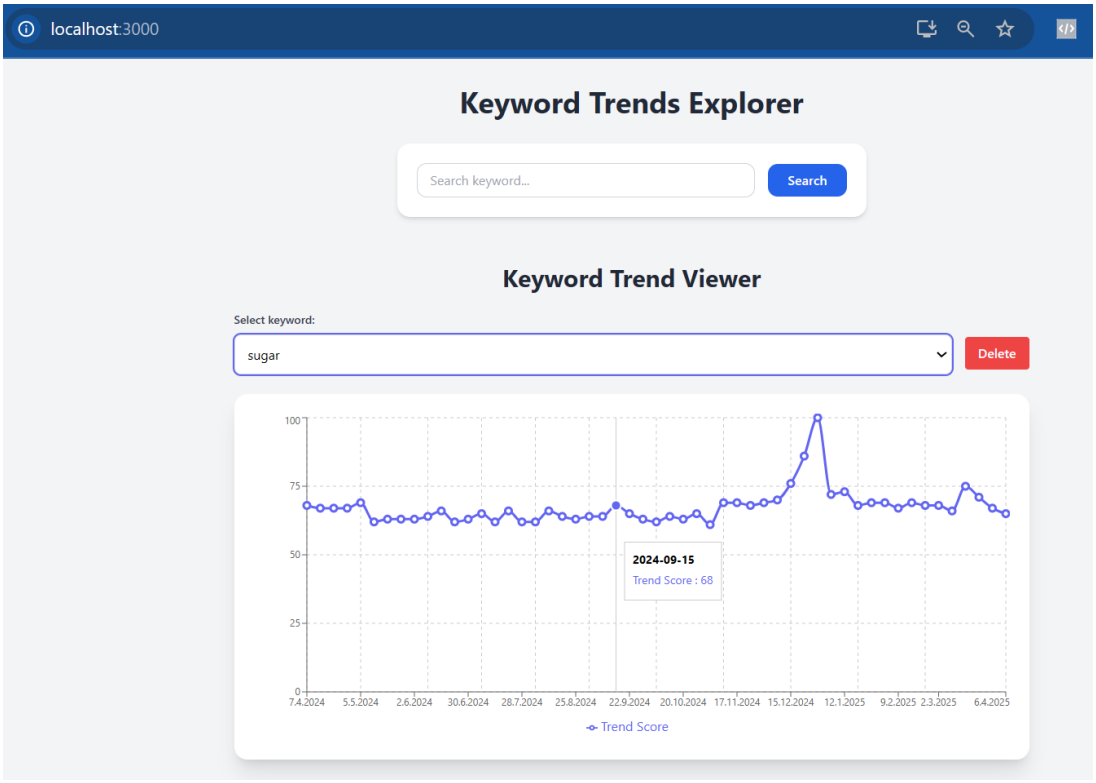
Kuvassa 23 käyttöliittymän alasvetovalikko tallennetuille hakusanoille, josta käyttäjä voi valita hakusanan, joka visualisoidaan viivakaaviolla alasvetovalikon alle.



```
      {/* Chart */}
      <div className="bg-white rounded-2xl shadow-xl p-6">
        {selectedKeyword && groupedData[selectedKeyword] ? (
          <ResponsiveContainer width="100%" height={400}>
            <LineChart data={groupedData[selectedKeyword]}>
              <CartesianGrid strokeDasharray="4 4" />
              <XAxis
                dataKey="searchDate"
                tick={{ fontSize: 12 }}
                tickFormatter={(date) => new Date(date).toLocaleDateString("fi-FI")}
              />
              <YAxis tick={{ fontSize: 12 }} />
              <Tooltip contentStyle={{ fontSize: "14px" }} labelStyle={{ fontWeight: "bold" }} />
              <Legend />
              <Line
                type="monotone"
                dataKey="trendScore"
                name="Trend Score"
                stroke="#6366f1"
                strokeWidth={3}
                dot={{ r: 4 }}
                activeDot={{ r: 6 }}
              />
            </LineChart>
          </ResponsiveContainer>
        ) : (
          <p className="text-center text-gray-500">No data to display.</p>
        )}
      </div>
    </div>
  );
};

export default KeywordTrendsChart;
```

