LAPIN AMK
Lapland University of Applied Sciences

WEB-ALUSTA PIENYRITYKSEN TOIMINTOJEN YLLÄPITOON JA SEURANTAAN

Tirroniemi Eelis

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2024

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Eelis Tirroniemi	**Vuosi**	2024
Ohjaaja	Aku Kesti		
Toimeksiantaja	Psykoutsi		
Työn nimi	Web-alusta pienyrityksen toimintojen ylläpitoon ja seurantaan		
Sivumäärä	35		

Tässä opinnäytetyössä perehdytään web-sovelluksen suunnitteluun ja kehittämiseen yhteistyössä hyvinvointialalla toimivan yrityksen kanssa. Sovelluksen toiminta keskittyy yrityksen toimintojen ja tapahtumien kirjaamiseen sekä tietojen käsittelyyn. Työn tuloksena syntyy web-selaimella toimiva hallintatyökalu hyvinvointipalveluita tarjoavalle pienyritykselle. Sovellus toteutettiin JavaScript-ohjelmointikieleen perustuvaa Next.js:ää ja MongoDB Atlaksen NoSQL-tietokantajärjestelmää käyttäen.

Työssä käydään läpi toteutuksessa käytettyjä ohjelmia, sovelluksen suunnitteluprosessia yrityksen kanssa sekä edellä mainittujen teknologioiden ja palveluiden käyttöä ja käyttöönottoa teknisessä osuudessa. Työn eri vaiheet käydään pääpiirteittäin läpi käyttäen apuna esimerkkejä sovelluksessa toteutuvasta koodista ja käyttöliittymässä näkyvistä näkymistä. Lisäksi perehdytään keskeisiin asioihin sovelluskehityksen näkökulmasta.

Opinnäytetyössä yhdistetään web-sovelluksen tekninen toteutus sekä yrityksen kanssa yhdessä käyty suunnitteluprosessi. Lopputuloksena on tarkoitus toteuttaa suunnitelma osuudessa kerätty visio sovelluksen toiminnallisuuksista ja ulkonäöllisistä ominaisuuksista.

Työ toteutettiin toiminnallisena opinnäytetyönä projektiluontoisesti. Työssä isoa roolia näytteli jo aiemmin hankittu osaaminen ja tietotaito erilaisten web-sovellusten kehityksestä. Työssä etsitty ja käytetty tieto perustuu pitkälti teknologioiden ja menetelmien Internetsivuilta löytyneisiin dokumentaatioihin. Opinnäytetyön tuloksena syntyi sovellus, joka vastasi toimeksiantajan odotuksia. Työn aikana selvisi, että näillä menetelmillä on hyvä luoda selkeitä useilla laitteilla toimivia web-sovelluksia, jotka ovat myös helposti päivitettävissä/muokattavissa mikäli yrityksen toiminta muuttuessaan tai kasvaessaan sitä vaatii.

Avainsanat ohjelmistokehitys, ohjelmointi, tietokannat, web-ohjelmointi

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Study Programme in Information
and Communication Technology
Bachelor of Engineering

Author	Eelis Tirroniemi	**Year**	2024
Supervisor(s)	Aku Kesti		
Commissioned by	Psykoutsi		
Title	A Web Platform for Maintaining and Monitoring Small Business Operations		
Number of pages	35		

The aim of this thesis study was to design and develop a web application in collaboration with a well-being sector company.

The study combined the technical implementation of the web application with the collaborative design process with the company. The end goal was to succeed in creating an application that matches the plan and insights gathered in the design part of the study. The study was conducted as a functional thesis project. Previously acquired skills and knowledge in the development of various web applications played a significant role in the project. Information sought and used in the work was largely based on documentation found on the Internet regarding technologies and methods. The application was implemented using JavaScript-based Next.js and MongoDB Atlas NoSQL database system.

As a result, the thesis provides an application that meets the expectations of the client. It became apparent during the study that these methods are effective in creating clear, multi-device web applications that can easily be updated or modified if the company's operations change or grow. The application's functionality is centered around recording the company's operations and events, as well as processing data. The thesis covers the programs and tools used in the implementation, the application design process with the company, and the usage and implementation of the mentioned technologies and services in the technical section. The various stages of the study are outlined, utilizing examples of code from the application. Key aspects of application development are explored.

Keywords: databases, programming, software development, web-programming

SISÄLLYS

KÄYTETYT LYHENTEET JA TERMIT

API	Application Programming Interface, ohjelmointirajapinta
CSS	Cascading Style Sheets, verkkosivuille kehitetty tyylisivu dokumentti tiedosto.
IDE	Integrated Development Environment, Integroitu kehitysympäristö.
Kehys	Framework, joukko rakenteita, funktioita ja toimintoja ohjelman kehittämiseen.
npm	Node Package Manager, avoimen lähdekoodin paketinhallintajärjestelmä.
URI	Uniform Resource Identifier, uniikki merkkijono kertomaa tiedon sijainti erityisesti web kehityksessä.

1 JOHDANTO

Toimivan yrityksen yksi keskeisistä tekijöistä on helppo ja tehokas asiakastietojen, palvelu-/tuotetietojen sekä näitä yhdistämällä saatavien yhteenveto ja raporttitietojen, kuten ajanvaraustietojen ylläpito ja hallinnointi. Digitaalisten teknologioiden kehittyessä ja yritysten siirtyessä yhä enemmän sähköiseen toimintaympäristöön tarve tehokkaalle ja joustavalle yrityksen tapahtumien hallinnalle on korostunut entisestään myös aloilla, joilla ei aiemmin ollut tarvetta tietoteknisille ratkaisuille. Yrityksen toimintojen hallintaan liittyvät sovellukset tarjoavat yrityksille mahdollisuuden kerätä, tallentaa ja hallinnoida haluamiansa tapahtumien tietoja keskitetysti ja automatisoidusti, parantaen siten liiketoiminnan ajankäytön tehokkuutta.

Tämän opinnäytetyön tarkoituksena on kehittää Psykoutsi-nimisen hyvinvointipalveluita tarjoavan yrityksen henkilökunnan käyttöön tapahtumien hallintaan tarkoitettu helppokäyttöinen sovellus. Opinnäytetyön tuloksena saatavan sovelluksen avulla yritys voi tallentaa ja päivittää asiakas-, tuote- sekä palvelutietoja, seurata ajanvarauksia, analysoida sekä kerätä tallennetuista tiedoista haluamiaan kokonaisuuksia esimerkiksi laskutietojen tekemistä varten.

Työssä keskitytään kahteen pääosaan, jotka ovat suunnittelu ja tekninen toteutus. Suunnitteluosuudessa käydään läpi sovelluskehityksen suunnittelua asiakkaan kanssa ja sitä, miten hyvinvointialalla työskentelevän asiakkaan toiveet saadaan mahdollisimman hyvin huomioitua sovelluksen toteutuksessa. Teknisen toteutuksen osuudessa keskitytään pääasiassa ohjelmointipuolen menetelmiin ja käytäntöihin, erityisesti siihen, miten tässä projektissa käytetyt teknologiat ja palvelualustat on otettu käyttöön. Tässä osiossa perehdytään Next.js-sovellusten perusajatukseen ja kerrotaan, miten tätä ajatusta on sovellettu projektin tuloksena syntyvän sovelluksen toteutuksessa. Osiossa käytetään runsaasti esimerkkejä ja kuvia käyttöliittymän näkymistä ja sen taustalla tapahtuvasta koodista.

Opinnäytetyön tutkimusmenetelmänä käytetään käytännön kehitysprojektia, jota toteutetaan sekä Scrum- että Lean-menetelmiä hyödyntäen. Projektissa tuotetulle sovelluksen asetetaan pääpiirteet ja raamit alussa saadun idean perusteella, ja niitä lähdetään toteuttamaan järjestelmällisesti osio kerrallaan. Projektin aikana

ollaan kuitenkin koko ajan tiiviisti vuorovaikutuksessa kohdeyrityksen kanssa, tarkoituksena pyrkiä tarjoamaan projektin toteutukselle joustavuutta pääpiirteiden ulkopuolella. Projektin etenemisen aikana haluttiin mahdollistaa toimeksiantajalla heräävien ideoiden muutosten toteuttaminen mahdollisimman joustavasti. Projektin aikana arvioidaan sovelluksen toiminnallisuutta ja käytettävyyttä. Tuloksena odotetaan käytännönläheistä selkeäkäyttöistä sovellusta, joka vastaa kohdeyrityksen tarpeita ja toiveita sekä toiminnallisesti että visuaalisesti. Sovellukselta odotetaan toimeksiantajan yritystoiminnassa toimintojen suunnittelun helpottamista ja selkeyttämistä sekä sitä kautta ajankäytön tehostamista.

2 KÄYTETYT TYÖKALUT JA MENETELMÄT

2.1 Yleisesti web-kehityksestä

Ohjelmistokehittämisen, web-kehittämisen tai ohjelmoinnin parissa ylipäätään, on vaikea sanoa tismalleen mitkä työkalut ovat parhaita tai ”oikeita” mihinkin tarkoitukseen. On erilaisia teknologioita, joilla kehitetään ja luodaan nettisivuja, ja joita yhdistämällä saadaan lopullisia pitkälle kehittyneitä ja monimutkaisia sovelluksia. Jotta sovellus saadaan toimimaan haluttujen käytäntöjen ja menetelmien mukaan, voi sen toiminta koostua suuresta joukosta erilaisia teknologioita. (Mozilla Corporation 2023.)

Web kehityksen teknologioiden käyttöön ja käyttöönottoon luodaan paljon standardeja ja käytäntöjä. Näistä dokumenteista selviää, miten teknologiat saadaan käyttöön omassa projektissa, mutta ne eivät ole kovin hyödyllisiä opettamaan miten niitä sovelluksessa tulisi käyttää omaksi eduksi. Se miten käyttöönotetun teknologian soveltaminen tapahtuu oman sovelluksen tarpeisiin ja mitä teknologioita yhdistellään ja integroidaan keskenään saavuttamaan halutun kaltainen lopputulos, on pitkälti kiinni sovelluksen kehittäjän mieltymyksistä. (Mozilla Corporation 2023.)

