Jaakko Kuivasniemi

Shop Easyn projektinhallinta ja kehitys React Nativella

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tieto- ja viestintätekniikka
Insinöörityö
6.5.2025

Tiivistelmä

Tekijä:	Jaakko Kuivasniemi
Otsikko:	Shop Easyn projektinhallinta ja kehitys React Nativella
Sivumäärä:	42 sivua + 1 liitettä
Aika:	6.5.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Amir Dirin, Lehtori Jukka Alhorinne, Tekstinohjaaja

Tässä opinnäytetyössä seurataan mobiilisovelluksen prototyypin Shop Easyn kehitystä. Shop Easy yhdistää kivijalkamyymälöiden tuotteet ja tarjoaa käyttäjäystävällisen alustan tuotteiden vertailulle sekä varaamiselle.

Sovelluksen kehitykseen käytettiin React Nativea ja ketteriä kehitysmenetelmiä, erityisesti Agilea ja Scrumia. Projektinhallintatyökaluina hyödynnettiin Jiraa, Discordia, Git:iä sekä WhatsAppia, jotka tukivat kommunikointia ja sprinttien seurantaa. Prototyyppi rakennettiin kuudessa noin kahden viikon sprintissä, joissa painopisteinä olivat muun muassa käyttäjien ja kirjautumisen toteutus, käyttäjäliittymän suunnittelu, karttanäkymä Google Maps API:n avulla, hakutoiminto sekä tietokannan integrointi sovellukseen. Projektin aikana esiintyneet haasteet, kuten riippuvuuksien hallinta, natiivikomponenttien integraatio ja näkemyserot sovelluksen toiminnasta ja ulkoasusta tarjosivat tärkeitä oppimiskokemuksia.

Lopputuloksena syntynyt prototyyppi vastaa työn tilaajan tarpeita sovellukselle ja osoittaa, että systemaattinen projektinhallinta yhdistettynä ketteriin menetelmiin on tärkeässä asemassa onnistuneessa ohjelmistokehitysprojektissa.

Avainsanat: React Native, Agile, Scrum, Ohjelmistokehitys, Projektinhallinta, Prototyyppi

Tämän insinöörityön alkuperä on tarkastettu Turnitin Originality Check -ohjelmalla.

Abstract

Author:	Jaakko Kuivasniemi
Title:	Shop Easy Project Management and Development with React Native
Number of Pages:	42 pages + 1 appendices
Date:	6.5.2025
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Software Development
Supervisors:	Amir Dirin, Principal Lecturer Jukka Alhorinne, Language Supervisor

---

This thesis follows the development of a mobile application prototype called Shop Easy. Shop Easy brings together products from local physical stores and provides a user-friendly platform for product comparison and reservations.

The development process utilized React Native and agile development methodologies, particularly Agile and Scrum. Project management tools such as Jira, Discord, Git, and WhatsApp supported communication and sprint tracking. The prototype was built over six sprints, each lasting approximately two weeks. Key development areas included implementing user accounts and authentication, designing the user interface, building a map view using the Google Maps API, developing a search function, and integrating a database into the application. Challenges encountered during the project—such as dependency management, integration of native components, and differing views on the app's functionality and appearance—provided valuable learning experiences.

The resulting prototype meets the client’s requirements and demonstrates that systematic project management combined with agile methodologies plays a crucial role in the success of a software development project.

Keywords:	React Native, Agile, Scrum, Software Development, Project Management, Prototype

Sisällys

Lyhenteet ja käsitteet

Git: Versionhallintajärjestelmä, jota käytetään lähdekoodin hallintaan.

Agile: Ketterä ohjelmistokehitysmenetelmä, joka korostaa iteratiivista kehitystä, asiakasyhteistyötä ja joustavuutta muutoksiin.

API (Application Programming Interface): Ohjelmointirajapinta, joka mahdollistaa eri ohjelmistojen tai komponenttien välisen vuorovaikutuksen.

Backlog: Projektin tehtävälista, joka sisältää kaikki suunnitellut ominaisuudet, parannukset ja korjaukset.

Dependency: Projektin riippuvuus ulkoisesta kirjastosta tai paketista, jota tarvitaan toiminnallisuuden toteuttamiseen.

Discord: Viestintäalusta, jota käytetään projektin sisäiseen kommunikaatioon ja tiedostojen jakamiseen.

Figma: Pilvipohjainen suunnittelutyökalu käyttöliittymien ja prototyyppien luomiseen.

Git: Versionhallintajärjestelmä, jota käytetään lähdekoodin muutosten hallintaan ja seuraamiseen.

GitHub: Verkkopohjainen alusta Git-versionhallinnan käyttöön, joka mahdollistaa yhteistyön ja koodin jakamisen.

GitHub Actions: CI/CD-automaatiotyökalu, joka mahdollistaa ohjelmistokehityksen vaiheiden, kuten testauksen ja julkaisun, automatisoinnin GitHubissa.

Jira: Projektinhallintasovellus, jota käytetään tehtävien hallintaan, sprinttien suunnitteluun ja ketterän kehitysprosessin tukemiseen.

Scrum: Agile-menetelmään perustuva projektinhallintatapa, joka jakaa työn sprintteihin.

Sprint: Aikarajattu kehitysjakso Scrum-projektissa, jonka aikana toteutetaan tietty osa projektin kokonaisuudesta.

Tiketti (Issue): Yksittäinen tehtävä, käyttäjätarina, virheilmoitus tai muu yksikkö projektinhallintajärjestelmässä.

VSCode (Visual Studio Code): Lähdekoodieditori, jota käytetään ohjelmistokehitykseen ja joka tukee laajasti laajennuksia ja ohjelmointikieliä.