Kuva 24. Kuvakaappaus viivakaavion rakenteesta

Kuvan 24 viivakaaviolla näytetään 12 kuukauden trenditiedot viikon välein käyttäjän valitsemasta hakusanasta. Viivakaavio käyttää aiemmin asennettua Recharts-kirjastoa, jonka avulla voi luoda näyttäviä animoituja kaavioita datan visualisoimiseen.

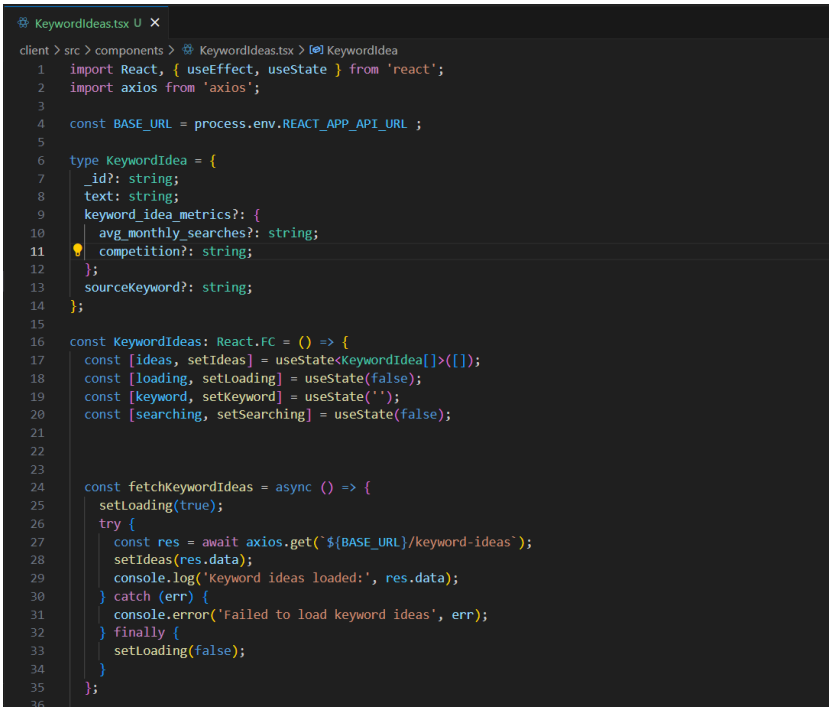


Kuva 25. Kuvakaappaus sovelluksen käyttöliittymästä

Kuvassa 25 on kuvakaappaus, jossa näkyvät trenditietojen hakulomake, tallennettujen hakusanojen alasvetovalikko, delete-painike ja viivakaavio käyttöliittymässä. Kuvaushetkellä tallennetuista hakusanoista on valittuna sugar, ja viivakaavio näyttää tämän hakusanan trenditiedot viimeisen 12 kuukauden ajalta.