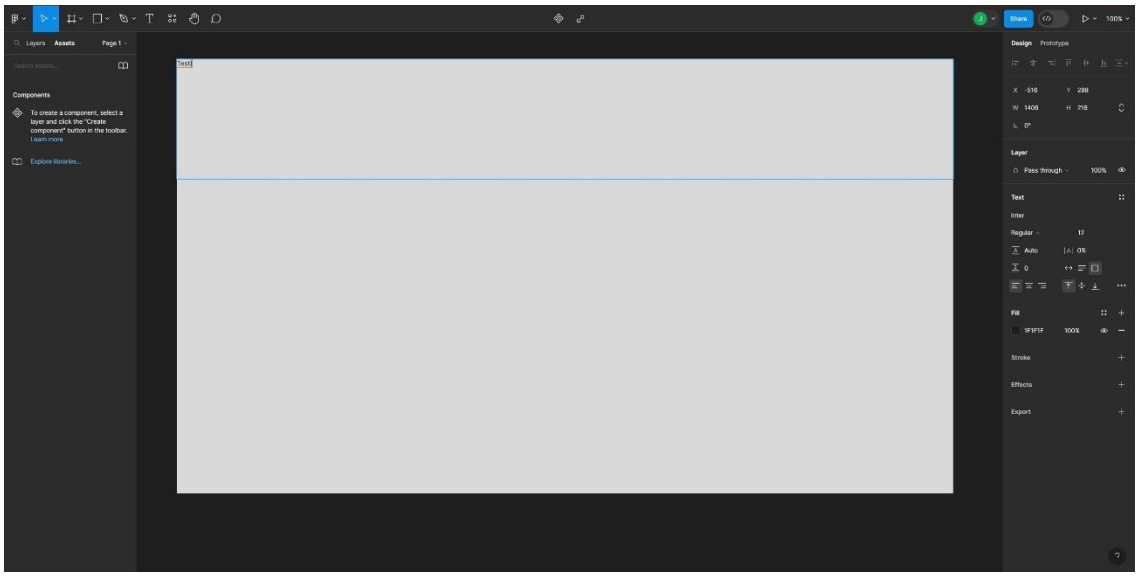
2.2 Visuaalisen suunnittelun työkalut

Tässä työssä valmistuvan sovelluksen on tarkoitus toimia yrityksen henkilökunnalle selkeänä sovelluksena yrityksen toimien ja tapahtumien arkistointiin ja suunnitteluun. Tästä syystä päätin sovelluksen visuaalisessa toteutuksessa keskittyä näyttävien kuvien, animaatioiden ja siirtymien sijaan selkeisiin ja helposti hahmotettaviin näkymiin ja toimintoihin sekä yhtenäisiin värikokonaisuuksiin. Tässä luvussa on tarkoitus esitellä visuaalisen suunnitellun apuna käytetyt työkalut. Enemmän visuaalisen suunnitellun toteutuksesta ja siihen vaikuttavista tekijöistä kerrotaan luvussa 3.

Figma on Internetissä käytettävä käyttöliittymien suunnitteluun tarkoitettu sovellus. Se mahdollistaa ryhmissä työskennellessä web-sivujen/sovellusten hahmot-

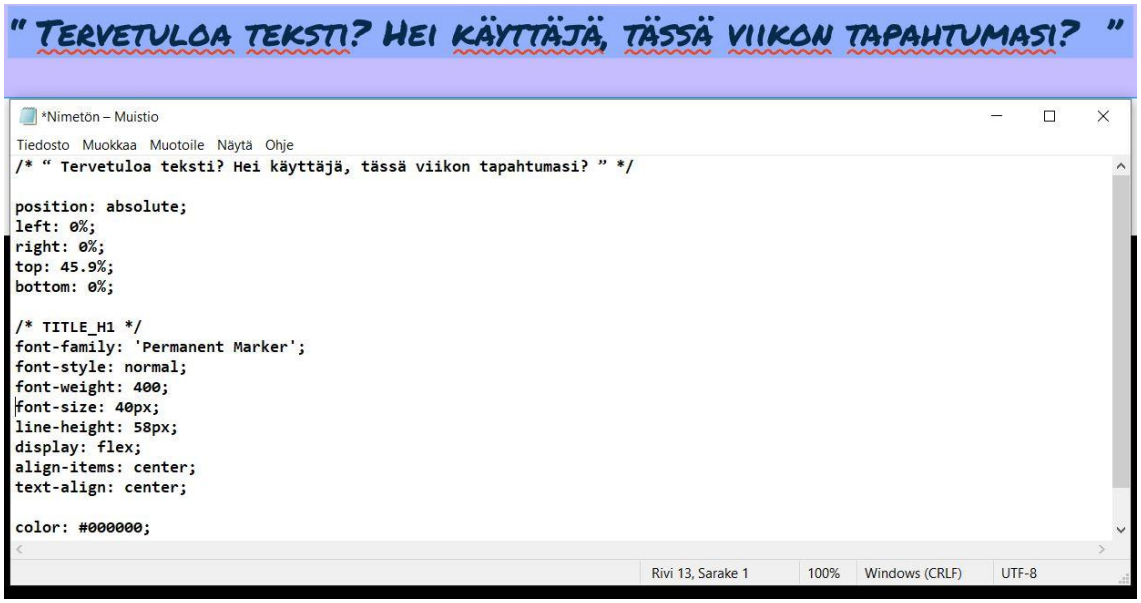
telun ja suunnittelun useamman henkilön kesken yhtäaikaisesti. Figman kaltaisissa sovelluksissa etuna sovelluskehitystä ajatellen on se, että tehdyistä sommitelmista saadaan suoraan esille ohjelmoijalle erittäin hyödyllisiä eri elementtien visuaalisuuteen vaikuttavista arvoista, kuten fontti/fonttikoko, riviväli, ikkunoiden leveys/korkeus, värikoodit, ja niin edelleen.

Kuviosta 1 nähdään Figma-sovelluksen editointinäkymä. Keskellä on sommitelmalle varattu tyhjä taulu. Vasemmalla palkissa sijaitsee lisätyt sivut ja niiden sisällä olevat elementit, kuten tekstit ja eri osioita vastaamaan tarkoitetut tilat. Oikealla palkissa on elementtien muokattavia arvoja, kuten fonttiasetukset, elementin leveys / korkeus ja niin edelleen.



Kuvio 1. Figma-sovelluksen perusnäkymä

Kuviossa 2 nähdään sovelluksella tehty otsikkoelementti ja siitä saadut arvot CSS-tiedoston muodossa. Niitä voidaan helposti käyttää käyttöliittymän ohjelmoinnin tukena saamaan sommitelman kanssa identtisen näköinen lopputulos.



Kuvio 2. Figmassa tehty otsikkoelementti ja sen arvot CSS-tiedoston muodossa

GIMP on avoimen lähdekoodin ilmainen kuvankäsittelyohjelma, joka on verrattavissa esimerkiksi monelle tuttuun Adobe PhotoShop-ohjelmaan (GIMP 2023). Joissain tapauksissa kuvien muokkaaminen ennen web-sivulle asettelua on tehokkaampi tapa saada halutun kaltainen lopputulos. Yritysten logot ja kuvat voivat vaatia joskus pientä kuvankäsittelyä, kuten taustan muuttamista läpinäkyväksi, jotta ne sopivat web-sivuille paremmin. Tässä projektissa kuvankäsittelyn käyttö oli vähäistä, mutta sitä tarvittiin muun muassa yrityksen logojen muokkaamiseen.

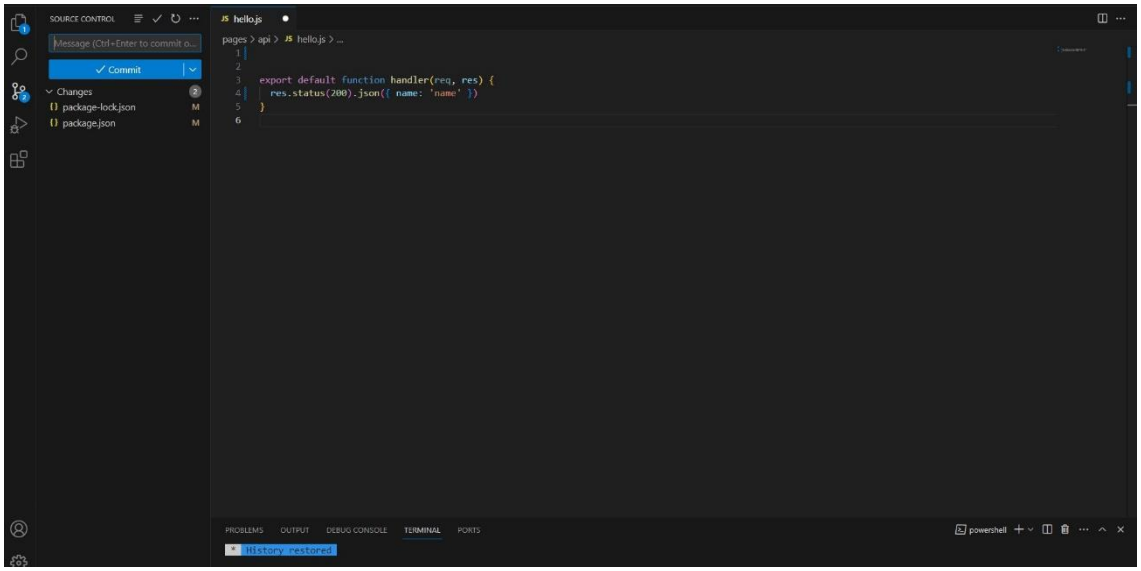
2.3 Teknisen toteutuksen työkalut

Haluan tässä luvussa tuoda esille työkaluja, joita käytin työn teknisessä toteutuksessa. Vaikka kaikki mainitut työkalut eivät näyttele kovin näkyvää roolia, ovat ne silti erittäin oleellisia luotettavan ja tehokkaan ohjelmoinnin toteutuksessa. Ohjelmointikielen ja erilaisten viitekehysten osalta valintojen tekeminen projektia varten voi olla hankalaa, varsinkin jos tarkoitukseen on useita varta vasten tehtyjä hyviä vaihtoehtoja. Projektissa tuotetun sovelluksen lopullinen käyttötarkoitus pitäisi olla määräävin kriteeri näitä valintoja tehdessä. Eri ohjelmointikielet sisältävät eri ominaisuuksia ja vahvuuksia ja ovat siten paremmin soveltuvia eri käyttötarkoituksiin. Web-kehityksessä yleisimmin käytetyt kielet ovat JavaScript ja Python (Chauhan 2023,1).

