LAPIN AMK
Lapland University of Applied Sciences

HEADLESS WORDPRESSIN JA NEXT.JS:N INTEGRAATIO

Kyrölä Eemeli

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2024

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Koulutus
Tutkintonimike

Tekijä	Eemeli Kyrölä	**Vuosi**	2024
Ohjaaja	Aku Kesti		
Toimeksiantaja	Erja Kyrölä		
Työn nimi	Headless WordPressin ja Next.js:n integraatio		
Sivumäärä	28		

Opinnäytetyöni tarkoituksena oli kehittää verkkosivu, jonka avulla toimeksiantajan palvelut tulevat paremmin esille ja helpommin löydettäväksi verkossa. Verkkosivu toteutettiin hyödyntäen Headless-teknologiaa, jossa backendinä toimii WordPress ja frontendinä Next.js. Opinnäytetyöni toimeksiantajana toimii Kyrola Consulting Oy.

Opinnäytetyö toteutettiin kehittämispainotteisena opinnäytetyönä, jossa otettiin huomioon koko verkkosivun kehittämiskaari verkkosivun suunnittelusta kehittämiseen asti. Teknologioiden taustoissa käydään läpi kaikki teknologiat, joita opinnäytetyössä hyödynnettiin. Keskeisenä osana verkkosivustoa oli WordPressin ja Next.js:n integraatio, jossa WordPressissä hallinnoitu sisältö tuodaan näyttävästi esille Next.js-sivustolla Vercen pilvipalvelimella.

Opinnäytetyön lopulliseksi lopputulukseksi tuli Kyrola Consulting Oy:lle valmis verkkosivu, jonka kautta toimeksiantajan palvelut ovat helposti ja selkeästi saatavilla verkossa.

Avainsanat Headless-teknologia, Next.js, Vercel, WordPress

LAPIN AMK
Lapland University of Applied Sciences

Abstract of the thesis

Degree programme
Degree title

Author	Eemeli Kyrölä	**Year**	2024
Supervisor	Aku Kesti		
Commissioned by	Erja Kyrölä		
Title	The integration of Headless WordPress and Next.js		
Number of pages	28		

The purpose of my thesis was to develop a website that showcases the client's services more effectively and makes them easier to find online. The website is implemented using Headless technology, with WordPress as the backend and Next.js as the frontend. Kyrola Consulting Oy served as the client for my thesis.

The thesis was conducted as a development-oriented project, considering the entire development cycle of the website from planning to implementation. The background sections on technologies cover all the technologies utilized in the thesis. A key aspect of the website was the integration of WordPress and Next.js, where content managed in WordPress is prominently displayed on the Next.js site hosted on the Vercel cloud platform.

The final outcome of the thesis was a completed website for Kyrola Consulting Oy, through which the client's services are easily and clearly accessible online.

Keywords Headless technology, Next.js, Vercel, WordPress

SISÄLLYS

1 JOHDANTO 5

2 TEKNOLOGIOIDEN TAUSTAT 6

2.1 WordPress-sisällönhallintajärjestelmänä 6
2.2 Vercelin dynaaminen web 6
2.3 Next.js frontend-kehityksessä 7
2.4 Healdess-teknologia 8

3 WORDPRESS-TEEMAN KEHITTÄMINEN 10

3.1 Yksilöllisen teeman kehittäminen 10
3.2 Local WP:n käyttö teeman kehityksessä 11
3.3 Headless-teknologian vaikutus kehitysprosessiin 12
3.4 Räätälöidyn sisältötyypin kehittäminen 13
3.5 Perussivujen rakentaminen WordPressissä 14
3.6 Navigaation ja footerin hallinta WordPressissä 15

4 INTEGRAATIO PROSESSI 16

4.1 Datan hakeminen WordPressistä 16
4.2 Ylävalikon ja alavalikon kehitys 17
4.3 Sivujen kehittäminen 19
4.4 Yhteydenottolomakkeen kehittäminen 21
4.5 Testaus 22
4.6 Headless-integraation edut ja haasteet 24

5 POHDINTA 26

LÄHTEET 28

1 JOHDANTO

Projektin aloitustilanne on sellainen, että asiakkaalla ei ole voimassa olevaa verkkosivustoa, mutta hänellä on olemassa oleva WordPress-asennus domain-hotellissa. Ottaen huomioon toimeksiantajan tarpeet ja olemassa olevan infrastruktuurin, päätin valita backendiksi WordPressin. Tämä päätös perustui WordPressin käyttäjäystävällisyyteen ja sen tarjoamiin laajoihin mahdollisuuksiin sisällönhallintaan ja julkaisuun.

Tämän opinnäytetyön aiheena on kertoa, miten pystyy kehittämään ja julkaisemaan verkkosivuston Headless-ratkaisulla, jossa frontend toteutetaan Next.js:n avulla ja WordPress toimii backendinä. Tällainen lähestymistapa antaa mahdollisuuden hyödyntää WordPressin helppokäyttöistä sisällönhallintaa ja samalla tarjota moderni ja joustava käyttöliittymä käyttäjille.

Headless-teknologian valinta tarjoaa useita etuja verrattuna normaaliniin WordPressiin, jossa backend ja frontend toimivat yhdessä. Headless-teknologia mahdollistaa sisällön joustavan esittämisen eri alustoilla ja laitteilla. Parantaa sivuston suorituskykyä ja nopeutta, erityisesti mobiililaitteilla. Headless-ratkaisu antaa mahdollisuuden hyödyntää moderneja frontend-teknologioita, kuten Reactia tai Angularia, mikä voi parantaa sivuston käyttökokemusta entisestään.

Tässä opinnäytetyössä keskitytään siihen, miten yhdistetään WordPressin sisällönhallinta ja Next.js:n tuoma joustavuus ja suorituskyky Headless-teknologian avulla. Pyrkimyksenä on tarjota käytännönläheisiä ohjeita, jotka auttavat muita kehittäjiä ja sivuston ylläpitäjiä hyödyntämään Headless-teknologiaa parhaalla mahdollisella tavalla.

2 TEKNOLOGIOIDEN TAUSTAT

Tässä luvussa tarkastellaan olennaisia tekijöitä, jotka ovat keskeisessä roolissa verkkosivuston toteutuksessa. Pohdin WordPressin sisällönhallintajärjestelmän merkitystä, tutkin Next.js:n ja Vercelin tarjoamia moderneja kehitysmahdollisuuksia sekä syvennyn Headless-teknologian rooliin. Käyn läpi näitä aiheita, koska ne ovat olennainen osa valintojani ja käytäntöjäni kehitysprosessissa.

2.1 WordPress-sisällönhallintajärjestelmänä

Sisällönhallintajärjestelmästä käytetään yleisesti lyhennettä CMS (Content Management System). CMS mahdollistaa käyttäjän päivittää verkkosivun sisältöä ilman teknistä osaamista. (Kinsta Inc. 2024a.) Yksi suosituimmista CMS-alustoista on WordPress. WordPress perustuu avoimeen lähdekoodiin, jonka ansiosta alusta on käyttäjille ilmainen. Arvioilta yli 42 prosenttia kaikista verkkosivuista on kehitetty WordPressin ympärille. (Kinsta Inc. 2024b.)

WordPressin suosio pohjautuu pitkälti palvelun helppokäyttöisyyteen, monipuolisiin lisäosiin ja valmiisiin teemoihin. WordPress-teema on visuaalinen ulkoasu, joka määrittelee verkkosivuston ulkonäön ja tyylin WordPress-sisällönhallintajärjestelmässä. Teemat vaikuttavat sivuston ulkoiseen ilmeeseen, kuten värimaailmaan, fontteihin, layoutiin ja navigointiin. Ne tarjoavat helpon tavan muokata sivuston ulkoasua ilman syvällistä ohjelmointiosaamista. (Maulidina 2024.) WordPressillä on laaja valikoima ilmaisia ja maksullisia teemoja eri käyttötarkoituksiin, kuten bloggaamiseen, yrityssivustoihin, portfolioihin ja verkkokauppoihin. Käyttäjät voivat vaihtaa teemaa milloin tahansa säilyttäen samalla sisällön ja rakenteen.

2.2 Vercelin dynaaminen web

Vercel on nykyaikainen ja edistyksellinen pilvipalvelu, joka tarjoaa tehokkaan alustan web-kehitykseen. Sen pääpaino on tarjota kehittäjille nopea ja vaivaton tapa julkaista ja hallita web-sovelluksia. (Vercel, Inc. 2024a.) Vercelin keskeiset piirteet tekevät siitä suositun valinnan kehittäjien keskuudessa.

Vercel tunnetaan erityisesti huippunopeasta automaattisesta skaalautuvuudestaan. Palvelu tarjoaa CDN-tuen (Content Delivery Network), mikä tarkoittaa, että sovellukset ovat optimoituja suorituskyvyn ja latausnopeuden kannalta kaikkialla maailmassa. (Vercel, Inc. 2024b.)