5.2 Hakusanaideoiden hakeminen ja listaus

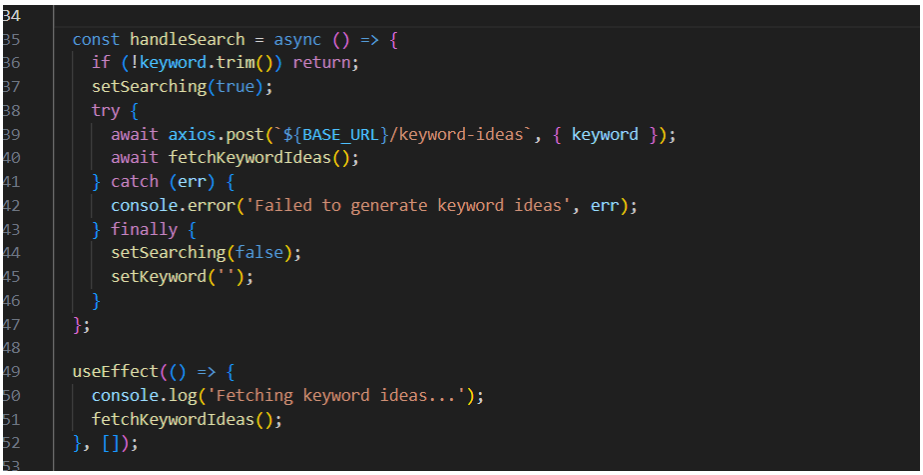
Seuraava kehitettävä moduuli oli Google hakusanaideoiden listaus Google Ads API:n kautta. Moduuli sisältää hakukentän, jonka alle listataan Googlen ehdottamat hakusanaideat kuukausittaisten hakumäärien ja kilpailutilanteen kera.



```tsx
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const BASE_URL = process.env.REACT_APP_API_URL ;

type KeywordIdea = {
  _id?: string;
  text: string;
  keyword_idea_metrics?: {
    avg_monthly_searches?: string;
    competition?: string;
  };
  sourceKeyword?: string;
};

const KeywordIdeas: React.FC = () => {
  const [ideas, setIdeas] = useState<KeywordIdea[]>([]);
  const [loading, setLoading] = useState(false);
  const [keyword, setKeyword] = useState('');
  const [searching, setSearching] = useState(false);



  const fetchKeywordIdeas = async () => {
    setLoading(true);
    try {
      const res = await axios.get(`${BASE_URL}/keyword-ideas`);
      setIdeas(res.data);
      console.log('Keyword ideas loaded:', res.data);
    } catch (err) {
      console.error('Failed to load keyword ideas', err);
    } finally {
      setLoading(false);
    }
  };
```

Kuva 26. Kuvakaappaus KeywordIdeas.tsx-tiedostosta

Kuvassa 26 on esillä KeywordIdeas.tsx-tiedosto, jonka ensimmäinen funktio hakee tietokannasta aiemmin tallennetut hakusanaehdotukset. Tämä fetchKeywordIdeas-funktio suoritetaan Reactin useEffect()-funktion sisällä, joten käyttöliittymän näkymän avaus ja päivitys suorittaa hakusanaideoiden hakemisen tietokannasta.



```tsx

  const handleSearch = async () => {
    if (!keyword.trim()) return;
    setSearching(true);
    try {
      await axios.post(`${BASE_URL}/keyword-ideas`, { keyword });
      await fetchKeywordIdeas();
    } catch (err) {
      console.error('Failed to generate keyword ideas', err);
    } finally {
      setSearching(false);
      setKeyword('');
    }
  };

  useEffect(() => {
    console.log('Fetching keyword ideas...');
    fetchKeywordIdeas();
  }, []);
```

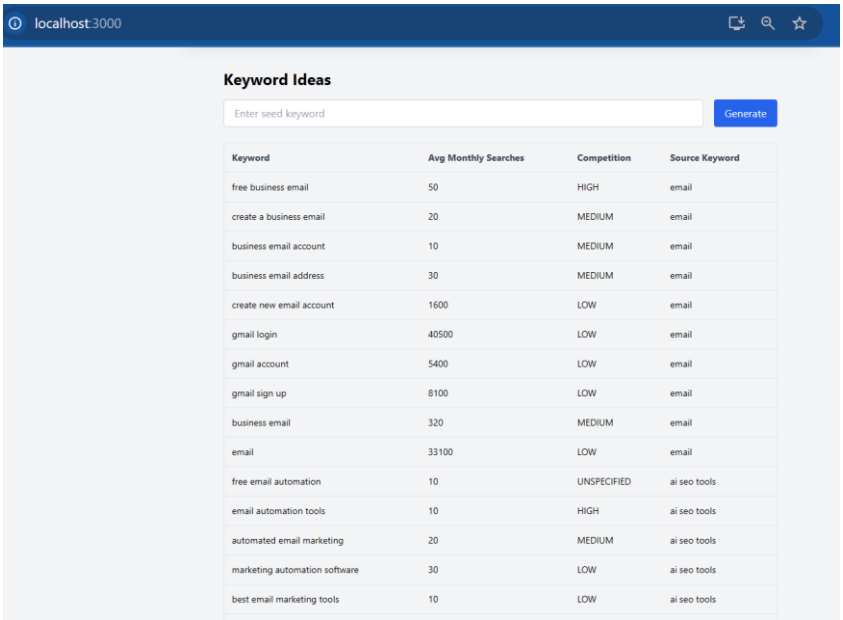
Kuva 27. Kuvakaappaus handleSearch-funktiosta