2.3.1 Kehitysympäristö

IDE eli integroitu kehitysympäristö on ohjelmistokehittäjille tarkoitettu työkalu, joka tarjoaa joukon ominaisuuksia ja työkaluja helpottamaan ohjelmistokehitystä. Kehittäjä voi suorittaa monia eri tehtäviä (esim. tekstin muokkaus, ohjelman rakennus ja kielen kääntö työkalut, ”debuggaus”, jne.) yhden sovelluksen sisällä sen sijaan, että käyttäisi useita erillisiä työkaluja. Yleensä IDE sisältää myös ominaisuuksia, jotka tekevät ohjelmoinnista tehokkaampaa ja selkeämpää, kuten syntaksien korostaminen ja ennakoiva koodin syöttö. Tarkoituksena on tehdä ohjelmistokehitys tehokkaammaksi ja mukavammaksi tarjoamalla yhtenäinen ja integroitu työympäristö. (Gillis 2018, 1.)

Tätä projektia suorittaessa käytin Visual Studio Codea, joka on Microsoftin kehittämä ja ylläpitämä avoimen lähdekoodin IDE. Se on yksi suosituimmista editoreista kehittäjien käytössä ja tarjoaa monipuolisen tuen eri kielille ja kirjastoille sekä sisäänrakennetut Git-komennot, joilla työn tallentaminen ja järjestelmällinen version ylläpito versionhallintatyökalun välillä tapahtuu helposti. Kuviossa 3 näkymä Visual Studio Code kehitysympäristöstä.



Kuvio 3. Näkymä Visual Studio Code

2.3.2 Ohjelmointikielet ja kehykset

React on JavaScript-ohjelmointikieltä käyttävä kirjasto, joka on suunniteltu käyttöliittymien rakentamiseen erityisesti yksisivuisiin sovelluksiin ja dynaamisiin käyttöliittymiin. Reactin avulla pystyy helposti luomaan yksittäisiä rakennuselementtejä, joita kutsutaan komponenteiksi ja näitä yhdistelemällä saadaan kokonaisia sivunäkymiä ja sovelluksia. (Bondar 2023, 1.) Aikaisempaan kokemukseen perustuen koen, että React-tyyppisten sovellusten käyttö web-kehityksessä on hyvin selkeää ja joustavaa siinä mielessä, että ohjelman toiminta on jaettu erillisiin komponentteihin. Tarpeen vaatiessa on sovelluksen yksittäisten osioiden ja toimintojen muokkaaminen helppoa.

Tässä työssä käytin React-pohjaisten sovellusten kehitykseen tarkoitettua Next.js-kehystä. Next.js tarjoaa valmiiksi konfiguroidun rakenteen ja joukon ominaisuuksia helpottamaan React-sovellusten kehitystä. Next.js tukee Client-Side Renderingin lisäksi Server-Side Renderingiä mikä voi monessa tilanteessa parantaa sivun/sovelluksen suorituskykyä sekä hakukoneoptimointia. (Niżyński 2023,1.) Perimmäinen syy miksi päädyin näihin valintoihin sovelluksen teossa oli, se että JavaScriptillä ja React-kirjastoilla tekeminen oli minulle entuudestaan tuttua. Next.js-kehystä käyttäen sain React-sovellukseeni tässä projektissa erittäin hyödylliseksi tulleita ominaisuuksia, kuten API-reitit, joista enemmän työn teknisessä osuudessa.

2.4 Versionhallinta

Sovelluskehityksessä tiedostojen ja koodin määrä kasvaa yleensä varsin suureksi ja niissä tapahtuvien muutoksien seuranta sekä hallinta voi muodostua ongelmaksi ilman erillistä työkalua. Versionhallinta on osa modernia ohjelmistokehitystä, ja sillä tarkoitetaan prosesseja ja työkaluja, joiden avulla seurataan ja hallitaan ohjelmakoodin muutoksia ajan kuluessa. Kehittäjät voivat esimerkiksi luoda uusia haaroja (*branches*) työstääkseen ominaisuuksia tai korjatakseen virheitä ilman, että ne vaikuttavat suoraan pääkehityshaaraan. Tämä helpottaa kokeiluja ja antaa mahdollisuuden testata muutoksia ennen niiden yhdistämistä päähaa-

raan. Lisäksi versioiden hallinta mahdollistaa helpon siirtymisen vanhoihin versioihin, jos ilmenee ongelmia uusimman version kanssa. (Harvie 2018,1.) Työssäni hyödynsin GitHub-palvelua, joka tarjoaa tallennustilan Git-versionhallintajärjestelmälle sekä graafisen käyttöliittymän Git-toiminnoille. Tämä palvelu mahdollistaa projektien hallinnan verkossa. Käyttämäni IDE oli myös integroitavissa GitHub-palveluun, minkä ansiosta pystyin suorittamaan Git-toimintoja suoraan kehitysympäristöstä käsin.

2.5 Tietokannat

Pilvimuotoisen tietokannan valinta tähän projektiin oli itsestäänselvyys, erillistä laitteistoa tai palvelimen ylläpitoa ei haluttu omasta toimesta järjestää. Tässä projektissa on kyse yhden henkilön yrityksestä, joten voi olettaa, että tietokannan koko pysyy hyvinkin maltillisena. Tietokannan varmuuskopioinnit ja toiminnan päivitykset tässä tekee automaattisesti MongoDB, jolloin kehittäjänä ei tarvitse huolehtia itse tietokannan toimintojen ylläpidosta. Halusin tietokannan olevan helposti muokattava ja varastoivan strukturoimatonta dataa, jotta tiedon analysointi ja muokkaaminen voitaisiin suorittaa mahdollisimman joustavasti työssä kehitettävää sovellusta käyttäen. Näiden tietojen perusteella valinnaksi koitui pilvitietokantapalvelu MongoDB Atlas, joka tarjoaa pilvessä toimivan tietokannan palveluna käyttäen dokumenttimuotoista NoSQL-tietokantaa.

NoSQL-tietokannat ovat relaatiotietokannoista poiketen ei-taulukkomuotoisia ja tallentavat dataa eri tavalla. NoSQL-tietokantoja on useita tyyppejä perustuen niiden datamalliin. Tähän voidaan ottaa esimerkkinä tässä sovelluksessa käytetyn palvelun MongoDB:n käyttämä dokumenttimuotoinen tietokanta, joka tallentaa tiedot JSON/BSON-dokumentteina. NoSQL-tietokanta tarjoaa joustavuutta datan siirtoon, ja ne skaalautuvat helposti suurilla datamäärillä ja suurella käyttäjäkuormituksella. (Schaefer 2023.) Kuviosta 4 nähdään esimerkki MongoDB palveluun tallentuneesta datasta.

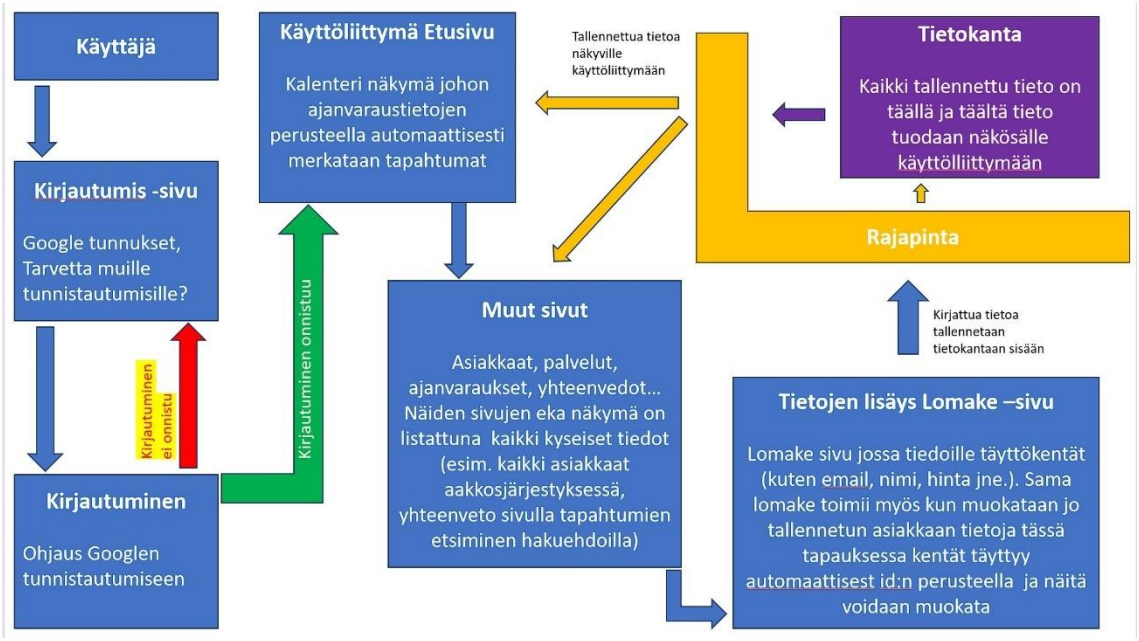
Kuvio 4. Esimerkki sovelluksella tallennetusta varaus-nimisestä kohteesta Mongo DB Atlas-palvelussa.

3 SUUNNITTELU

3.1 Yksinkertaisen sommitelman luonti

Saatuani yritykseltä idean sovelluksen toiminnasta ja vaatimuksista, halusin luoda sitä vastaavan kaavion sekä sommitelman käyttöliittymän ulkonäöstä, jotta voisin helpottaa sovelluksen toiminnan demonstroimista ensimmäisessä kehitys-palaverissa toimeksiantajan kanssa. Tässä vaiheessa ei ollut vielä tarkoitus luoda täydellistä kuvaa valmiista sovelluksesta, vaan enemmänkin luoda yhteisen hahmottelun avuksi työkaluja, joiden pohjalta voidaan toimeksiantajan kanssa lähteä vapaasti jatkokehittämään ideoita sivuston toimintaa ja ulkonäköä koskien.

Kuviossa 5 nähdään tekemäni kaavio sovelluksen toiminnasta saamani idean perusteella. Koska toimeksiantajalla ei ollut kokemusta sovellusten teknisestä toteutuksesta, se millä ratkaisuilla kaavion mukainen alusta toteutettaisiin, oli pitkälti minun vastuullani. Tekniikoista ja työkaluista, joilla päädyin kokonaisuuden tekemään kerrottiinkin jo luvussa 2 ja teknisestä toteutuksesta enemmän työn teknisessä osuudessa (luku 4).



Kuvio 5. Alussa kehitetty kaavio sovelluksen toiminnasta.