Vercel ei ainoastaan tarjoa huippunopeaa automaattista skaalautuvuutta, vaan myös korkeatasoista turvallisuutta kehittäjille ja heidän sovelluksilleen. Palvelu sisältää useita turvallisuustoimintoja, jotka auttavat suojaamaan sovelluksia haitallisilta hyökkäyksiltä. Vercel hyödyntää edistyksellistä turvallisuusmekanismeja, kuten HTTPS-tukea, joka varmistaa salatun yhteyden käyttäjän ja palvelimen välillä. (Vercel, Inc. 2024c.) Tämä on olennainen tekijä sekä käyttäjien yksityisyyden, että tietojen eheyden turvaamiseksi. Lisäksi Vercel tarjoaa turvallisen ja helposti hallittavan ympäristön salasanojen ja muiden arkaluontoisten tietojen käsittelyyn. Palvelu toteuttaa monipuolisia suojausmekanismeja estääkseen mahdolliset tietoturvauhat ja noudattaa alan parhaita käytäntöjä. (Vercel, Inc. 2024c.)

Vercelin helppokäyttöisyys on myös keskeinen tekijä sen suosiossa. Kehittäjät voivat julkaista sovelluksensa vain muutamalla klikkauksella, ja Vercel tarjoaa intuitiivisen käyttöliittymän sekä monipuolisia työkaluja projektien hallintaan.

2.3 Next.js frontend-kehityksessä

Next.js on avoimen lähdekoodin frontend framework, jonka on kehittänyt Vercel. Next.js on kehitetty Rect frameworkina, laajentaen Reactin ominaisuuksia ja tarjoten kehittäjille tehokkaita työkaluja modernin web-kehityksen tarpeisiin. (Vercel, Inc. 2024b.)

Yksi Next.js:n keskeisistä piirteistä on mahdollisuus toteuttaa Server-Side Rendering (SSR). Tämä tarkoittaa, että sivuston sisältö voidaan generoida palvelimella, mikä parantaa merkittävistä sivuston suorituskykyä ja edistää tehokasta hakukoneoptimointia. SSR mahdollistaa käyttäjille nopeamman latausajan ja paremman käyttökokemuksen sivustolla. (Vercel, Inc. 2024e.)

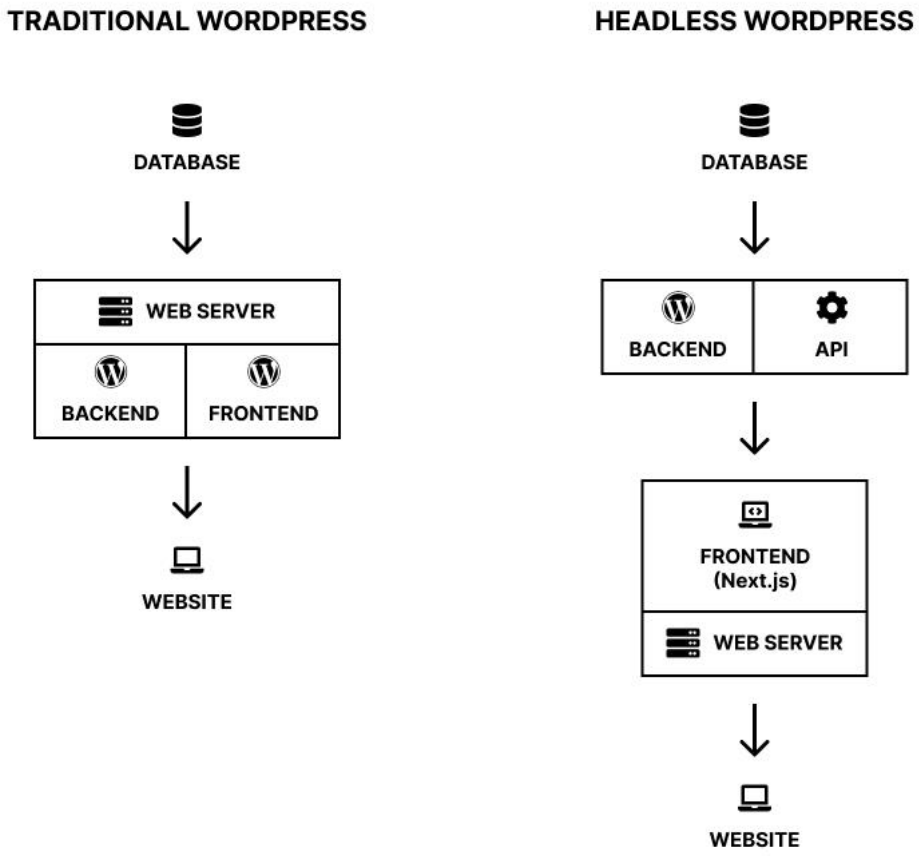
Next.js tarjoaa myös kyvyn generoida staattisia sivuja, sekä mahdollisuuden rakentaa dynaamisia sovelluksia tarpeen mukaan. Staattiset sivut luodaan

etukäteen ja sivun sisältö pysyy samana jokaisella latauskerralla. Staattinen sivu ladataan valmiina tiedostona suoraan palvelimelta käyttäjän selaimelle. (Vercel 2024f.) Dynaaminen sivu sen sijaan luodaan palvelimella joka kerta, kun käyttäjä päivittää sivun. Dynaamiset sivut ovat joustavampia ja voivat tarjota yksilöllisiä käyttökokemuksia käyttäjän ja tilanteen mukaan. (Vercel, Inc. 2024g.) Staattiset sivut sopivat paremmin tilanteisiin, joissa sisältö on harvemmin muuttuva.

Lisäksi Next.js erottuu automaattisella optimoinnillaan, erityisesti kuvien, fonttien ja muiden resurssien latautumisessa. (Vercel, Inc. 2024h.) Tämä ominaisuus parantaa merkittävästi käyttäjäkokemusta vähentämällä latausaikoja ja optimoimalla resurssien käyttöä. Kehittäjät voivat luottaa siihen, että Next.js huolehtii automaattisesti monista suorituskykyyn liittyvistä näkökohdista muun muassa kuvien lataamisen nykyaikaiseen muotoon WebP ja AVIF, mikä helpottaa laadukkaiden ja tehokkaiden web-sovellusten kehittämistä. (Vercel, Inc. 2024i.)

2.4 Healdess-teknologia

Headless-teknologia viittaa lähestymistapaan, jossa verkkosivun tai sovelluksen frontend ja backend erotetaan toisistaan (kuvio 1). Perinteisessä verkkosivustossa frontedn ja backend ovat tiiviisti kytkettyjä toisiinsa. Headless-arkkitehtuurissa CMS-järjestelmä tarjoaa sisällöstä tietoa API-rajapintojen (Application Programming Interface) kautta. Frontend puolestaan hakee ja näyttää haetun datan dynaamisesti. (Knowit Oy 2024.)



Kuvio 1. Normaalin WordPressin ja Headless WordPressin eroavaisuudet

Headless-teknologian hyödyt ovat moninaiset ja tarjoavat merkittävää parannusta verkkosivujen ja sovellusten kehityksessä. Joustavuus ja monipuolisuus ovat keskeisiä etuja, kun Headless-ratkaisu mahdollistaa modernien frontend-teknologioiden käyttämisen. Monikanavaisuus tekee Headless-tekniikasta ihanteellisen monipuolisen sisällön jakeluun. API-rajapintojen kautta haettu data mahdollistaa sisällön käytön eri kanavilla, kuten verkkosivustolla, mobiilisovelluksessa ja muissa älylaitteissa. (Knowit Oy 2024.)

3 WORDPRESS-TEEMAN KEHITTÄMINEN

3.1 Yksilöllisen teeman kehittäminen

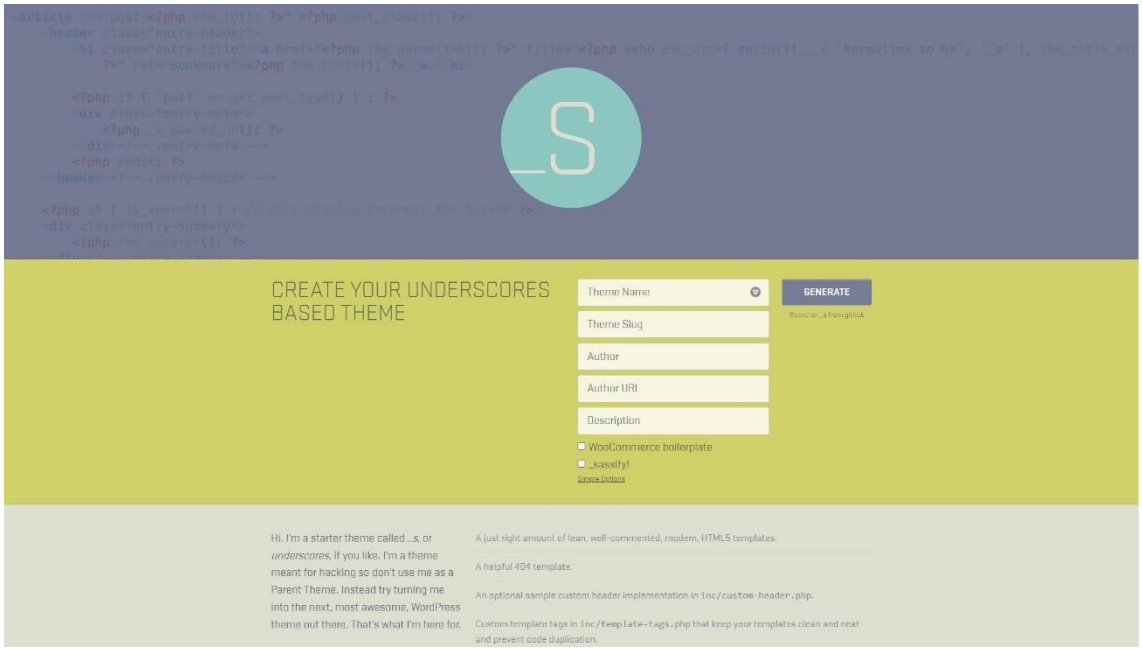
Oman teeman luominen WordPressiin tarjoaa ainutlaatuisen mahdollisuuden vaikuttaa suoraan sivuston rakenteeseen ja visuaaliseen ilmeeseen. Tämä on erityisen tärkeää, kun tavoitellaan Headless-ratkaisua WordPressin kanssa. Luomalla oman teeman pystyin hallitsemaan ja optimoimaan sivuston rakenteen nimenomaan Healdess-arkkitehtuuria varten. Tiesin, miten sivuston data lopulta haetaan WordPressistä ja esitellään frontendissä. Pystyin suunnittelemaan teeman siten, että se tukee API-rajapintojen tehokasta käyttöä. (Olawanle 2023.)