Kuvan 27 handleSearch-funktio lähettää POST-pyynnön palvelimelle hakusana parametrinään, minkä jälkeen kutsuu edellä mainittua fetchKeywordIdeas-funktiota.

```
return (
  <div className="max-w-4xl mx-auto p-6">
    <h2 className="text-2xl font-bold mb-4">Keyword Ideas</h2>
    <div className="flex gap-4 mb-6">
      <input
        type="text"
        className="flex-1 border border-gray-300 rounded px-4 py-2 focus:outline-none focus:ring-2 focus:ring-blue-500"
        placeholder="Enter seed keyword"
        value={keyword}
        onChange={(e) => setKeyword(e.target.value)}
      />
      <button
        onClick={handleSearch}
        disabled={searching}
        className="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700 transition disabled:opacity-50"
      >
        {searching ? 'Searching...' : 'Generate'}
      </button>
    </div>

    {loading ? (
      <p className="text-gray-600">Loading...</p>
    ) : ideas.length === 0 ? (
      <p className="text-gray-500">No keyword ideas found.</p>
    ) : (
      <div className="overflow-x-auto">
        <table className="w-full border-collapse border border-gray-200">
          <thead>
            <tr className="bg-gray-100 text-left text-sm text-gray-700">
              <th className="p-3 border-b">Keyword</th>
              <th className="p-3 border-b">Avg Monthly Searches</th>
              <th className="p-3 border-b">Competition</th>
              <th className="p-3 border-b">Source Keyword</th>
            </tr>
          </thead>
          <tbody>
            {ideas.map((idea) => (
              <tr key={idea._id} className="hover:bg-gray-50 text-sm">
                <td className="p-3 border-b">{idea.text}</td>
                <td className="p-3 border-b">
                  {idea.keyword_idea_metrics?.avg_monthly_searches || '-'}
                </td>
                <td className="p-3 border-b">
```

Kuva 28. Kuvakaappaus hakusanaideoiden hakulomakkeen rakenteesta

Hakusanaideoiden hakeminen tapahtuu kuvan 28 lomakkeella ja tallennetut hakusanaideat iteroidaan taulukkoon. Taulukon sarakkeisiin tulostetaan hakusana, kuukausittainen hakumäärä, kilpailutaso ja lähteenä toiminut hakusana. Sarakkeisiin olisi voinut myös lisätä hakusanan tarjoustasot, mutta testidatassa harvalla hakusanalla oli tarjoustasot saatavilla johtuen niiden vähäisistä hakumääristä. Tarvittaessa ne ovat kuitenkin helposti lisättävissä kyseiseen taulukkoon.



Kuva 29. Kuvakaappaus hakusanaideoiden lomakkeesta ja taulukosta

Kuva 29 havainnollistaa, miltä hakusanaideoiden hakulomake ja taulukko näyttävät. Kuten aiemminkin, tyyli on tehty käyttäen Tailwind CSS-kehystä. Käyttäjän kirjoittaessa hakusanan kenttään ja painamalla generate-painiketta, hakee sovellus 10 hakusanaideaa Google Ads API:n kautta.