Käyttöliittymän visuaalisen sommitelman luomisessa avuksi otin Figma-nimisen web-työkalun. Figman avulla voi helposti luoda sommitelman web-sivuista PowerPoint-tyylisesti. Pystyin hahmottelemaan asiakkaalle alustavia sivurakenteita, tyylikokonaisuuksia sekä demonstroimaan erilaisten sivujen prototyyppien avulla sivujen mahdollista toimintaa käytännössä, ennen varsinaista ohjelmointityön aloittamista. Näin pystyin ottamaan ohjelmoinnista kokemattoman asiakkaan mukaan suunnitteluprosessiin tehden asiakkaan toiveiden esille tuomisen helpoksi. Alla olevasta kuvioista 6 nähdään ensimmäiset sommitelmani sovelluksen sivuista.

Kuvio 6. Ensimmäiset sommitelmat sovelluksen sivuista

3.2 Yhteinen ideointi yrityksen kanssa

Ennen sommitelmani esittelyä yritykselle ja keskittymistä sen enempää sovelluksen visuaaliseen puoleen halusin tehdä mahdollisimman selväksi sovelluksen toiminnan kysymällä kysymyksiä, kuten

- mitä etusivulla halutaan näyttää
- mitä tietoja sovelluksella käsitellään
- miten kirjattuja tietoja halutaan käyttää / hyödyntää (Forbes Technology Council 2018,1).

Näillä kysymyksillä ja kuvion 5 kaaviota käyttäen saatiin hyvin hahmoteltua sivuston rakenne, montako sivua sovellus vaatii ja mitä kullakin sivulla käsitellään. Toimeksiantajalla oli selkeä kuva mielessään sovelluksen toiminnasta, joka tässä vaiheessa oli tarkoitus saada mahdollisimman tarkasti esille silmin nähtävään tuotokseen.

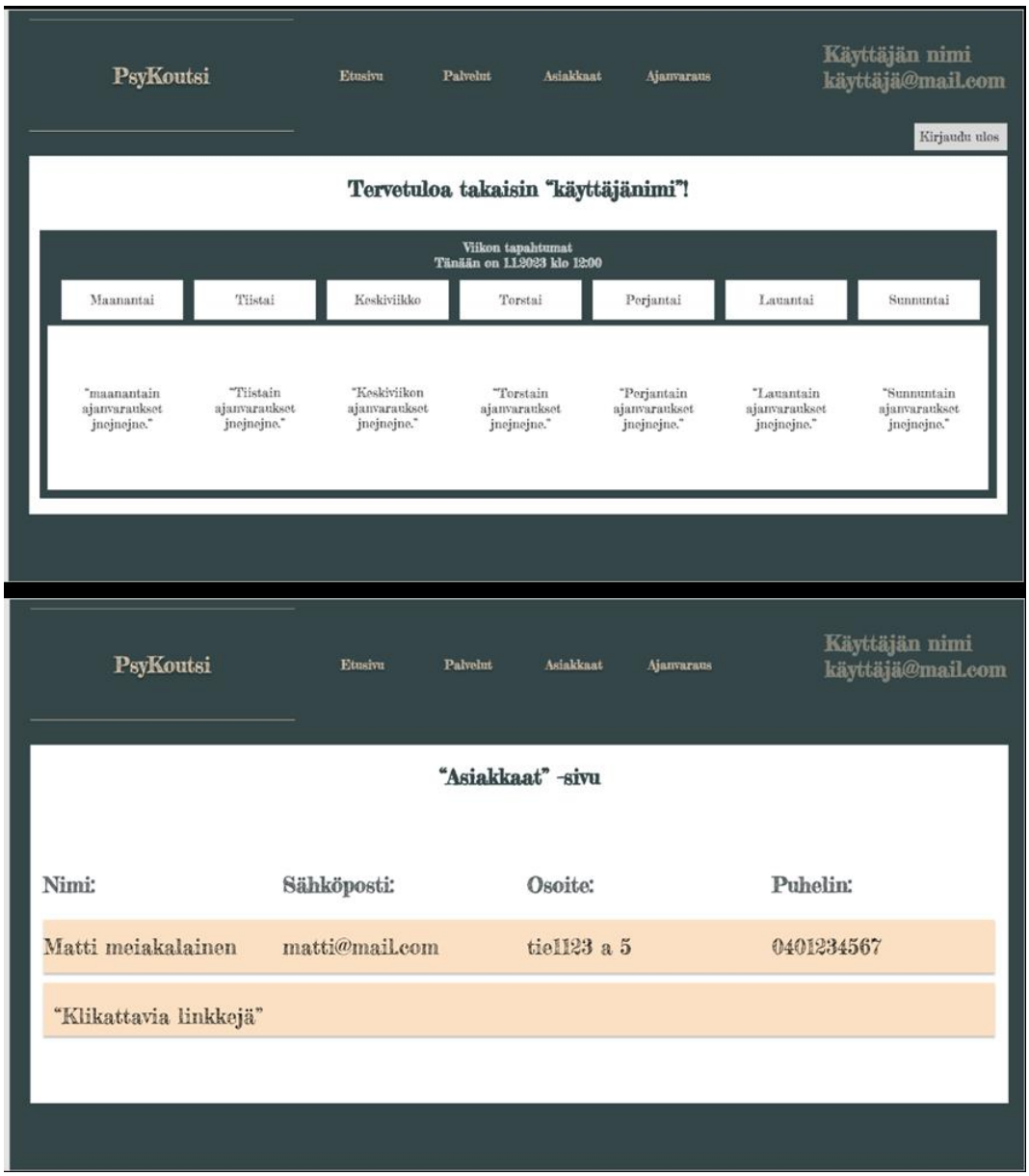
Sovelluksessa pitäisi olla omat sivut asiakkaiden, palveluiden ja ajanvarausten tiedoille. Uusia asiakas- ja palvelutietoja pitäisi pystyä lisäämään, muokkaamaan ja poistamaan aina halutessaan. Uusia asiakas- ja palvelutietoja lisätessä pitäisi olla tietyt vakiotietokentät, joita ei tarvitse kirjoittaa arvoa lukuun ottamatta, kuten hinta, nimi ja sähköposti. Näiden lisäksi pitäisi olla myös vapaa kirjoituskenttä, johon voisi luoda vapaamuotoisen kirjoitelman/huomautuksen tarvittaessa. Ajanvaraussivulla pitäisi pystyä valitsemaan valmiiksi tallennettuja asiakas- ja palvelutietoja ja pysytä yhdistämään näitä päivämäärään ja kellonaikaan. Ajanvaraukset olisi tarkoitus automaattisesti ilmestyä myös etusivun kalenterinäkymään.

Näiden lisäksi olisi tarvetta vielä erilliselle ”yhteenveto”-sivulle, jossa olisi mahdollisuus hakea kaikki tapahtuneet ajanvaraukset esimerkiksi aikavälillä, asiakkaan, palvelun nimen perusteella tai yhdistelemällä näitä hakukriteereitä. Sivun pitäisi luoda kriteereillä haetuista ja löydetyistä kohteista jokaisesta oma yhteenveto, jossa on asiakkaan tiedot, palvelun tiedot sekä tapahtuman päivämäärä ja kellonaika. Kirjautuminen sovellukseen pitäisi olla mahdollista ainoastaan yrityksen ainoan työntekijän Gmail-tunnuksilla. Lisätoiveena oli, että osia tietokantaan

tallennetuista tiedoista olisi mahdollisimman helppo saada näytettyä tulevaisuudessa myös mahdollisen kaikille avoimen sivun kanssa, jossa asiakkaat voisivat itse tutkia palveluita, luoda käyttäjätunnuksen ja sitä käyttäen varata vapaita aikoja.

3.3 Suunnittelun yhteenveto

Luvun 3.1 tietojen perusteella pystyttiin siirtymään suunnittelussa toimeksiantajayrityksen kanssa aiemmin kuviossa 6 nähtyjen sommitelmien tutkimiseen ja kehittämiseen. Toimeksiantajayrityksellä ei ollut vahvaa näkemystä siitä, miltä sovelluksen tulisi näyttää visuaalisten elementtien puolesta. Aiemmissa sommitelmissa (kuvio 6) halusin korostaa värieroilla eri osioiden tilanvientiä, kuten esimerkiksi navigaatiopalkki violetilla taustalla kalenteri näkymä mustalla taustalla ja nappi vihreällä taustalla tehden sivun hahmottamisen suunnittelutilanteessa helpoksi kaikille osapuolille Alla olevassa kuvioissa 7 nähdään lopputulos, joihin yhteisissä sommitelmissa päästiin



Kuvio 7. Sommitelmat sovelluksen sivuista toimeksiantajan kanssa

Kuvioita 6 ja 7 vertaamalla pystyy huomaamaan, että sivurakenne pysyy toimeksiantajan kanssa tehdyissä sommitelmissa hyvin pitkälti samana. Tämä onkin hyvin perinteinen rakenne mitä nähdään, kun seurataan mitä tahansa web-palvelua, jossa asioita kategorisoidaan osioita eri ryhmiin (nettikaupat, suoratoistopalvelut). Näin voidaan todeta, että ensimmäiset sommitelmat olivat hyvin onnistuneita rakenteen osalta, eikä toimeksiantajalla ollut suurempia muutos ideoita rakenteen selkeyttämisen osalta tässä vaiheessa.