WordPress tarjoaa valmiina REST API:n, joka mahdollistaa sisällönhallinnan ja käsittelyn ulkoisista sovelluksista tai palveluista käsin. REST-API toimii perinteisenä tapana hakea ja muokata dataa verkkopalvelusta HTTP-pyyntöjen avulla. Se tarjoaa valmiit reitit eri tietotyyppien, kuten sivujen, artikkeleiden ja käyttäjien käsittelyyn. (WordPress.org 2024.)

GraphQL on toinen API-tyyppi, joka tarjoaa joustavamman lähestymistavan datan hakemiseen ja muokkaamiseen. Sen avulla käyttäjät pystyvät pyytämään tarkalleen tarvitsemansa datan yhdellä pyynnöllä, mikä voi parantaa suorituskykyä ja vähentää tarpeetonta datan siirtoa. GraphQl ei sido käyttäjiä tiukasti tietyn datan rakenteeseen, vaan he voivat pyytää haluamansa kentät ja liitetyt tiedot dynaamisesti. (Gavald 2023.)

Lähdin tietoisesti kehittämään omaa teemaani valitsemalla UNDERSCORES-teeman, siinä on valmiina vain välttämättömät tiedostot ja perustoiminnallisuudet. Teema toimii eräänlaisena aloitusteemana kehittäjille tarjoten valmiin sivuston perusrakenteen. Valmiit tyylitiedostot, sivupohjat ja widget-alueet toimivat hyvänä pohjana, josta lähteä liikenteeseen teeman kehitystyössä. Tämä mahdollistaa keskittymisen olennaiseen eli oman teeman kehittämiseen sen sijaan, että olisin luonut perusrakenteet alusta asti.

Lisäksi UNDERSCORES -teeman luominen on vaivatonta teeman omalta sivulta käsin. Teeman generoinnissa on mahdollista antaa teemalle nimi, kehittäjän nimi ja muita tarvittavia tietoja, mikä tekee prosessista erityisen käyttäjäystävällisen ja räätälöitävän (kuvio 2). Näiden ominaisuuksien ansiosta useat kehittäjät ja yritykset käyttävät kyseistä teemaa pohjana kehittäessään itselleen WordPress-teemaa.

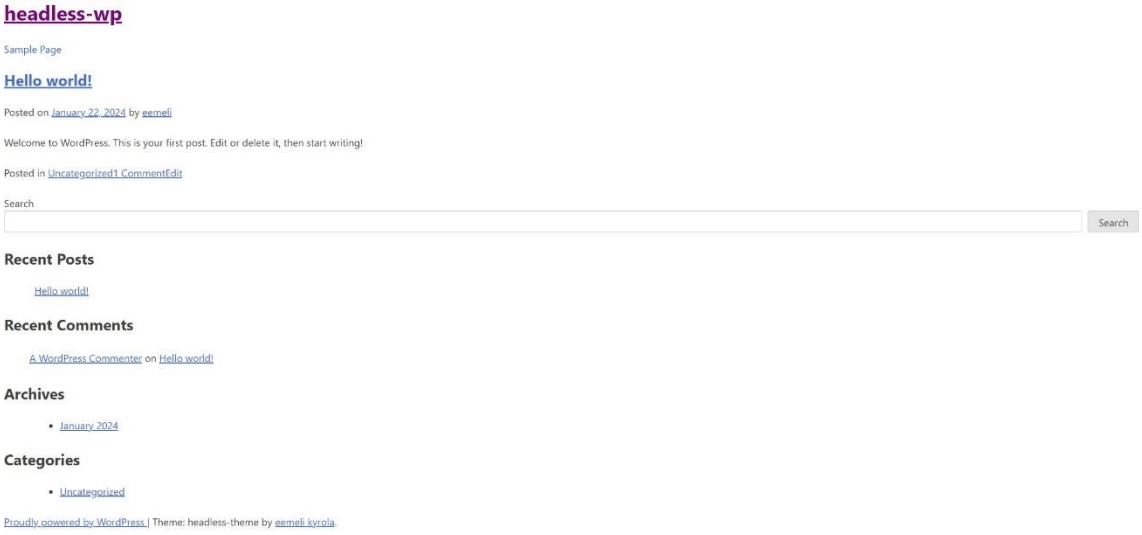


Kuvio 2. UNDERSCORES-teeman luominen

3.2 Local WP:n käyttö teeman kehityksessä

Oman teeman luomisen jälkeen siirryin Local WP -kehitysympäristöön kehittämään WordPress-teemaa. Local WP:n avulla pystyin kehittämään ja testaamaan teemaani ja sivustoa paikallisesti ennen sen julkaisua live-ympäristöön. (Local WP 2024.) UNDERSCORES-teema on suunniteltu toimimaan pohjana kehittäjille, jotta he voivat luoda täysin räätälöidyn ulkoasun ja toiminnallisuuden tarpeidensa mukaan. Siksi WordPressin hallintapuolella ulkoasu saattaa näyttää varsin yksinkertaiselta ja perustavanlaatuiselta, kun UNDERSCORES-teema on otettu käyttöön (kuvio 3). Oman teeman luomisen jälkeen siirryin Local WP -kehitysympäristöön kehittämään sivuston rakennetta. Local WP:n avulla pystyin kehittämään ja testaamaan teemaani ja sivustoa paikallisesti ennen, sen julkaisemista live-ympäristöön. (Local WP 2024.) UNDERSCORES-teema on sunniteltu toimimaan pohjana kehittäjille, jotta he voivat luoda täysin räätälöidyn ulkoasun

ja toiminnallisuuden tarpeidensa mukaan. Siksi WordPressin hallintapuolella ulkoasu saattaa näyttää varsin yksinkertaiselta ja perustavanlaatuiselta, kun UNDERSCORES-teema on otettu käyttöön.



Kuvio 3. UNDERSCORES-teeman ulkoasu

3.3 Headless-teknologian vaikutus kehitysprosessiin

Keskityin pääasiassa WordPressin-hallintapuolen toiminnallisuuksien ja sivuston rakenteen kehittämiseen. Tämä valinta perustui siihen, että sivusto tulee toimimaan Headless-teknologialla. Headless-ratkaisussa WordPress toimii erillisenä CMS-alustana, kun taas sivuston käyttöliittymän kehitin Next.js teknologialla ja julkaisin Vercelin palvelimella. Keskittymällä teeman toiminnallisuuksiin varmistin, että sivusto integroituu saumattomasti valitsemieni teknologioiden kanssa.

WordPressin hallintapuolen kehittämisessä hyödynsin Custom Post Type UI- ja Advanced Custom Field -lisäosia. Näiden lisäosien avulla hallitsin helposti tallennettua dataa WordPressin tarjoaman API:n kautta, mikä mahdollisti datan käytön ja integroinnin verkkosivun käyttöliittymään.

Custom Post Type UI (CPT UI) on kätevä WordPress-lisäosa, joka helpottaa mukautettujen sisältötyyppien, kuten projektien, referenssien tai tapahtumien, luomista (WebDevStudios 2024.)

Advanced Custom Field (ACF) puolestaan tarjoaa laajennetut mahdollisuudet räätälöityjen kenttien luomiseen WordPress-sivustolla. ACF:n avulla pystyin lisäämään erillaisia kenttiä, kuten teksti- ja kuvakenttiä sekä valintaruutuja, mukautetuille sisältötyypeille. (WPENGINE, INC 2024.) Tämä tekee sisällönhallinnasta joustavampaa ja mahdollistaa monipuolisen datan tallentamisen WordPress-sivustolla.