5.3 Tekoälymoduulin lisääminen käyttöliittymään

Viimeinen moduuli, joka opinnäytetyössä lisätään käyttöliittymään, on hakutekoälyavusteisten hakusanaehdotusten haku- ja listaustoiminto. Tekoälyn toimintoja varten loin uuden komponentin nimeltään AIAssistant.tsx, joka on sijoitettuna muiden komponenttien tapaan React-sovelluksen components-kansioon.

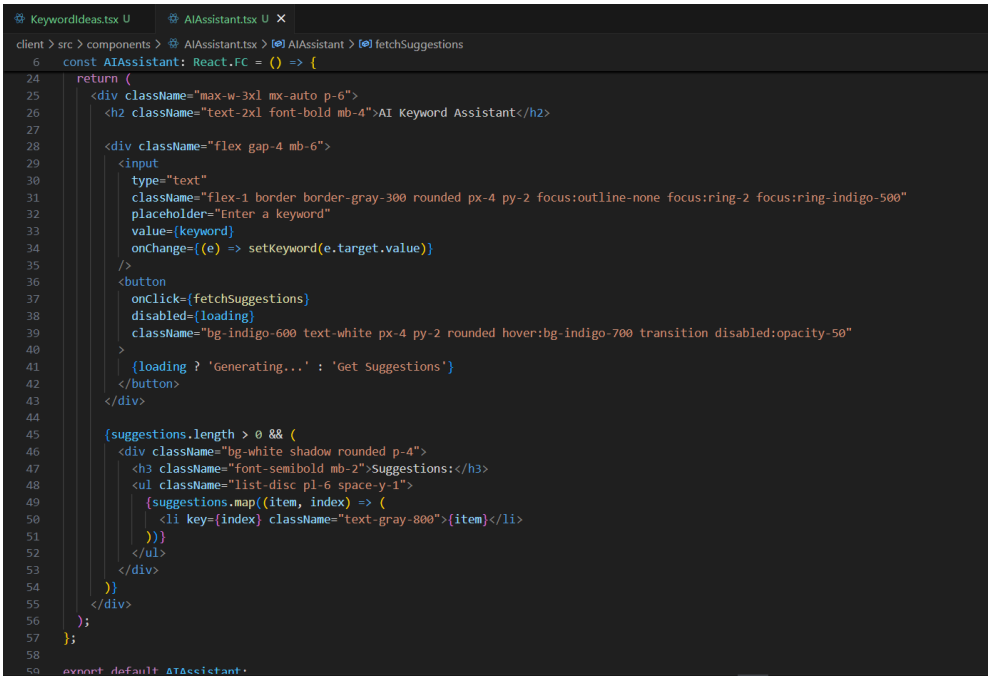
KeywordIdeas.tsx U | AIAssistant.tsx U

client > src > components > AIAssistant.tsx > AIAssistant > fetchSuggestions

```
1  import React, { useState } from 'react';
2  import axios from 'axios';
3
4  const BASE_URL = process.env.REACT_APP_API_URL;
5
6  const AIAssistant: React.FC = () => {
7    const [keyword, setKeyword] = useState('');
8    const [suggestions, setSuggestions] = useState<string[]>([]);
9    const [loading, setLoading] = useState(false);
10
11   const fetchSuggestions = async () => {
12     if (!keyword.trim()) return;
13     setLoading(true);
14     try {
15       const res = await axios.post(`${BASE_URL}/ai-keywords`, { keyword });
16       setSuggestions(res.data || []);
17     } catch (error) {
18       console.error('Error fetching suggestions:', error);
19     } finally {
20       setLoading(false);
21     }
22   };
23
```