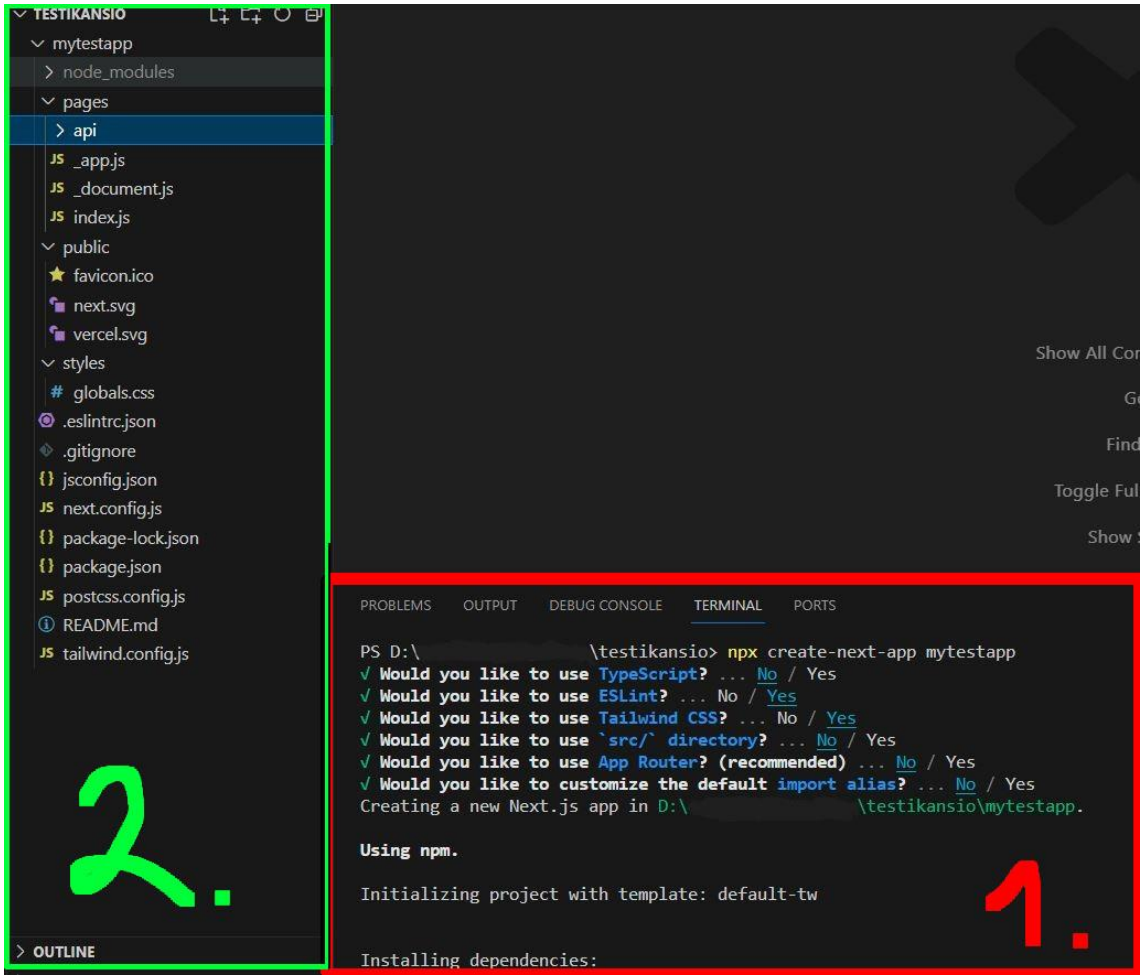
Värimaailmasta muokattiin yhtenäisempi toimeksiantajan väri toiveilla, näin sovelluksen ulkoasusta saatiin paljon valmiimman oloinen. Huomataan myös, että

vaikka kuviossa 7 nähdään kaksi eri sovelluksen sivua, vain valkoisen taustan sisällä oleva tieto muuttuu. Tämä oli molempien osapuolten mielestä sivuilla liikkumisen selkeyden kannalta oleellinen elementti. Tämä havainto helpottaa myös ohjelmointityötä, kun sivujen luomisessa voidaan uudelleen käyttää tiettyjä elementtejä. Tässä vaiheessa toimeksiantaja oli hyvin tyytyväinen kuvioissa 7 esiintyviin sommitelmiin. Nämä sommitelmat toimivat sovelluksen käyttöliittymän teknisen toteutuksen mallina ja runkona pitkälti koko projektin ajan.

4 TEKNINEN TOTEUTUS

4.1 Projektin alustus

Ennen kuin Next.js-projekti voitiin aloittaa, asennettiin Node.js. Node.js on avoimen lähdekoodin palvelinpuolen JavaScript-ympäristö, joka on suunniteltu suorittamaan JavaScript-koodia palvelimella. Node.js mahdollistaa myös npm:n käytön, joka on JavaScript-kirjastojen ja -työkalujen asentamista helpottava paketinhallintajärjestelmä ja jota tässäkin tapauksessa hyödynnettiin projektin alustamiseen. Tämän jälkeen siirryttiin kehitysympäristössä sovelluskansion terminaalinäkymään ja alustettiin uusi Next.js-projekti komennolla *npx create-next-app*. Tässä vaiheessa npm asensi kansioon next.js-sovelluksen sen sisältämine ominaisuuksineen ja, jonka tiedostorakenne on esitetty kuviossa 8.



Kuvio 8. Näkymä kehitysympäristöstä Visual Studio Code

Kuviossa 8 nähdään kehitysympäristössä Next.js-sovelluksen alustus. Kuviossa punaisella kehyksellä nähdään terminal-näkymä, jossa D-asemalla sijaitsevaan testikansioon luodaan *mytestapp*-niminen Next-sovellus käyttämällä komentoa *npx create-next-app mytestapp*. Tämän jälkeen asennusprosessi kysyi, halutaanko saman asennuksen yhteydessä asentaa muutamia Next-sovelluksen kehityksessä hyvin tyypillisiä lisäkirjastoja/moduuleja. Näiden valintojen perusteella npm suoritti asennusprosessin ja kuvion 8 tapauksessa testikansioon ilmestyi *mytestapp*-niminen sovellus, jonka tiedostorakenne näkyy kuviossa vihreällä kehyksellä. Seuraavissa kappaleissa käydään läpi tässä sovelluksessa toteutuneesta Next.js-alustuksen keskeisimmät kansiot ja tiedostot.

Pages-kansioon tallennetaan sovelluksen sivut, kuten etusivu, yhteystiedotsivu ja kaikki muut mitä sovellukseen kehityksen aikana tulee. Se sisältää asennuksen jälkeen index.js-sivun, joka vastaa juurisivua (www.yrityksenosoite.com, ”/”). Tämä on yleisesti se tiedosto, josta muokataan omaan tarkoitukseen sovelluksen ensimmäinen sivu, se joka sovelluksen avatessa ilmestyy eli etusivu. Jokainen tähän kansioon tallennettu *.js*-tiedosto (sivu) luo reitin sovellukselle, johon on helppo siirtyä Next.js-sovelluksen reititinominaisuutta käyttäen. Esimerkiksi ”palvelut.js” niminen tiedosto pages-kansiossa avautuu selaimessa *yrityksenosoite.com/palvelut* ja niin edelleen.

Public-kansio toimii staattisten tiedostojen säilytyspaikka. Tällaisia tiedostoja ovat esimerkiksi kuvat ja *faviconit*. Tämän kansion tiedostot ovat julkisesti saatavilla verkkosivun juurihakemistosta.

Styles-kansioon tallennetaan CSS-tiedostot, mikäli semmoisia käytetään. CSS-tiedostot ovat tekstitiedostoja, joissa määritellään verkkosivun ulkoasuun liittyviä arvoja. Ne sisältävät arvoja, jotka ohjaavat esimerkiksi tekstin fonttia ja värimaailmaa.

Node-modules-kansiossa säilytetään projektin riippuvuudet (*dependencie*s). Tällaisia ovat esimerkiksi kolmannen osapuolen kirjastot ja moduulit jotka ladataan npm-paketinhallintajärjestelmää käyttäen.

Tämän lisäksi sovelluksesta löytyy tiedostoja, kuten ”package.json” ja ”next.config.js”, jotka sisältävät riippuvuuksia (*dependencies*), skriptejä (esim. kehityspalvelimen käynnistäminen) ja muita sovelluskohtaisia mukautettuja asetuksia. Tässä vaiheessa alustus oli valmis ja voitiin aloittaa sivujen kehitys sekä avata paikallinen kehityspalvelin kehityksen tueksi. Kehityspalvelin mahdollistaa nopean palautteen kehittäjälle. Koodiin tehdyt muutokset ja niiden vaikutus sivuun nähdään välittömästi jokaisen kooditiedoston tallennuksen jälkeen.

4.2 React-sovellusten keskeisistä toiminnoista

Tässä työssä käytetty Next.js on JavaScript-ohjelmointikieleen perustuva React-pohjaisten sovellusten kehitykseen tarkoitettu kehys. React-sovellusten ydinajatus on jakaa käyttöliittymä osiin, joita kutsutaan komponenteiksi. Komponentit toimivat pieninä käyttöliittymäelementteinä, jotka voivat välittää tietoa näkymälle props-ominaisuutena, ja tämä tieto voi muuttua ajan myötä. Komponentit ovat uudelleenkäytettäviä, joten niitä voidaan käyttää sovelluksessa toistuvasti.

Komponentit helpottavat kehittäjiä rakentamaan käyttöliittymää jakamalla sen useisiin osiin, mikä tekee suunnittelusta ja rakentamisprosessista tehokasta ja hyvin rakenteellista. Yksinkertaisesti tarkasteltuna komponentit ovat kuin mikä tahansa JavaScript-funktio. Ne suorittavat niille määrätyn tietyn tehtävän, mutta eri ympäristössä ja lähestymistavalla. Niitä kutsuttaessa voidaan syöttää arvoja (joita kutsutaan props-ominaisuuksiksi) ja ne palauttavat ne React-elementteinä, jotka näkyvät käyttäjän näytöllä. (Lazuardy, Anggraini. 2022, 2).

4.3 Sovelluksen sivujen luonti ja rakenne

Kuten suunnitelmaosuudessa tehdyistä sommitelmista (kuvio 7) huomattiin, sovelluksen sivujen rakenne koostui vihreästä taustasta, ylhäällä sijaitsevasta navigointipalkista ja valkoisesta tilasta, johon päivitettiin kullakin sovelluksen sivulla ollessa sen sivun näytettävä sisältö. Tämän perusteella lähdin kehittämään ”tausta”-komponenttia nimeltä ”Layout”.

Kuviossa 9 näkyvä Layout-komponentti palauttaa HTML-elementin, jota voisi kuvailla ”säiliönä”. Se voi pitää sisällään useita eri elementtejä (kuvia <img>, tekstiä <h1>, <h2>,<p>, toisia säiliöitä , jne.). Tässä tapauksessa se sisältää tekemäni NavBar-komponentin ja toisen div-säiliön, joka sisältää {children} tagin. Kaarisulkujen sisällä ({children}) näytetään se tieto, mitä Layout-komponenttia kutsuttaessa asetetaan <Layout> ja </Layout>-tagien sisään ikään kuin Layout-tagin ”lapsi”-elementiksi.

```
return (
  <div className="bg-themeDark min-h-screen flex flex-col">
    <NavBar/>
    <div className="bg-white  rounded-md  mx-6 mb-6  p-6">
      {children}
    </div>
  </div>
)
}
```

Kuvio 9. Sovelluksen ”/components”-kansiossa sijaitseva ”Layout.js” tiedosto eli ”tausta”-komponentti

Kuviossa 10 nähdään sovelluksen etusivun koodi kokonaisuudessaan ja miten tehty Layout.js-tiedosto (”tausta”-komponentti) saadaan käyttöön sivujen kehityksessä. Alussa nähdään ”import”-lausekkeet, joilla komponentit tuodaan Index.js-tiedoston käytettäväksi käyttäen tiedostopolkua (Import Layout from ”@/components/Layout”). Etusivu on funktio nimeltä Home, jonka toiminta on palauttaa ”*return*”-komennon sulkujen sisältämä elementti, joka tässä tapauksessa on luomani Layout-komponentti. Layout-tagien (<Layout> </Layout>) sisälle on asetettu etusivulle haluttuja elementtejä, kuten otsikko sekä Viikkokalenteri (<h1>,) Viikkokalenteri on tekemäni komponentti, joka tuo kalenterinäkymän viikoksi eteenpäin nykyhetkestä ja asettaa tietokannasta saadut ajanvarustiedot ajan perusteella kalenterinäkymään. Nämä elementit, kuten kuviosta 12 selvisi, näytetään {children}-muuttujan paikalla.