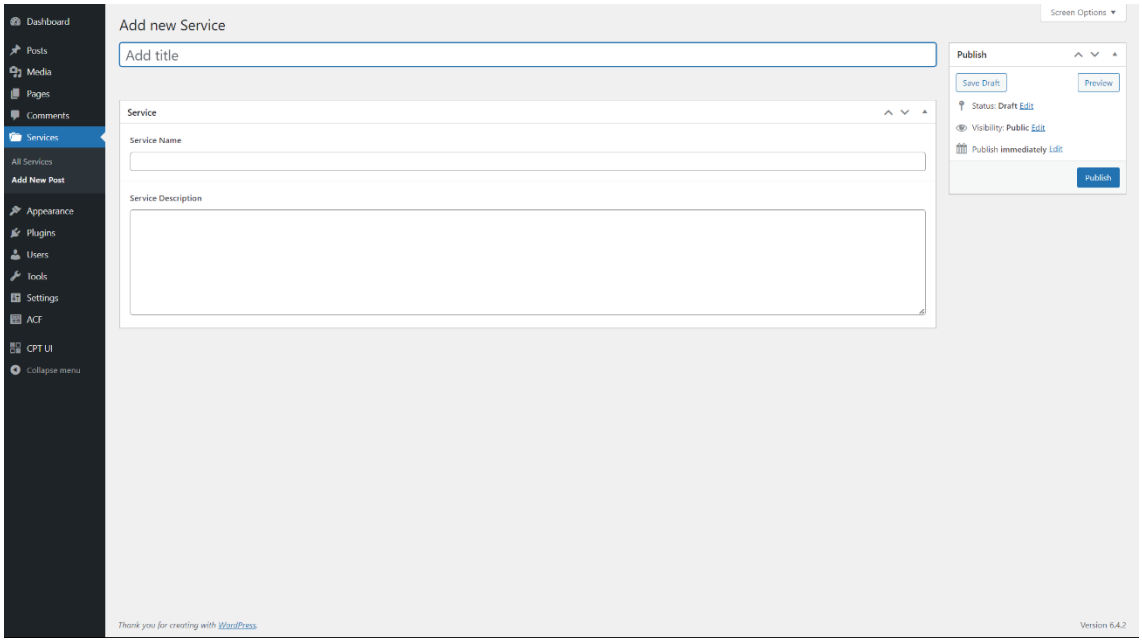
3.4 Räätälöidyn sisältötyypin kehittäminen

Aloitin teeman sisällön kehittämisen ensimmäisenä luomalla toimeksiantajalle omat sivut, jotka keskittyvät yrityksen tarjoamiin palveluihin. Käytin tähän tarkoitukseen CPT UI -lisäosaa, joka mahdollisti räätälöidyn Services-sisältötyypin kehittämisen. Tämä osio tarjoaa kätevän tavan esitellä ja järjestellä selkeästi yrityksen palveluvalikoimaa. Samalla toimeksiantaja voi vaivattomasti lisätä, muokata ja poistaa palveluita tarpeen mukaan. Tämä helpottaa pitämään sivuston kävijät aina ajan tasalla yrityksen palveluista.

CPT UI:lla luotu Services-osio ei tarjoa lainkaan sisältökenttiä sivustolle, lukuun ottamatta otsikon lisäämistä. Otsikon lisääminen automaattisesti luo palvelulle vastaavan URL-osoitteen, mikä helpottaa sisällöntuottajan työtä, sillä hänen ei tarvitse itse luoda jokaiselle sivulle URL-osoitetta.

Siksi hyödynsin ACF-lisäosaa jatkaakseni sisällön kehittämistä. ACF:n avulla loin kaksi erillistä kenttää Services-osiolle, joissa toimeksiantaja tai sivuston sisällön päivittäjä voi lisätä palvelun nimen ja sen sisällön. ACF:n asetuksista valitsin, että kyseisiä kenttiä käytetään ainoastaan Services-tyypin sisällöissä. Tämän avulla pystyin varmistamaan, että kyseisiä osioita käytetään pelkästään toimeksiantajan palveluiden esittelyssä.

Kuten alla olevasta kuvasta näkee, sivuston sisällönpäivittäjän on nyt mahdollista lisätä palvelulle nimi ja kuvaus (kuvio 4). On huomionarvoista, että kun käyttäjä luo uuden palvelun, nämä kentät tulevat aina näkyville. Tämän ansiosta jokainen palvelu sisältää samat sisältökentät, mikä tekee palveluiden muokkaamisesta ja lisäämisestä vaivatonta. Vaikka jokainen palvelu jakaa samat perustiedot, kuten nimen ja kuvauksen, sisältö näissä kentissä voi olla yksilöllinen kullekin palvelulle. Tämä mahdollistaa tehokkaan ja yhtenäisen sisällönhallinnan samalla tarjoten joustavuutta palveluiden yksilölliseen esittelyyn.



Kuvio 4. Services-sivun sisältökenttä

3.5 Perussivujen rakentaminen WordPressissä

Seuraavaksi suuntasin huomioni jokaisen perussivun luomiseen WordPress-sivustollani, kuten etusivu, tietoja meistä, palvelut ja yhteystiedot. Halusin varmistaa sivujen sisällön joustavuuden ja helpottaa tulevia muokkauksia, joten loin jokaiselle sivulle oman ACF-kentän. Näillä räätälöidyillä ACF-kentillä säilytin tarvittavien sivujen elementtien tiedot paikoillaan, mikä mahdollistaa vaivattoman sisällön muokkaamisen ja varman datan hakemisen API:n kautta. Tämä strategia antoi mahdollisuuden päästä käsiksi jokaiseen sivuston elementtiin erikseen, mikä helpotti yksilöllisten muutosten tekemistä ja takasi sivuston joustavuuden tulevaisuudessa. Kun kentät olivat valmiina, sain helposti lisättyä toimeksiantajan antaman sisällön jokaiselle sivulle.

3.6 Navigaation ja footerin hallinta WordPressissä

Navigaatio ja footer muodostavat keskeisen osan verkkosivuston käyttäjäkokemuksesta ja käytettävyydestä. Navigaatio sijaitsee verkkosivuston yläosassa ja sisältää linkit tärkeisiin sivuihin, kuten etusivu, palvelut, tietoja meistä ja yhteystiedot. Tarkkaan suunniteltu navigaatio tekee sivuston käytöstä miellyttävää ja auttaa kävijöitä löytämään tarvitsemansa tiedot nopeasti. Footer puolestaan löytyy sivuston alaosasta ja sisältää usein lisätietoja, kuten yhteystiedot, linkit sosiaalisen median profiileihin ja sivuston tekijänoikeustiedot. Footerissa voi myös olla navigaatio, joka tarjoaa linkkejä tärkeisiin sivuihin tai sivuston eri osioihin. Huolellisesti suunniteltu footer täydentää käyttäjäkokemusta ja tarjoaa lisätietoja, jotka voivat olla hyödyllisiä sivuston kävijöille.

UNDERSCORES-teeman valmiiksi luotu navigaatio tarjoaa erinomaisen perustan, ja sen muokkaaminen on vaivatonta WordPressin hallintapaneelin Menus-osiossa. Tavoitteenani oli integroida toimeksiantajan logo osaksi navigaatiota, ja aloitin tämän prosessin muokkaamalla teeman header.php-tiedostoa. Samalla varmistin, että sain tarvittavat rajapintayhteydet logoa ja tulevia linkkejä varten.

Footerin muokkaaminen ei ole suoraan mahdollista Menus-osiossa, joten tein tarvittavat muutokset footer.php- ja functions.php-tiedostoihin. Footer.php-tiedoston päivittäminen sallii alavalikon integroimisen sivuston alaosaan, ja functions.php-tiedoston muokkaaminen on välttämätöntä, jotta alavalikko voidaan liittää saumattomasti WordPressin hallintapaneelin Menus-osioon. Samalla varmistin, että sain tarvittavat rajapintayhteydet footeriin tulevista tiedoista.

Keskityin siihen, että käyttäjä voi keskitetysti hallita sekä pää- että alavalikkoa WordPressin hallintapaneelissa. Tämä tarkoittaa, että sivuston ylläpitäjä voi helposti tehdä tarvittavat muutokset ja päivitykset navigaatioon yhdessä paikassa. Tulevaisuudessa käyttäjä voi lisätä, muokata ja poistaa linkkejä pää- ja alavalikkoon näppärästi WordPressin hallintapaneelin kautta. Näin pyrin yksinkertaistamaan sivuston ylläpitoa ja tekemään siitä intuitiivisempaa käyttäjälle.