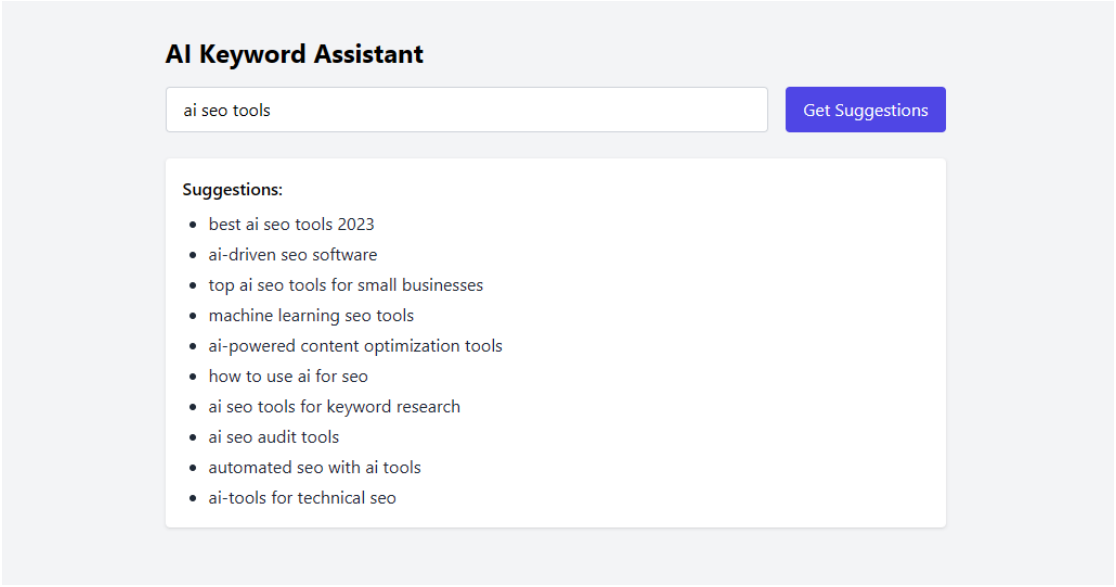
Kuva 30. Kuvakaappaus fetchSuggestions-funktiosta

Tekoälyn hakusanaehdotusten hakemiseen käytetään kuvassa 30 näkyvää fetchSuggestions-funktiota, joka lähettää POST-pyynnön palvelimelle käyttäjän antama hakusana parametrinään. Vastauksena palautuu tekoälyn tuottamat 10 hakusanaehdotusta ja pitkän hännän hakusanaa.



Kuva 31. Kuvakaappaus lomakkeen rakenteesta tekoälymoduulissa

Kuvassa 31 on tekoälyn ehdotuksia varten lomake lähteenä toimivan hakusanan kirjoittamiseen sekä painike, joka suorittaa fetchSuggestions-funktion. Tekoälyn tuottamat ehdotukset listataan lomakkeen alle iteroimalla suggestions-tilamuuttuja, johon ehdotukset on tallennettu.



Kuva 32. Kuvakaappaus tekoälymoduulista käyttöliittymässä

Kuva 32 havainnollistaa, miltä komponentti näyttää käyttöliittymässä. Hakusanalla on kirjoitettu tekstikenttään ja painettu Get Suggestions-painikketta, minkä jälkeen ehdotukset ovat tulostuneet vaalean laatikon sisään listana.

6 PÄÄTÄNTÖ

Opinnäytetyön tekemisen aikana sain vastauksen tutkimusongelmaan, joka keskittyi siihen, kuinka tekoälyavusteinen hakusanojen tutkimustyökalu voi tehostaa hakusana-analyysia. Tekoäly pystyy tuottamaan suuren määrän luovia hakusanaehdotuksia ja pitkiä hakufraaseja annetusta hakusanasta nopeassa ajassa verrattuna manuaalisesti tehtävään hakusanatutkimukseen. Sovelluksessa hakusanojen määrää oli rajoitettu OpenAI API:n ja Google Ads API:n osalta resurssien säästämiseksi, mutta todellisuudessa hakusanaideoita pystytään tuottamaan huomattavasti enemmän kuin 10 kappaletta kerrallaan. Muita rajoituksia, joita työssä havaittiin, olivat API-kiintiöt ja aineiston laajuus. Google Ads API:n-, SerpAPI- ja OpenAI API -palveluiden käyttäminen on rajoitettua ja sovelluksen toimivuus on riippuvainen näiden palveluiden jatkuvasta saatavuudesta ja maksullisista käyttöoikeuksista. Google Trends -data on myös aggregoitua, joten se ei kata tietoja tarkoista käyttäjäkohtaisista hakutavoista.