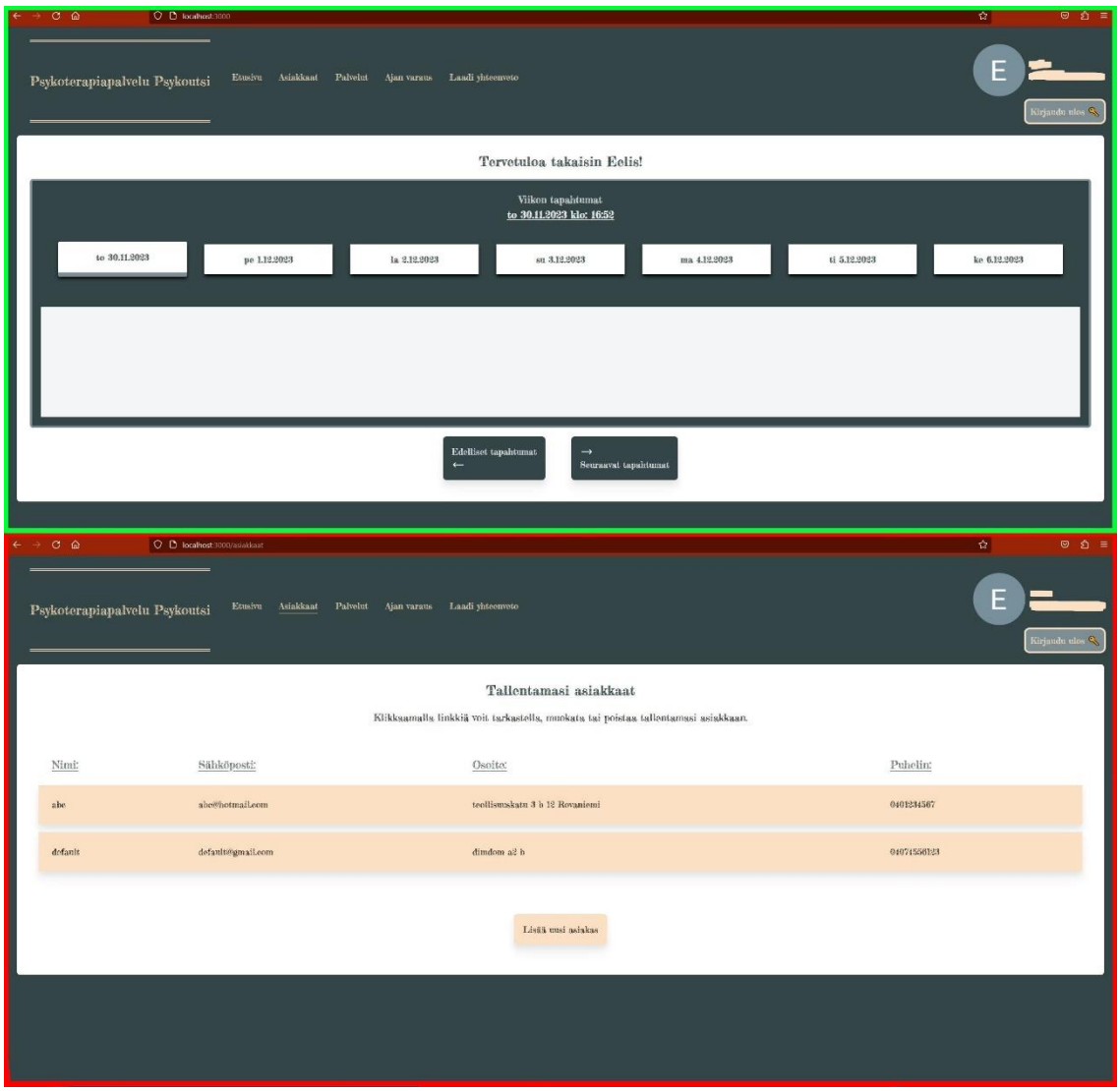
```
import Layout from "@/components/Layout";
import ViikkoKalenteri from "@/components/ViikkoKalenteri";

export default function Home() {
  const {data: session}=useSession();

  return(
  <Layout>
    <h1>Tervetuloa takaisin {session?.user?.name}!</h1>
      <div>
        <ViikkoKalenteri/>
      </div>
  </Layout>
  )
}
```

Kuvio 10. Sovelluksen ”/pages”-kansiossa sijaitseva Index.js-tiedosto eli ohjelman etusivu

Tämä on yksi hyvä esimerkki siitä, mitä tarkoitetaan React-sovellusten ydinajatuksella jakaa sovelluksen toiminta osiin, joita kutsutaan komponenteiksi ja miten saadaan tuotettua ohjelmassa kompaktia ja helposti seurattavaa koodia. Tässä työssä toteutetussa sovelluksessa kaikki sivut noudattavat tätä rakennetta, jossa ’return’-komennon sisällä palautetaan aina Layout-komponentti, mutta Layout-tagien sisällä oleva sisältö muuttuu. Näin saadaan kaikille sivuille yhtenäinen tyyli (vertaa kuvio 11), jossa nähdään vihreä tausta, ylhäällä navigaatiopalkki ja valkoinen ”sivualue”, jonka sisältö vaihtelee sivusta riippuen.



Kuvio 11. Kuvakaappaus sovelluksen etusivusta ja Asiakkaat-sivusta

4.4 Tietokannan implementointi sovellukseen

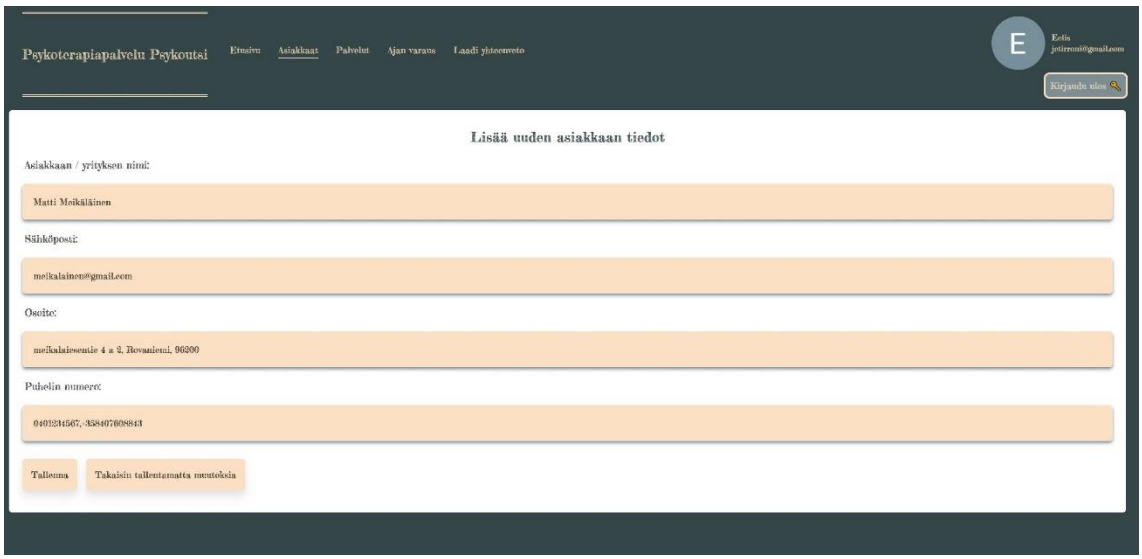
Sovelluksen perimmäisenä ideana on sen sivuja käyttäen kerätä käyttäjän kirjaamaa tietoa ja tallettaa ne MongoDB Atlas-palveluun tehtyyn henkilökohtaiseen tietokantaan. Sovellukseen itsessään ei ole tarkoitus tallentaa mitään tietoja, vaan käyttää rajapintaa, joka välittää tiedot käyttöliittymästä tietokannalle (luodaan tai päivitetään tietoja) ja päinvastoin (tarkastellaan jo tallennettuja tietoja). MongoDB:n kotisivuilla luodaan käyttäjä, minkä jälkeen voidaan luoda helposti tietokanta pilvessä toimivalle palvelimelle. Kun palvelin on asetettu toimimaan si-

vun ohjeita seuraten, saadaan tietokannan asetuksista otettua uniikki URI-merkkijono, jota sovelluksessa käyttämällä pystytään saamaan yhteys sovelluksen ja tietokannan välille.

Yhteyden muodostaminen sovelluksessa tapahtuu tallentamalla saatu URI-merkkijono sovelluksessa ”env”-tiedostoon. Mongodb-moduulin asentamalla npm-komentoa käyttäen (npm install mongodb) saadaan työkalut ja riippuvuudet asennettua sovelluksen ”moduuli”-kansioihin. Nämä työkalut eivät vielä kuitenkaan määrää miten yhteyden muodostaminen tietokantaan tapahtuu, sillä tämä on hyvin yksilöllistä riippuen esimerkiksi tekniikoista ja ohjelmointikielistä, joista oma sovellus koostuu. Yhteyden muodostamiseen tehdään yleensä oma tiedosto sovelluksen ”lib”-kansioon, jossa määritellään tietokantaan yhdistämisen asetukset. Näiden asetelmien jälkeen sovellus on valmis käyttämään Next.js-sovellusten sisäänrakennettua API-reittiä pyyntöjen lähettämiseen tietokantaan.