4 INTEGRAATIO PROSESSI

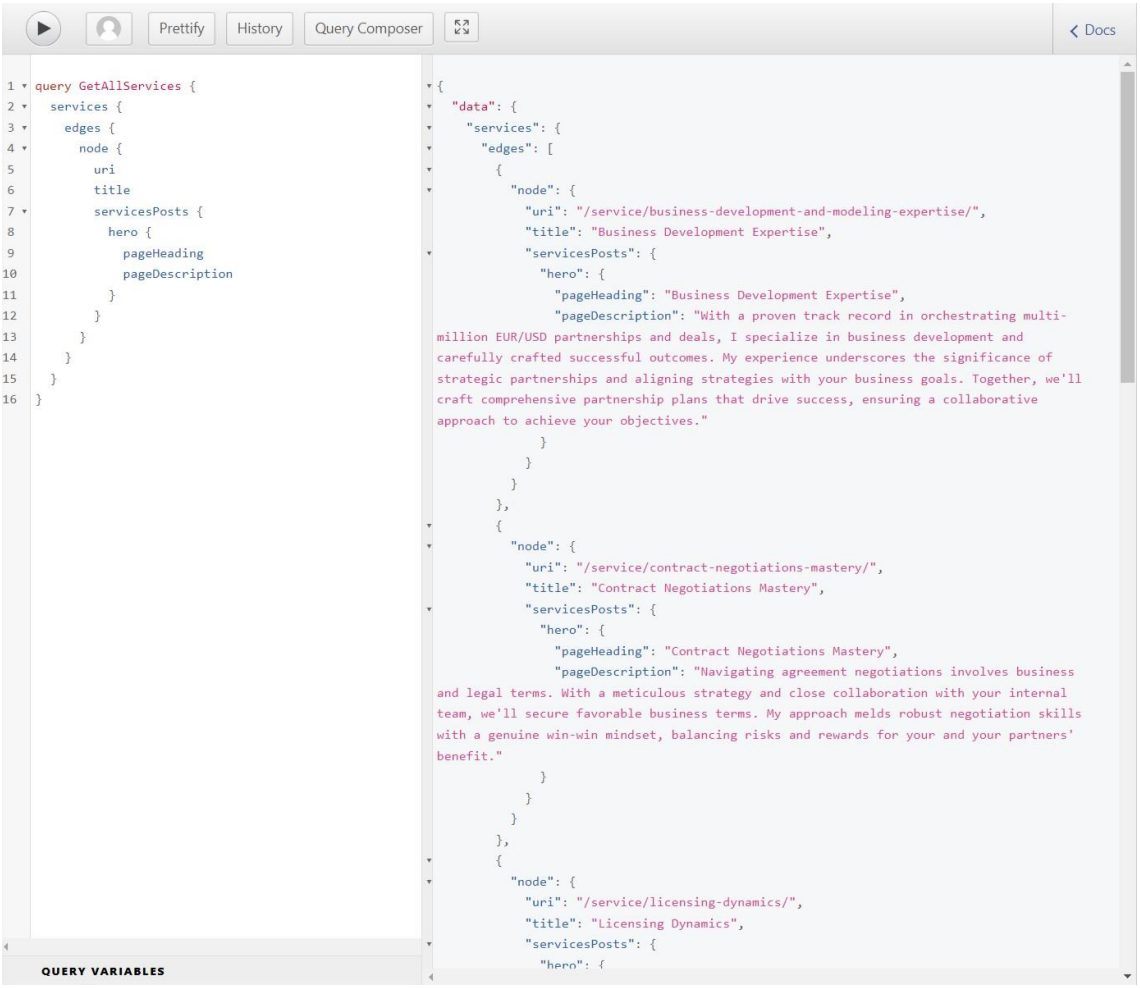
Projektin käynnistäminen Vercelissä tarjosi vauhdikkaan alun verkkosivustoni rakentamiseen. Päädyin valitsemaan Vercelin Next.js-aloitusteeman. Teema sopi loistavasti tarpeisiini, koska teemassa ei ole valmiina mitään ylimääräistä. (Vercel, Inc. 2024d.)

Vercelin käyttöliittymä teki projektin aloittamisesta vaivatonta. Projektin luominen Vercelissä yhdessä GitHub-integraation kanssa mahdollisti tehokkaan sivuston versionhallinnan. GitHubin integraation ansiosta sivuston kehittäminen ja päivitykset olivat jouhevasti hallittavissa ja pystyin tekemään muutoksia luottavaisesti tietäen, että versionhallinta on tehokkaasti käytössä. Näiden alkuaskelten jälkeen oli aika alkaa kehittämään sivuston komponentteja ja datan hakemista WordPressistä.

4.1 Datan hakeminen WordPressistä

Kun päätin kehittää WordPress-sivustoani, tulin siihen tulokseen, että tarvitsen tehokkaan tavan hallita sisältöä. Tässä vaiheessa päätin hyödyntää GraphQL:ää WPGraphQL-lisäosan avulla, joka tarjosi minulle vankan ja joustavan rajapinnan WordPress-sisältöön. Tämän ratkaisun avulla pystyin suunnittelemaan etukäteen, mitä tietoa tarvitsen eri sivuille, kuten etusivulle, about-sivulle, yhteydenottosivulle. Tiesin myös, että jokaisella toimeksiantajan palvelusivulla tarvittava tieto olisi identtistä.

Hyödyntämällä WPGraphQL:n tarjoamaa selkeää GraphQL-rajapintaa WordPressissä (kuvio 5) sain kattavan kuvan tulevista API-kutsuista jo ennen niiden toteuttamista Next.js-sovelluksessa. Tämä lähestymistapa antoi minulle mahdollisuuden suunnitella ja optimoida datan hakemisen ja käytön etukäteen, mikä puolestaan tehosti projektin kehitystä ja vähensi mahdollisia ongelmia tulevaisuudessa.



Kuvio 5. GraphQl-kysely Services-sivujen hakemisesta

Lisäksi lisäsin linkin Next.js:n ja Vercelin env-tiedostoon. Yleisesti ottaen .env-tiedosto (Environment Variables) on konfiguraatiotiedosto, joka sisältää ympäristömuuttujia, kuten API-avaimia tai muita salaisuuksia, jotka eivät saa näkyä versionhallinnassa. Tämä mahdollistaa sovelluksen käyttämien muuttujien hallinnan eri ympäristöissä, kuten kehityksessä, testauksessa ja tuotannossa, ilman että koodia tarvitsee muokata. Lisäämällä linkin Next.jsin .env-tiedostoon sain varmistettua, että sovellus pystyy kommunikoimaan oikean WordPress-rajapinnan kanssa ja käyttämään oikeaa dataa suunnitelmani mukaisesti.

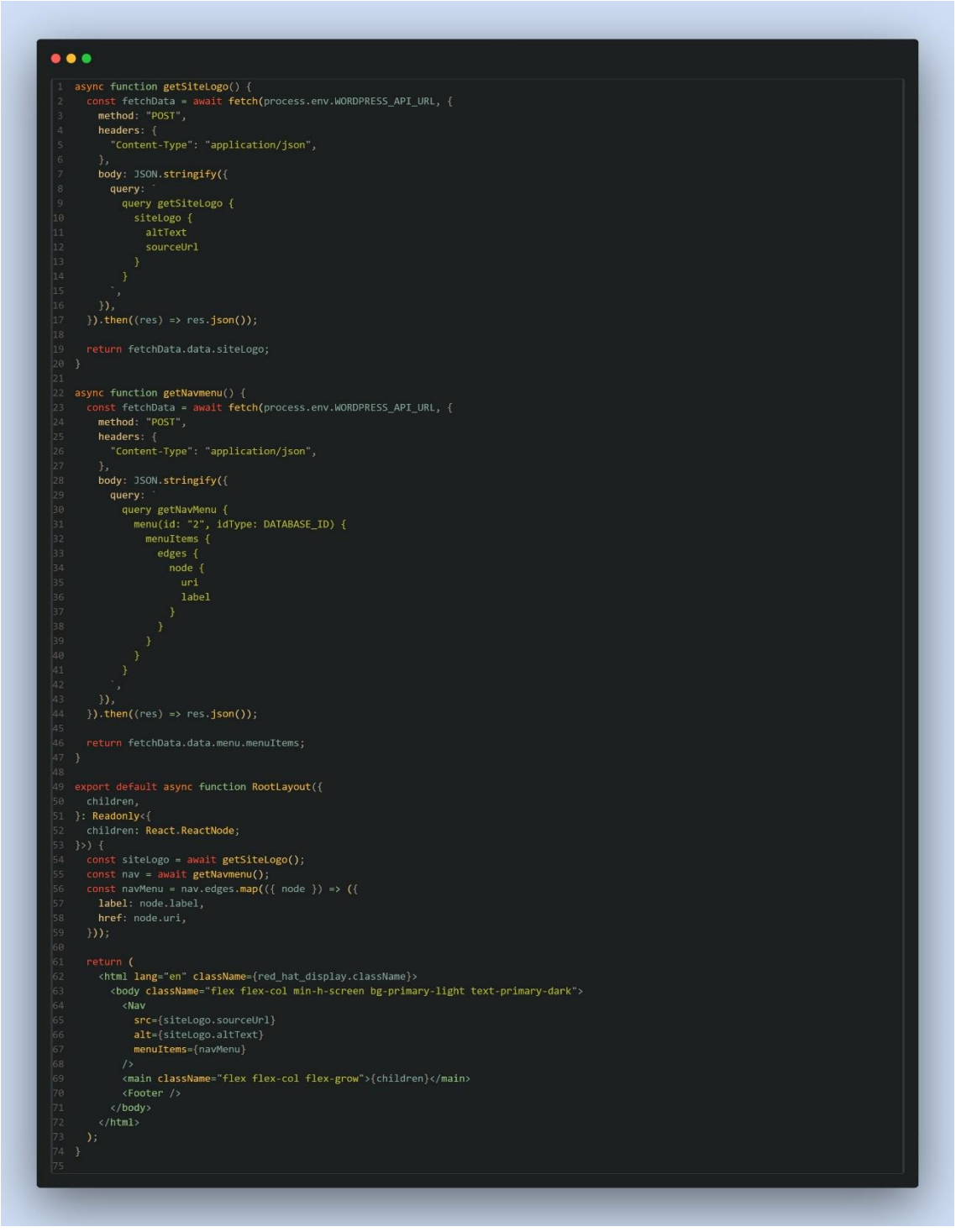
4.2 Ylävalikon ja alavalikon kehitys

Frontend-kehityksen päätin aloittaa luomalla responsiivisen navigaatio- ja alatunniste-elementin, jotka olisivat johdonmukaisia ja helposti muokattavissa olevia koko sivuston läpi. WordPressin puolella olin luonut ylävalikolla valikon, jossa

käyttäjä pystyy määrittämään sivuston logon ja haluamansa linkit. Sovimme toimeksiantajan kanssa, että navigaatiossa näkyisivät pääsivujen linkit about, service ja contact, jotka johtaisivat kyseiselle sivulle. Alavalikkoon päädyimme hakemaan yrityksen nimen ja yhteystiedot.