Google Ads API:n etu verrattuna tekoälyyn on se, että Googlen hakusanaehdotukset ovat erittäin luotettavia, koska ne perustuvat Googlen omaan mainostietokantaan ja todellisiin hakuvolyymeihin. Tekoälyn etuna on nopea mukautuminen nouseviin trendeihin ja niiden hyödyntäminen hakusanaehdotuksissa ja pidemmissä avainfraaseissa, joiden hakuvolyymeista ei vielä löydy tilastoitua tietoa. GPT-kielimallin hakusanaehdotuksissa saattaa kuitenkin toisinaan ilmetä täysin asiayhteyteen kuulumattomia termejä, joten on syytä käyttää omaa harkintakykyä niiden käyttämisessä. Olen kuitenkin sitä mieltä, että molemmat hakusanaideoita tuottavat palvelut ovat tarpeellisia hakusanatutkimuksessa, koska ne ominaisuuksiltaan täydentävät toisiaan. Samoin Google Trends -palvelu tuo oman lisänsä hakusanatutkimukseen havainnollistamalla suunnan, johon hakusanan trendi on suuntaamassa. Tekoälyn tuottamien hakusanojen osalta tarvitaan kuitenkin lisätutkimusta esimerkiksi hakusanamainonnan avulla, koska niiden suorituskyvystä ei ole vielä olemassa luotettavaa dataa.

Sovelluksen kehitystyön tavoitteet toteutuivat opinnäytetyön osalta. Tavoitteena oli rakentaa toimivat API-integraatiot sovelluksen ja palveluiden välille,

tallentaa haettu data tietokantaan ja mahdollistaa palveluiden käyttäminen yksinkertaisessa käyttöliittymässä. Opinnäytetyön aikana saatiin toteutettua toimiva prototyyppi, jonka jatkokehitys jatkuu opinnäytetyön jälkeen. Tämän takia sovelluksesta ei tehty build-versiota vielä opinnäytetyön aikana. Todennäköinen jatkokehityksen kohde on sovelluksen moduulien yhteistoiminta eli yhden hakukentän kautta haetaan tiedot kaikista sovellukseen integroiduista palveluista. Lisäksi käyttöliittymän toimintojen lisääminen, kaaviovaihtoehtojen lisääminen ja valmiiden hakusanamainoskampanjoiden lähetys Google Ads -palveluun ovat seuraavia sovelluksen kehitysaskeleita tulevaisuudessa.

Mahdollista jatkotutkimusta aiheesta voisi tehdä tutkimalla, kuinka tekoälyavusteinen SEO-työkalu voi vaikuttaa sisältömarkkinoinnin tehokkuuteen. Tutkimuksessa voitaisiin vertailla tuotetun sisällön hakukonenäkyvyyden kasvua tekoälyavusteisesti toteutettuna ja manuaalisesti toteutettuna. Lisätutkimuksen tarvetta olisi myös muissa hakukoneoptimoinnin vaiheissa, kuten tekoälyn roolista sisältöstrategiassa, sisällön arvioinnissa ja hakuintentioiden tunnistamisessa.

LÄHTEET

Beeceptor s.a. Authorization HTTP Header. WWW-dokumentti. Saatavissa: https://beeceptor.com/docs/concepts/authorization-header/ [viitattu 31.3.2025].

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. PDF-dokumentti. Saatavissa: https://aclanthology.org/N19-1423.pdf [viitattu 30.3.2025].