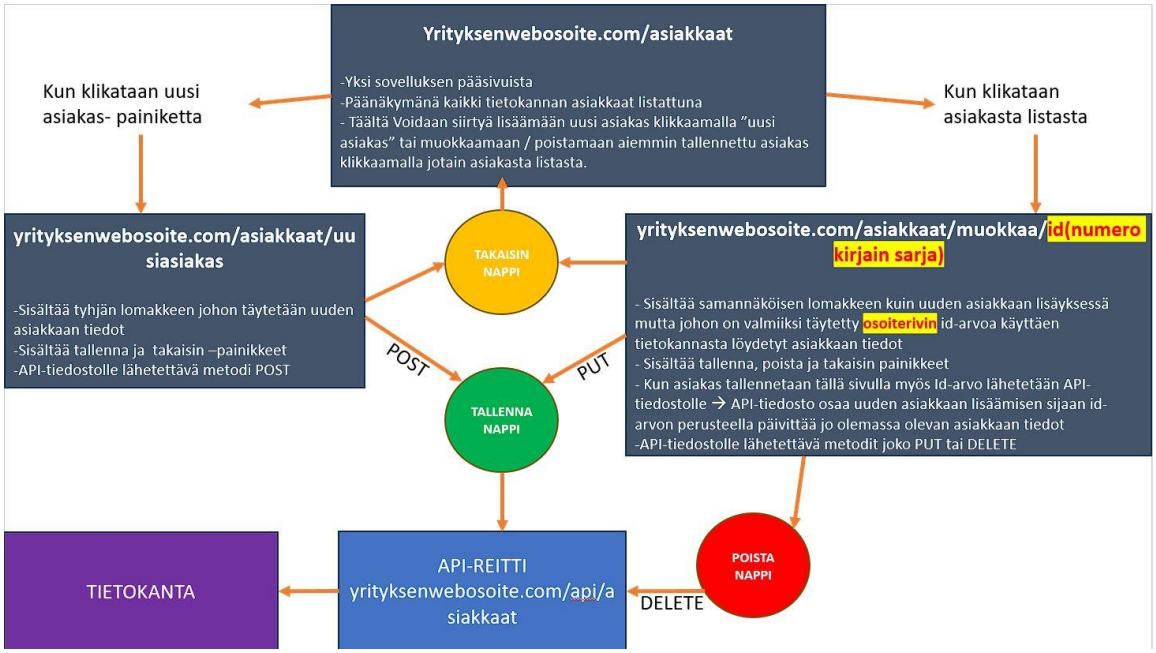
4.4.1 Tietojen kerääminen käyttöliittymässä

Tämän sovelluksen tietojen kerääminen tapahtuu kaikkien (asiakkaat, palvelut, ajanvaraukset) sivujen kohdalla hyvin samanlaista periaatetta käyttäen. Jotta asian selostaminen helpottuisi tarkastellaan tässä asiakkaat-sivua ja asiakastietojen hallinnointia. Käyttäjä voi asiakkaat-sivulla siirtyä luomaan uuden asiakkaan sivulle *’/asiakkaat/uusiasiakas’* klikkaamalla ”lisää uusi asiakas”-painiketta. Kuviossa 12 nähdään uuden asiakkaan lisääminen käyttöliittymässä.



Kuvio 12. Täytettävä lomake käyttöliittymässä

Vaihtoehtoisesti käyttäjä voi siirtyä muokkaamaan tai poistamaan aiemman asiakastiedon ’/asiakkaat/muokkaa/id’-sivulle (id osoiterivissä vastaa klikatun asiakkaan id-arvoa) klikkaamalla asiakkaat-sivulla listattua yksittäistä asiakasta. Molemmissa tapauksissa käyttäjän näkymäksi ilmestyy samaa komponenttia käyttäen identtinen lomakenäkymä. Mikäli kyseessä on uuden asiakkaan luominen, lomake on tyhjä ja tallenna-nappia painaessa lähetetään tiedot komponentissa POST-metodia käyttäen. Mikäli ollaan muokkaamassa aiemmin lisättyä asiakasta, täyttyy lomake automaattisesti osoiterivin id-arvolla tietokannasta löydetyn asiakkaan tiedoilla. Muokkaus-sivulla tallenna-nappia klikatessa tietoihin lisätään myös osoiterivin id-arvo, jolloin komponentti osaa POST-metodin sijaan käyttää PUT-metodia. Kuviossa 13 nähdään kaavassa esitettynä sivuilla liikkuminen.



Kuvio 13. Kaavio asiakkaat-sivun siirtymistä sekä sivuilla tapahtuvista tietojen käsittelystä

4.4.2 API-reitit ja pyynnöt

Next.js-kehyksellä luotujen sovellusten yksi vahvuus on siinä, että ne sisältävät *frontend*- ja *backend*-ominaisuuksia samassa sovelluksessa yhdistelevien projektien tekemistä helpottavia ominaisuuksia, kuten sisäänrakennetut API-reitit. Kun pages/api-kansioon luodaan tiedosto, sitä kohdellaan ”sivu”-tiedoston sijaan API-reittinä, jolle voidaan suorittaa http-pyyntöjä, kuten GET, POST ja PUT. (Vercel, Inc 2023.) Äskeisessä luvussa kerrottiin, miten käyttöliittymästä siirretään tietokantaan tietoa API-reitin avulla. Tässä luvussa keskitytään miten ’pages/api’-kansioon luotu API-reittitiedosto käsittelee API-reitille lähetettyä dataa, jotta tiedonsiirto tietokannan kesken onnistuu. Tässäkin tapauksessa haluan pitää tapauksen selittämisen mahdollisimman selkeänä, joten keskitytään asiakkaat-sivuilla ja asiakastiedoilla tapahtuviin tapahtumiin.

Sovelluksessa käytin tietojen siirtämiseen funktiota, joka lähettää axios-kirjastoa hyödyntäen http-pyynnön haluttuine tietoineen API-reitille. Kuviossa 14 nähdään tässä sovelluksessa toteutettu funktio, joka käynnistyy, kun lomake käyttöliittymässä täydennetään ja klikataan ”*tallenna*”-painiketta. Objekti asiakasData sisältää kaikki käyttäjän lomakkeessa kirjaamat arvot (nimi, email, osoite, puhelin).

Tarkastellaan tässä asiakkaan lisäystä. ’axios.post’ luo HTTP POST-pyynnön API-reitille(’/api/asiakkaat’), jossa se samalla lähettää ’asiakasData’-objektin mukana. Reittiin luodun tiedoston on tarkoitus ottaa vastaan pyyntö ja käyttää sitä tuottamaan vastaus *(res).* API-reittitiedosto pääsee käsiksi asiakasData-muuttujaan ja haluttuun metodiin, koska ne ovat sisällytettynä sille lähetetyssä html-pyynnössä (metodi=req.method ja asiakasData=req.body). Se mitä reittitiedosto palauttaa vastauksena funktiolle ja takaisin käyttäjälle on määritelty API-tiedostossa.

```
async function luoAsiakas(ev){
    ev.preventDefault();
    if (_id){ /*asiakkaan muokkaus*/
        const asiakasData = {nimi,email,osoite,puhelin,_id}
        await axios.put('/api/asiakkaat',asiakasData)
        console.log("Asiakas päivitetty")
    } else { /*asiakkaan lisäys*/
        const asiakasData = {nimi,email,osoite,puhelin}
        await axios.post('/api/asiakkaat', asiakasData)
        console.log("Asiakas lisätty")
    }
```

Kuvio 14. Asiakkaan lisäys-funktio

Kuviossa 15. nähdään esimerkki sovelluksen asiakkaat API-reittitiedoston POST-menetelmästä. Ohjelma tarkistaa, että pyynnössä saatu metodi on POST, jonka jälkeen se purkaa req.bodyssa saamansa tiedon (asiakasData) takaisin yksittäisiin osiin, kuten nimi, osoite, email ja puhelin. Tämän jälkeen uusi asiakas luodaan purettuja tietoja ja tietokantamallia käyttäen (Asiakas.create). Tämän jälkeen se lähettää vastauksena luodun asiakkaan takaisin (asiakasDoc).

```
if (method==='POST'){
    const {nimi,osoite,email,puhelin} = req.body
    const asiakasDoc = await Asiakas.create({
        nimi,osoite,email,puhelin
    })
    res.json(asiakasDoc)

}
```

Kuvio 15. Esimerkki POST-metodista /api/asiakkaat.js-tiedostossa

Esimerkissä käytiin läpi sovelluksen POST-menetelmä, joka lisää uusia asiakkaita tietokantaan. Lopullisessa sovelluksen toiminnassa on myös muita menetelmiä, kuten asiakkaan päivittäminen ja poistaminen. Nämä seuraavat samaa toimintamallia kuin POST-menetelmäkin. Niissä käsitellään joko eri parametrejä tai tietoja muokataan/poistetaan id:n perusteella. Kuviossa 16 esiteltynä käytetyistä eri menetelmistä asiakkaat sivuilla.

Metodi	Kuvaus	Käytettävät parametrit / tiedot	Paluuarvo
GET	Hakee kaikki asiakkaat tai yhden asiakkaan id:n perusteella	req.query.id (valinnainen)	Palauttaa haetut asiakkaat tai yksittäisen asiakkaan (mikäli id on saatavilla) tiedot JSON-muodossa
PUT	Päivittää asiakkaan tiedot	req.body: { nimi, osoite, email, puhelin, _id }	Palauttaa true, kun päivitys on suoritettu onnistuneesti
POST	Luo uuden asiakkaan	req.body: { nimi, osoite, email, puhelin }	Palauttaa luodun asiakkaan tiedot JSON-muodossa
DELETE	Poistaa asiakkaan id:n perusteella	req.query.id	Palauttaa true, kun poisto on suoritettu onnistuneesti