Loin erilliset komponentit sekä navigaatiolle että alatunnisteelle, jotka sijoittuivat omiin kansioihinsa projektini sisällä. Näiden komponenttien avulla varmistin, että molemmat osiot olivat helposti hallittavissa ja muokattavissa erikseen. Yhdistin nämä komponentit sitten yleiseen layout-tiedostoon (layout.tsx), jotta jokaisella sivulla olisi sama navigaatio ja alatunniste, mikä edistäisi sivuston yhtenäistä ulkoasua ja käyttökokemusta.

Ylänavigaation datan hakeminen tapahtui layout-tiedostossa. Ylänavigaation datan hakemiseen loin kaksi erillistä funktiota getSiteLogo ja getNavMenu (kuvio 6). Tämän jälkeen laitoin tiedot paikoilleen juuri luotuuni nav-komponenttiin. Näin varmistin, että navigaatio ja alatunniste päivittyvät automaattisesti. Tämä antaa joustavuutta ylläpidossa ja mahdollisti muutosten tekemisen ilman koodin muokkaamista.



```
async function getSiteLogo() {
  const fetchData = await fetch(process.env.WORDPRESS_API_URL, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      query: `
        query getSiteLogo {
          siteLogo {
            altText
            sourceUrl
          }
        }
      `,
    }),
  }).then((res) => res.json());

  return fetchData.data.siteLogo;
}

async function getNavmenu() {
  const fetchData = await fetch(process.env.WORDPRESS_API_URL, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      query: `
        query getNavMenu {
          menu(id: "2", idType: DATABASE_ID) {
            menuItems {
              edges {
                node {
                  uri
                  label
                }
              }
            }
          }
        }
      `,
    }),
  }).then((res) => res.json());

  return fetchData.data.menu.menuItems;
}

export default async function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  const siteLogo = await getSiteLogo();
  const nav = await getNavmenu();
  const navMenu = nav.edges.map(({ node }) => ({
    label: node.label,
    href: node.uri,
  }));

  return (
    <html lang="en" className={red_hat_display.className}>
      <body className="flex flex-col min-h-screen bg-primary-light text-primary-dark">
        <Nav
          src={siteLogo.sourceUrl}
          alt={siteLogo.altText}
          menuItems={navMenu}
        />
        <main className="flex flex-col flex-grow">{children}</main>
        <Footer />
      </body>
    </html>
  );
}

```

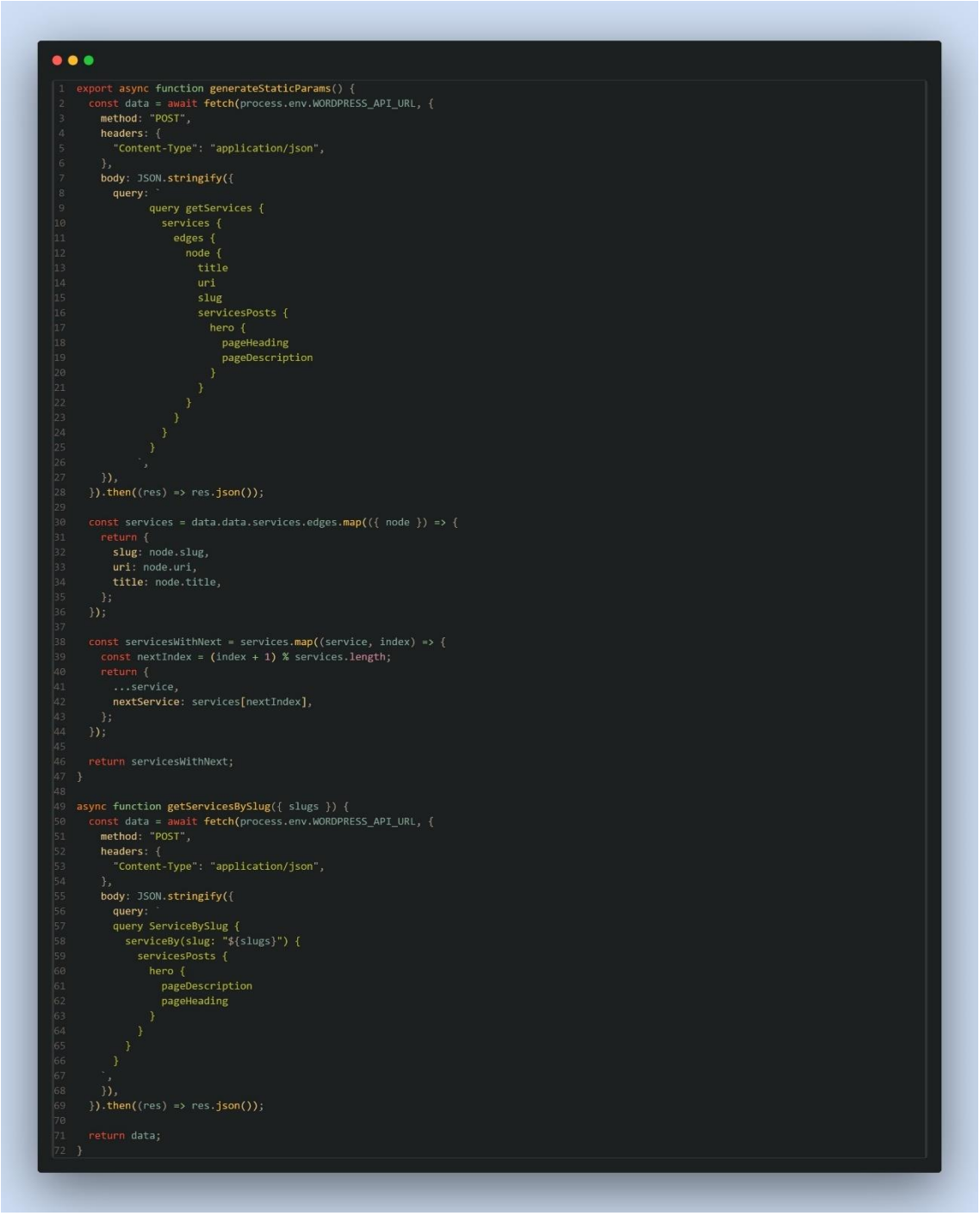
Kuvio 6. Koodi ylävalikon datan hakemisesta

4.3 Sivujen kehittäminen

Hyödynsin sivujen kehittämisessä Next.js:n sisäänrakennettua reititysjärjestelmää, jonka ansiosta pystyin joustavasti hallita ja navigoida eri sivujen välillä. Next.js:n app router-teknologialla loin about-, service- ja contact-sivuille omat

kansiot ja niiden sisälle page.tsx-tiedostot. Jokaiselle sivulle hain GrpahGL-kyselyllä tarvittavat sivukohtaiset datat näkyville. Koska olin itse kehittänyt ja luonut jokaiselle sivulle ACF-kentät tiesin mitä tietoja minun tulee hakea kullekin sivustolle näkyville.

Erityisesti palvelusivujen osalla päätin luoda jokaiselle palvelulle oman sivunsa lisäämällä services-kansioon uuden kansion nimeltä [slug]. Tämä mahdollisti dynaamisten polkujen luomisen, jossa jokaiselle palvelulle voidaan luoda oma ainutlaatuinen polku. Palvelusivujen datan hakemiseen Wordpressistä loin kaksi apufunktiota generateStaticParams ja getServicesBySlug, jotka käyttivät GraphQL-kyselyitä hakemaan WordPressistä palvelusivuihin liittyvän datan (kuvio 7). Ensimmäinen funktio hakee datan kaikista palveluista, kun taas jälkimmäinen hakee tietyn palvelun tiedot slugin perusteella.



```
export async function generateStaticParams() {
  const data = await fetch(process.env.WORDPRESS_API_URL, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      query: `
            query getServices {
              services {
                edges {
                  node {
                    title
                    uri
                    slug
                    servicesPosts {
                      hero {
                        pageHeading
                        pageDescription
                      }
                    }
                  }
                }
              }
            }
          `,
    }),
  }).then((res) => res.json());

  const services = data.data.services.edges.map(({ node }) => {
    return {
      slug: node.slug,
      uri: node.uri,
      title: node.title,
    };
  });

  const servicesWithNext = services.map((service, index) => {
    const nextIndex = (index + 1) % services.length;
    return {
      ...service,
      nextService: services[nextIndex],
    };
  });

  return servicesWithNext;
}

async function getServicesBySlug({ slugs }) {
  const data = await fetch(process.env.WORDPRESS_API_URL, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      query: `
      query ServiceBySlug {
        serviceBy(slug: "${slugs}") {
          servicesPosts {
            hero {
              pageDescription
              pageHeading
            }
          }
        }
      }
    `,
    }),
  }).then((res) => res.json());

  return data;
}
```