Fano Oy. 2023. SEO-optimoinnin menestystarina 1. Osa: Hakukoneoptimointi ja avainsanatutkimus. Blogi. Päivitetty 10.2.2023. Saatavissa: https://fano.fi/hakukoneoptimoinnin-menestystarina-1-osa-hakukoneoptimointi-ja-avainsanatutkimus/ [viitattu 29.3.2025].

Fielding, R. 2000. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine. Information and Computer Science. Väitöskirja. PDF-dokumentti. Saatavissa: https://ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf [viitattu 24.3.2025].

Gitonga, G. 2021. Understanding Rest Architecture. LinkedIn. WWW-dokumentti. Päivitetty 6.6.2021. Saatavissa: https://www.linkedin.com/pulse/understanding-rest-architecture-gabriel-gitonga/ [viitattu 24.3.2025].

Haltu Oy. 2024. Mitä on koneoppiminen? Malleja, hyötyjä ja haasteita. Blogi. Päivitetty 29.1.2024. Saatavissa: https://www.haltu.fi/blogi/koneoppiminen [viitattu 30.3.2025].

Helpotkotisivut.fi MS Oy. 2018. Ota Keyword Planner avuksi sisällön suunnitteluun. Blogi. Päivitetty 8.7.2022. Saatavissa: https://www.helpotkotisivut.fi/blogi/keyword-planner/ [viitattu 31.3.2025].

Ilomäki, O. 2023. Hakukoneoptimointi osana verkkomyynnin strategiaa. Blogi. Päivitetty 1.2.2023. Saatavissa: https://www.mkollektiivi.fi/julkaisut/hakukoneoptimointi-osana-verkkomyynnin-strategiaa [viitattu 24.3.2025].

Kallio, S. 2023. Avainsanatutkimus opas – 4 vaihetta parhaisiin avainsanoihin. Blogi. Päivitetty 22.11.2023. Saatavissa: https://santerikallio.com/avainsanatutkimus/ [viitattu 29.3.2025].

Karnik, N. 2018. Introduction to Mongoose for MongoDB. WWW-dokumentti. Päivitetty 11.2.2018. Saatavissa: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/ [viitattu 29.3.2025].

MongoDB Inc s.a. What Is MongoDB? WWW-dokumentti. Saatavissa: https://www.mongodb.com/company/what-is-mongodb [viitattu 29.3.2025].

Numminen, L. 2023. Mitä on luonnollisten kielten käsittely, eli NLP? Blogi. Päivitetty 17.10.2023. Saatavissa: https://www.finnishup.com/mita-on-luonnollisten-kielten-3.2025.202].5].

Rodriguez, A. 2015. Introduction to RESTful Web services. IBM Corp. WWW-dokumentti. Päivitetty 8.2.2015. Saatavissa: https://developer.ibm.com/articles/ws-restful/ [viitattu 24.3.2025].

SmartBear Software Inc. s.a. Understanding REST Headers and Parameters. WWW-dokumentti. Saatavissa: https://www.soapui.org/learn/api/understanding-rest-headers-and-parameters/ [viitattu 28.3.2025].

Thevapalan, A. 2023. A Beginner's Guide to The OpenAI API: Hands-On Tutorial and Best Practices. WWW-dokumentti. Päivitetty 18.10.2023. Saatavissa: https://www.datacamp.com/tutorial/guide-to-openai-api-on-tutorial-best-practices [viitattu 31.3.2025].

Tyyskä, I-M. 2023. Avainsanatutkimus – Näin teet sen paremmin kuin kilpailijasi. Blogi. Päivitetty 29.11.2023. Saatavissa: https://www.raikasdigital.fi/avainsanatutkimus/ [viitattu 29.3.2025].

Webbee Oy. 2022. Hakukoneoptimointi lyhyesti. Blogi. Päivitetty 25.1.2022. Saatavissa: https://webbee.fi/hakukoneoptimointi-lyhyesti/ [viitattu 29.3.2025].