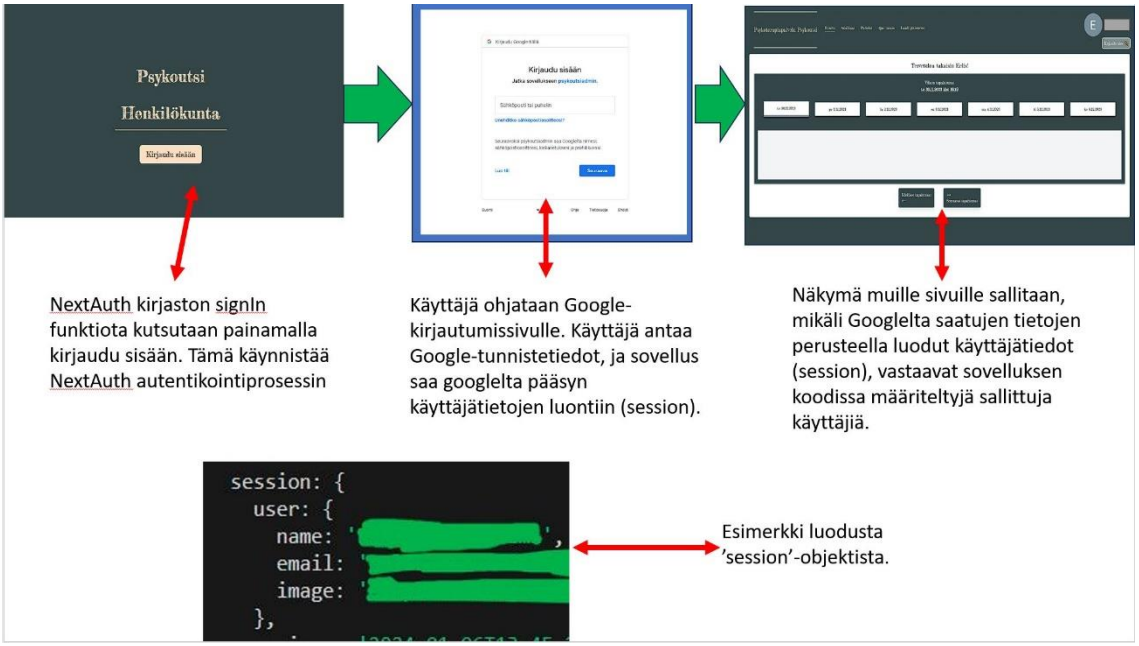
Kuvio 16. Taulukko /api/asiakkaat.js-tiedoston menetelmistä

4.5 Kirjautuminen sovellukseen

Sovelluksen tunnistautumiseen haluttiin käyttää Google-tunnuksia ja sen toteutuksessa päädyttiin käyttämään NextAuth.js-kirjastoa, joka on Next.js-sovelluksille suunniteltu. Se sisältää valmiita ratkaisuja useisiin autentikointiin liittyviin haasteisiin. Se myös sisältää tuen useisiin tietokantapalveluihin, kuten tässä projektissa käytetyn MongoDB:een, jota voidaan käyttää sovelluksen tunnistautumisprosessissa avuksi käyttäjätietojen tallentamiseen. Näistä syistä tämän kirjaston valinta sovelluksen tunnistautumisen luomiseen oli hyvin luonnollinen valinta, enkä kokenut, että olisi ollut tarvetta kehitellä omakehitteistä tunnistautumistapaa tähän tarkoitukseen.

NextAuth.js-ratkaisujen implementointi sovellukseen ei vaatinut erityisesti mainittavaa työskentelyä. Kirjasto sisältää esimerkiksi ’signIn’- ja ’signOut’-funktiot, jotka toimivat pitkälti sellaisenaan, kun NextAuth.js-tiedosto konfiguroidaan NextAuth-käyttöohjeiden mukaan. Kirjautumis-sivulle loin sovelluksessa ”kirjautumis”-painikkeen, joka ohjaa käyttäjän ’signInä funktiolla Googlen tunnistautumis- sivulle. Suoritettuaan Google kirjautumisen käyttäjälle luodaan session objektiin tiedot kyseisestä Google-tilistä. Salatut sivut voidaan näyttää, mikäli tämän session nimisen objektin tiedot vastaavat turvalliseksi luokitellun henkilön Google-tunnus tietoja. (Ruffles 2023.)

’signOut’-funktiota kutsutaan luonnollisesti ’kirjaudu ulos’-painikkeella, ja se poistaa käyttäjän kirjautumisprosessin aikana luodut tiedot taustalla ylläpidettävistä objekteista (session), jolloin näkymä jälleen sulkeutuu muille sivuille. Kuviossa 17 nähdyn ’session’-objektin tietoja voidaan käyttää muuhunkin tarkoitukseen sovelluksen toiminnassa, kuten tervehtimään kirjautunutta käyttäjää (Tervetuloa {user.name}!). Näin saadaan sivusto tervehtimään aina sillä hetkellä kirjautuneena olevaa käyttäjää.



Kuvio 17. Kirjautuminen käyttöliittymässä

5 POHDINTA

Opinnäyte koostui yrityksen toimintojen ja tapahtumien kirjaamiseen ja seurantaan suunnitellun sovelluksen luomisesta. Sovellus koostui verkkoselaimella toimivasta käyttöliittymästä, jolla kommunikoitiin pilvessä olevan tietokannan kanssa. Työn toteuttamisen keskeisimpiä aiheita olivat asiakasyrityksen kanssa palvelun suunnittelu sekä suunnittelussa kerätyn tiedon perusteella työn tekninen toteuttaminen. Työn tekniseen toteuttamiseen käytettiin pääasiassa JavaScript-kieltä ja Next.js-kehystä.

Projektissa tavoitteet ja sovellukselta haluttavat ominaisuudet tulivat toimeksiantajayrityksen kanssa käydyssä suunnitteluosuudessa hyvin esille ja mielestäni tässä työssä saatu lopputulos vastaa niitä varsin hyvin. Sovelluksessa toiminnallisuuksien osalta sovelluksella pystytään käsittelemään juuri niitä tietoja, mitä siltä alussa odotettiinkin ja tunnistautuminen on toteutettu yrittäjän toiveiden mukaan. Myös ulkoasuvaatimukset valmiin sovelluksen osalta vastaavat suunnitteluosuudessa suunnittelutyökalulla yhteisesti tehtyä hyväksi ja selkeäksi todettua lopputulosta. Työn toteutukseen valitut tekniikat ja toimintamallit ohjelmoinnissa mahdollistavat myös sen, että kaikkien näiden ominaisuuksien muokkaaminen mielen muuttuessa tapahtuu erittäin joustavasti ja helposti.

Vaikka sovellus tulikin opinnäytetyössä valmiiksi, ei sitä vielä voitu tämän opinnäytetyön aikana ottaa käyttöön yrityksen varsinaisessa liiketoiminnassa. Sovellusta pystyttiin kuitenkin yrittäjän kanssa kokeilemaan testitiedoilla ja palaute sen toiminnasta oli hyvä. Palautteen mukaan oli myös itsestään selvää, että se tehostaa yrityksen ajankäyttöä näiden toimintojen osalta aiempaan nähden, sillä tähän tarkoitukseen yrityksellä ei aiemmin ollut erityistä työkalua. Jatkokehitystä ajatellen yrityksen tähtäimessä on tulevaisuudessa kehitellä omat nettisivut yleisön käyttöön koskien osittain samoja toimintoja ja tietoja, joita tässäkin sovelluksessa käsitellään. Yleisölle luodun sivun luonnissa voitaisiin hyödyntää tällä sovelluksella tallennettua dataa helposti. Asiakkaat voisivat esimerkiksi itse luoda omat käyttäjätietonsa sekä ajanvarauksensa, jonka yhteydessä näiden tiedot liikkuisivat samaan tietokantaan kuin tässä sovelluksessa. Näin esimerkiksi kalenterinäkymä päivittyisi myös asiakkaiden tekemien toimien perusteella tehden alustasta entistäkin automatisoidumman.

Opinnäytetyön edetessä huomasin, että vaikka suunnitteluosuus vei odotettua enemmän aikaa, se säästi aikaa teknisessä toteutuksessa. Hyvin toteutettu suunnitteluosuus ja toimeksiantajayrityksen toiveiden ja mieltymysten tarkka selvitys, vei projektia joutuisasti eteenpäin. Hyvä luonnos halutusta lopputuloksesta ennen ohjelmoinnin aloitusta ja toimeksiantajan kanssa joustavasti käyty palautekeskustelu tuki, että teknisessä toteutuksessa on koko ajan selkeä suunta, jota kohti pyrkiä. Huomasin myös, että sovelluskehityksessä ja ohjelmoinnissa on paljon valmiiden kirjastojen, kehysten ja eri työkalujen kytkemistä ja ohjelmointia oman sovelluksen tueksi sen sijaan, että ohjelmoisi kaiken toiminnallisuuden itse. Mikäli valmiita ratkaisuja juuri omaan tarkoitukseen on tarjolla ja vapaasti saatavilla niitä voi ja kannattaa käyttää kehittämisen apuna.

LÄHTEET

Bondar, S. 2023. What is React JS? Viitattu 25.11.2023 https://rein-tech.io/blog/what-is-react-js-guide-for-software-developers

Chauhan, S. 2023. Steps in Choosing the Right Programming Language for Your Project. Viitattu 25.11.2023 https://www.scholarhat.com/tuto-rial/techtrends/choosing-the-right-programming-language-for-your-project

Forbes Technology Council 2018. 11 Questions To Ask Yourself Before Creat-ing An App For Your Business. Viitattu 28.11.2023 https://www.forbes.com/si-tes/forbestechcouncil/2018/12/12/11-questions-to-ask-yourself-before-creating-an-app-for-your-business/

Gillis, A. 2018. Integrated development environment (IDE). Viitattu 25.11.2023 https://www.techtarget.com/searchsoftwarequality/definition/integrated-develop-ment-environment

GIMP 2023. Introduction to GIMP. Viitattu 1.1.2024 https://www.gimp.org/about/introduction.html

Harvie, L. 2018. Version Control — Why Do We Need It? Viitattu 26.11.2023 https://medium.com/@lanceharvieruntime/version-control-why-do-we-need-it-1681f4888cec

Lazuardy, M., & Anggraini, D. 2022. Modern Front End Web Architectures with React.Js and Next.Js. Viitattu 30.11.2023 Saatavilla sähköisesti osoitteessa http://irjaes.com/wp-content/uploads/2022/02/IRJAES-V7N1P162Y22.pdf

Mozilla Corporation. 2023. The web and web standards. Viitattu 24.11.2023 https://developer.mozilla.org/en-US/docs/Learn/Getting_star-ted_with_the_web/The_web_and_web_standards

Niżyński, D. 2023. What is Next JS and Why Should You Use it in 2023? Viitattu 25.11.2023 https://pagepro.co/blog/what-is-nextjs/

Ruffles, J. 2023. Introduction. Viitattu 30.11.2023 https://next-auth.js.org/getting-started/introduction

Schaefer, L. 2023 MongoDB What is NoSQL? Viitattu 5.12.2023 https://www.mongodb.com/nosql-explained

Vercel Inc. 2023. API Routes. Viitattu 30.11.2023 https://nextjs.org/docs/pa-ges/building-your-application/routing/api-routes