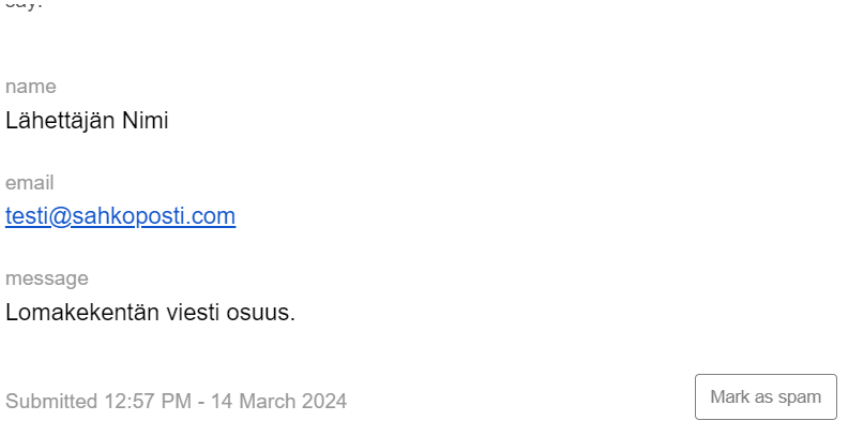
Kuvio 7. Koodi service-sivujen hakemisesta

4.4 Yhteydenottolomakkeen kehittäminen

Yhteydenottolomake tarjoaa helpon tavan käyttäjille ottaa toimeksiantajaan yhteyttä. Toimeksiantajan toiveesta yhdistin sivustolle yksinkertaisen lomakkeen, jossa kysytään käyttäjän nimi, sähköpostiosoite ja viesti. Päädyin luomaan lomakkeen käyttäen Formspree-onlinen palvelua, koska palvelu on ilmainen ja helposti yhdistettävissä verkkosivustolle. Loin tunnukset palveluun, tämän jälkeen

loin palvelussa lomakkeen ja lisäsin siihen oman sähköpostiosoitteeni aluksi. Halusin testata aluksi, että palvelun toimivuuden. Testauksen jälkeen vaihdoin toimeksiantajan sähköpostiosoitteen palvelussa.

Tämän jälkeen seurasin Formspree-palvelun ohjeita, jotta sain lomakkeen toimimaan oikein. Formspreen ohjeet olivat todella selkeät, jonka ansiosta palvelun yhdistäminen sujui vaivattomasti. (Formspree, Inc. 2024.) Koska laitoin oman sähköpostiosoitteen lomakkeen vastaanottajaksi pystyin testaamaan palvelua huoletta (kuvio 8). Testauksen jälkeen pystyin itse vaihtamaan palveluun toimeksiantajan haluaman sähköpostiosoitteen, jotta yhteydenotot tulevat lopulta oikeaan sähköpostiin.



Kuvio 8. Onnistunut lomakekentän toimiminen

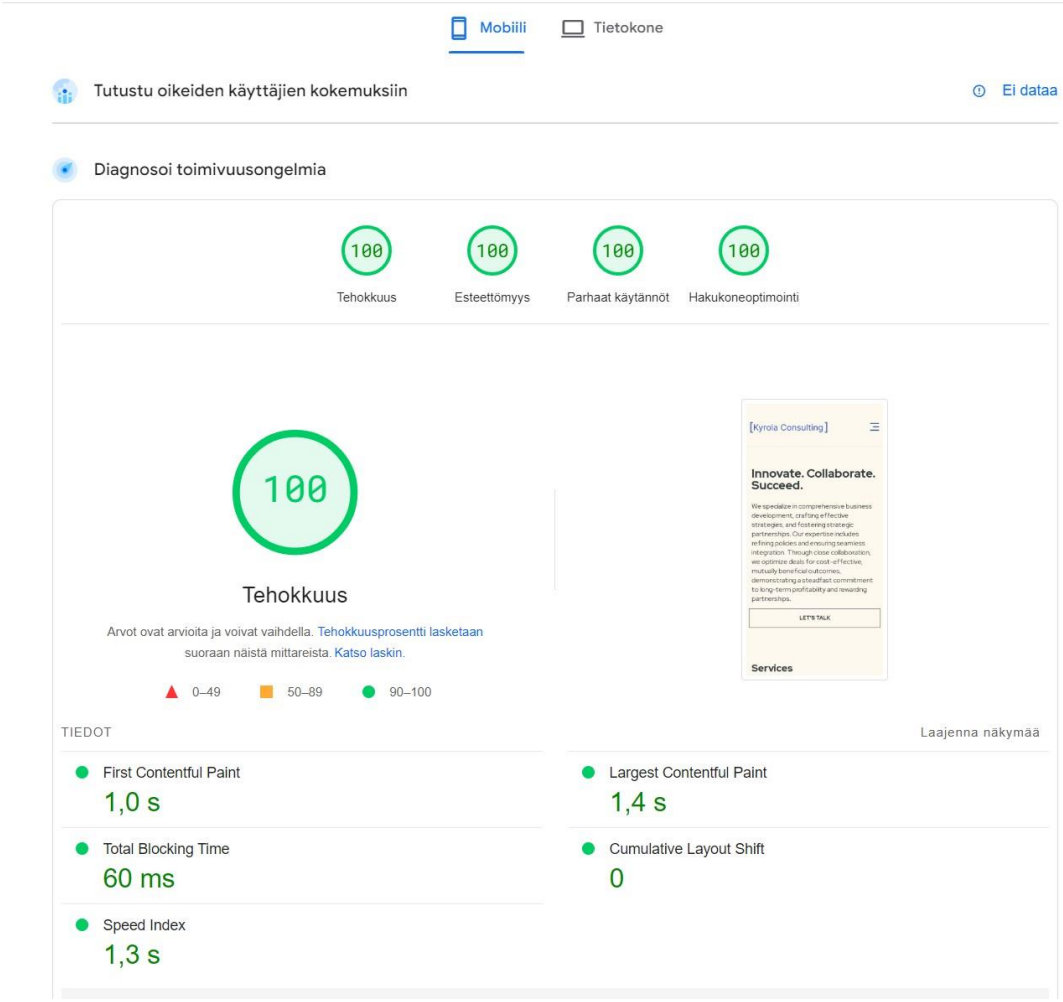
4.5 Testaus

Verkkosivujen kehittämisessä yksi keskeisimmistä tavoitteista on varmistaa, että sivusto toimii moitteettomasti eri selaimilla ja laitteilla. Erityisesti Headless-ratkaisua käytettäessä API-kutsujen on toimittava saumattomasti, koska sivusto voi luottaa raskaasti näihin taustapalveluihin tiedon noutamiseksi ja päivittämiseksi. Tässä kontekstissa ensimmäinen askel oli varmistaa sivuston responsiivisuus, joka tarkoittaa sitä, että sivusto skaalautuu ja sopeutuu optimaalisesti erikokoisiin näyttöihin ja laitteisiin. Tätä varten käytin selainkehityksen työkaluja, jotka mahdollistavat sivuston esikatselun eri näyttöko'oilla.

Kun varmistin responsiivisuuden, siirryin sivuston perusteelliseen testaukseen. Kävin läpi jokaisen sivun ja varmistin, että navigointi oli sujuvaa ja kaikki sisältö sekä elementit näkyivät oikein eri näyttöko'oissa, eri laitteilla ja eri selaimilla. Tämä sisälsi myös varmistuksen siitä, että sivuston latausaika oli optimaalinen ja että turhia API-kutsuja ei tehty. Googlen kehittämä PageSpeend Insights-työkalu oli korvaamaton työkalu tässä prosessissa.

PageSpeed Insights tarjoaa kattavan analyysin verkkosivun suorituskyvystä sekä ehdotuksia sen parantamiseksi. Se arvioi sivuston latausnopeutta sekä antaa suosituksia siitä, miten parantaa niin sivuston nopeutta kuin yleistä käyttökokemusta. Tämä sisältää ehdotuksia optimoida kuvia, pienentää JavaScript- ja CSS-tiedostoja. Sekä muita tekniikoita, jotka voivat nopeuttaa sivuston latausaikoja. PageSpeed Insightsin avulla pystyin tarkasti seuraamaan sivuston suorituskykyä ja tekemään tarvittavat parannukset sen optimoimiseksi (kuvio 9). Näin varmistin, että sivusto toimii tehokkaasti ja tarjoaa käyttäjille nopean ja miellyttävän selailukokemuksen kaikilla laitteilla ja selaimilla.

Sivuston nopea latausaika on ensiarvoisen tärkeää monista syistä. Ensinnäkin se parantaa käyttäjäkokemusta merkittävästi. Nopeat latausajat tekevät sivuston käyttämisestä sujuvaa ja miellyttävää, mikä puolestaan lisää kävijöiden viihtymistä ja sitoutumista sivustoon. Toiseksi nopea latausaika voi parantaa sivuston hakukonenäkyvyyttä. Hakukoneet, kuten Google arvostavat sivustoja, jotka tarjoavat hyvän käyttäjäkokemuksen, mukaan lukien nopeat latausajat. Nopeammat sivustot voivat siis saada parempia sijoituksia hakutuloksissa, mikä voi johtaa lisääntyneeseen liikenteeseen ja näkyvyyteen.



Kuvio 9. PageSpeed tulokset mobiililaitteella

4.6 Headless-integraation edut ja haasteet

Headless-integraatio tarjoaa useita etuja modernien verkkosovellusten kehityksessä, mutta samalla se kohtaa myös omat haasteensa. Yksi Headless-integraation merkittävimmistä eduista on joustavuus. Irrottautuminen perinteisestä monoliittisesta arkkitehtuurista mahdollistaa sovelluksen käyttöliittymän ja backendin kehittämisen erikseen, mikä antaa kehittäjille enemmän vapautta ja mahdollistaa nopeamman ketterän kehityksen. Lisäksi Headless-integraatio mahdollistaa sisällön toimittamisen useille eri kanaville ja laitteille, mikä parantaa käyttäjäkokemusta ja lisää markkinointitehokkuutta.

Toisaalta Headless-integraatioon liittyy myös haasteita. Yksi keskeisimmistä haasteista on integraation monimutkaisuus. Koska frontend ja backend toimivat erillisinä entiteetteinä, niiden välinen kommunikaatio ja synkronointi voivat olla

monimutkaisia. Tämä voi johtaa lisääntyneeseen kehitysaikaan ja vaatii huolellista suunnittelua ja ylläpitoa. Lisäksi Headless-integraatio saattaa vaatia uudenlaista osaamista ja työnkulkujen muutoksia kehitystiimissä, mikä voi aiheuttaa alkuvaiheen opettelukäyrän. Kaiken kaikkiaan Headless-integraatio tarjoaa merkittäviä etuja modernien verkkosovellusten kehityksessä, mutta sen käyttöönottoon liittyy myös haasteita, jotka on otettava huomioon suunnitteluvaiheessa ja toteutuksessa.

Headless-ratkaisujen käyttö WordPressin kanssa tarjosi minulle monia etuja, joista yksi keskeisimmistä oli mahdollisuus valita haluamani frontend-teknologia sivustolle. Perinteinen WordPress-sivusto on sidottu vahvasti selaimiin, mikä voi rajoittaa sen käyttöä erilaisten sovellusten, kuten mobiilisovellusten rakentamisessa. Headless-ratkaisussa taas WordPress toimii erillisenä backendinä, johon voin helposti kytkeä erillaisia frontend-ratkaisuja. Tämä antoi minulle joustavuutta ja mahdollisuuden tarjota sisältöä käyttäjille eri alustoilla ja laitteilla.

Valitessani Next.js:n Headless-ratkaisuuni huomioin erityisesti sen tarjoaman Server-Side Renderingin (SSR) ominaisuuden. SSR mahdollistaa sivuston sisällön generoinnin palvelimella jo ennen sen lähettämistä käyttäjälle, mikä parantaa sivuston latausnopeutta ja käyttökokemusta erityisesti mobiilikäytössä. Tämä on erinomainen esimerkki siitä, miten valitsemani teknologia voi tukea sivustoni löydettävyyttä ja käyttökokemusta esimerkiksi Google-haun näkökulmasta.

Näen siis Headless-ratkaisun valinnan ja Next.js:n käytön strategisena päätöksenä, joka mahdollistaa sivustoni kehittämisen monipuolisesti ja tarjoaa parhaan mahdollisen käyttäjäkokemuksen eri alustoilla ja tilanteissa. Tämä päätös auttoi minua saavuttamaan tarjoten nopean, responsiivisen ja monipuolisen sivuston.

5 POHDINTA

Kun lähdin toteuttamaan verkkosivustoa Headless-ratkaisulla käyttäen WordPressiä backendinä ja Next.js:ää frontendinä, kohtasin monia mielenkiintoisia haasteita. Yksi isoimmista haasteista oli saada service-sivut toimimaan Server-Side Rendering tavalla. SSR:n avulla halusin varmistaa, että sivuston sisältö generoituu palvelimella ja lähetetään käyttäjälle valmiina HTML-muodossa, mikä parantaa sivuston latausnopeutta ja hakukonenäkyvyyttä.

Toinen tärkeä näkökulma oli integraation sujuvuus WordPressin ja Next.js:n välillä. Vaikka Headless-ratkaisun ideana on erottaa selkeästi backend ja frontend, oli tärkeää varmistaa, että nämä kaksi osaa toimivat saumattomasti yhteen. Tämä vaati huolellista suunnittelua ja testausta varmistaakseni, että WordPressin tarjoama sisältö integroituu moitteettomasti Next.js:n kanssa ja että sivusto toimii odotetulla tavalla.

Lisäksi, kun harkitsin eri teknologioiden käyttöä projektissa, oli tärkeää ottaa huomioon myös ylläpidon ja skaalautuvuuden näkökulmat. Vaikka Next.js:ssä SSR-toiminnallisuus oli tärkeä valinta, myös sen helppo ylläpidettävyys ja laajennettavuus olivat merkittäviä tekijöitä päätöksenteossa. Halusin varmistaa, että valittu teknologia ei ainoastaan vastaa nykyisiin tarpeisiin, vaan myös mahdollistaa sivuston kehittämisen ja kasvun tulevaisuudessa. Päätin käyttää Verceliä frontendin julkaisualustana, koska olin aiemmin käyttänyt sitä ja se oli minulle tuttu. Tämä auttoi minua nopeuttamaan kehitystyötä ja varmisti, että voin keskittyä enemmän itse projektin toteuttamiseen kuin uuden julkaisualustan oppimiseen.

Vaikka WordPress oli minulle ennestään tuttu, minulla ei ollut aiempaa kokemusta koko teeman luomisesta. Tämä oli haaste, mutta samalla se tarjosi mahdollisuuden oppia uutta ja kehittää WordPress-taitojani entisestään.
Vaikka Headless-ratkaisun toteuttaminen osoittautui monimutkaisemmaksi ja aikaa vievämmäksi kuin, perinteisen toteutustavan WordPress-sivusto. Aion jatkossa suosia Headless-ratkaisua verkkosivujen kehittämisessä ja toteuttamisessa. Headless-ratkaisu tarjoaa mahdollisuuden käyttää erilaisia CMS-alustoja ja luoda erilaisia sovelluksia käyttäen samaa backendiä. Tämä lisää joustavuutta

ja skaalautuvuutta tulevaisuuden projekteissa sekä mahdollistaa monipuolisemman ja innovatiivisemman sisällön luomisen.

LÄHTEET

Formspree, Inc. 2024. Build a Contact From with React. Viitattu 13.3.2024 https://help.formspree.io/hc/en-us/articles/360053108134-Build-a-Contact-Form-with-React/.

Gavald, M 2023. WordPress Revolution with GraphQL. Kinsta Inc. 14.9.2023. Viitattu 20.4.2024 https://kinsta.com/blog/wordpress-revolution-with-graphql/.

Kinsta Inc. 2024a. What Is a Content Management System (CMS)? Viitattu 19.2.2024 https://kinsta.com/knowledgebase/content-management-system/.

Kinsta Inc. 2024b. What Is WordPress? Explained For Beginners. Viitattu 19.2.2024 https://kinsta.com/knowledgebase/what-is-wordpress/.

Knowit Oy 2024. Jamstack-arkkitehtuuri. Viitattu 20.2.2024 https://www.knowit.fi/palvelut/experience/web-ja-e-commerce/digitaaliset-palvelut-ja-alustat/jamstack/?gad_source=1&gclid=CjwKCAiAuNGuBhAkEiwAGId4ag8MO6EdcOxTBBFtVMtzCElQxnMKO-dGJDJ69WxhxXNEDh3KeKSyfxoC0OAQAvD_BwE/.

Local WP 2024. The #1 local WordPress development tool. Viitattu 21.2.2024 https://localwp.com/.

Maulidina, M 2024. What Is a WordPress Theme: Different Types and How to Choose One. Hostinger.com 28.2.2024. Viitattu 02.3.2024 https://www.hostinger.com/tutorials/what-is-a-wordpress-theme/.

Olawanle, J 2023. Understanding WPGraphQL and REST API for Headless WordPress. Kinsta Inc. 15.12.2023. Viitattu 04.3.2024 https://kinsta.com/blog/wpgraphql-vs-wp-rest-api/.

Underscores 2024. Create your underscores based theme. Viitattu 21.2.2024 https://underscores.me/.

Vercel, Inc. 2024a. The Native Next.js Platform 2024. Viitattu 20.2.2024 https://vercel.com/solutions/nextjs/.

Vercel, Inc. 2024b. Vercel is the Frontend Cloud. Build, scale, and secure a faster, personalized web. Viitattu 22.2.2024 https://vercel.com/home/.

Vercel, Inc. 2024c. Security 2024. Viitattu 24.2.2024 https://vercel.com/docs/security/overview/.

Vercel, Inc. 2024d. Next.js Broilerplate Viitattu 14.3.2024 https://vercel.com/templates/next.js/nextjs-boilerplate/.

Vercel, Inc. 2024e. Rendering. Viitattu 20.2.2024 https://nextjs.org/docs/app/building-your-application/rendering/.

Vercel, Inc. 2024f. Static Site Generation (SSG). Viitattu 20.2.2024 https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation/.

Vercel, Inc. 2024g. Static And Dynamic Rendering. Viitattu 20.2.2024 https://nextjs.org/learn/dashboard-app/static-and-dynamic-rendering/.

Vercel, Inc. 2024h. Optimizations. Viitattu 20.2.2024 https://nextjs.org/docs/app/building-your-application/optimizing/.

Vercel, Inc. 2024i. Image Optimization. Viitattu 20.2.2024 https://nextjs.org/docs/pages/building-your-application/optimizing/images/.

WebDevStudios 2024. Plugins. Viitattu 21.2.2024 https://wordpress.org/plugins/custom-post-type-ui/.

WordPress.org 2024. REST API Handbook. Viitattu 04.3.2024 https://developer.wordpress.org/rest-api/.

WPENGINE, INC. 2024. Advanced Custom Fields for WordPress Developers. Viitattu 21.2.2024 https://www.advancedcustomfields.com/.