WhatsApp: Viestisovellus, jota käytetään epäviralliseen ja nopeaan yhteydenpitoon projektiryhmän kesken.

1 Johdanto

Projektin voi määritellä joukoksi pienempiä tehtäviä, jotka täytyy suorittaa tietyn päämäärän saavuttamiseksi. Nämä tehtävät jaetaan viiteen eri projektin vaiheeseen, jotka ovat aloitus, suunnittelu, toteutus, seuranta ja lopetus (kuva 1). Projektipäällikköä taas voisi verrata öljyyn, joka voitelee koneiston, liimaan, joka pitää tuotteen kasassa, ja ohjaustankoon, joka ohjaa suuntaa. (Day 2024.)

Kuva 1. Projektin viisi eri vaihetta (Day 2024).

Projektinhallinta on tärkeää ohjelmistokehityksessä, koska se auttaa varmistamaan, että projekti valmistuu ajallaan, pysyy budjetissa ja täyttää laatuvaatimukset. Se myös ohjaa tiimiä kohti yhteistä tavoitetta.

Hyvä projektinhallinta auttaa myös ennakoimaan ja ratkaisemaan mahdolliset ongelmatilanteet, edistämään tehokasta viestintää ja ylläpitämään kehitysprosessin johdonmukaisuutta. Tällä varmistetaan, että projekti etenee hallitusti alusta loppuun ja tuottaa tarpeita vastaavan lopputuloksen.

2 Insinöörityön tavoite ja metodologia

Tässä insinöörityössä tutkitaan React Nativella rakennettavan kauppa-alustan prototyypin rakentamista pienessä tiimissä pääosin juuri projektinhallinnan näkökulmasta. Työssä ei niinkään käydä läpi koodausta tai itse koodia vaan kirjoittaja tutustuu työssä eri ohjelmatuotannossa käytettäviin projektinhallinnan menetelmiin ja työkaluihin sekä soveltaa niitä tässä insinöörityössä käsiteltävään projektiin.

Insinöörityön tavoitteena on tutkia projektinhallintaa ohjelmistokehityksessä ja vastata kysymyksiin kuten:

- Miten valitut kehitysmetodologiat sopivat pienelle tiimille?
- Miten valitut teknologiat sopivat sovelluksen kehitykseen React Nativella?
- Miten valitut sovellukset sopivat projektinhallintaan?
- Miten valitut teknologiat ja sovellukset vaikuttivat projektin lopputulokseen?
- Mitä haasteita projektinhallinnassa ja kehityksessä oli ja miten ne ratkaistiin?
- Mitä opittiin?

Projektissa on tavoitteena soveltaa ketteriä menetelmiä (Agile) Scrum-mallia hyödyntäen projektinhallinnan tukena. Näiden menetelmien avulla tiimi voi työskennellä tehokkaasti ja joustavasti, mukautuen muuttuviin tarpeisiin ja kehittyen jatkuvasti.

3 Ohjelmistotuotanto

3.1 Ohjelmistotuotannon tavoite

Ohjelmistotuotanto tarkoittaa ohjelmiston suunnittelua, kehitystä, testausta sekä ylläpitoa ja sen tavoitteena on luoda korkealaatuisia, käyttäjien tarpeet täyttäviä sekä kestäviä ohjelmistoja. Projektinhallinta on ohjelmistotuotannossa keskeisessä asemassa varsinkin kustannustehokkuuden sekä ajallisen hallinnan takia.

3.2 Ohjelmistotuotantometodeja

3.2.1 Agile

Agile eli ketterä kehitys on ohjelmistotuotannossa yleisesti käytetty projektinhallinnan filosofia, joka keskittyy nopeuteen, joustavuuteen ja asiakaslähtöisyyteen. Agile on kattotermi monille muille ohjelmistotuotannon käytännöille ja menetelmille. Agilen kehitys alkoi jo ennen vuotta 2001, mutta silloin termi vakiintui, kun 17 henkilöä kokoontui Utahissa Snowbirdin hiihtokeskuksessa keskustelemaan heidän käyttämistään erilaisista ohjelmistokehityksen käytänteistä ja näiden yhtenäisyyksistä. Vaikka kaikesta he eivät olleet samaa mieltä, keskustelujen lopputuloksena syntyi ”Manifesto for Agile Software Development” eli suomeksi ketterän ohjelmistokehityksen manifesti. (Agile Alliance 2024.)

Agile manifesto kuuluu seuraavasti:

> **Ketterän ohjelmistokehityksen julistus**
>
> Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:
>
> Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
>
> Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
>
> Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
>
> Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa
>
> Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.
>
> (Agile Manifesto 2001a.)

Ja sen 12. periaatetta kuuluvat seuraavasti:

> Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
>
> Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.
>
> Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.
>
> Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.
>
> Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.
>
> Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.
>
> Toimiva ohjelmisto on edistymisen ensisijainen mittari.
>
> Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.
>
> Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
>
> Yksinkertaisuus - tekemättä jätettävän työn maksimointi - on oleellista.
>
> Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.
>
> Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti. (Agile Manifesto 2001b.)

3.2.2 Scrum

Scrum on yleisesti käytetty ketterä prosessikehys, jonka avulla tiimit voivat työskennellä tehokkaasti ja tuottaa arvoa pienissä jaksoissa, joita kutsutaan sprinteiksi. Sprintit kestävät yleensä yhdestä neljään viikkoa ja niiden tarkoituksena on tuottaa aina jotain valmista ja käyttökelpoista. Palautetta annetaan koko sprintin ajan, ja tämä mahdollistaa työskentelyn ja lopputuloksen jatkuvan kehityksen. Sprinttien lopussa ja/tai alussa pidetään usein palaveri sidosryhmien kanssa, jossa arvioidaan, miten sprintti onnistui ja määritellään seuraavan sprintin tavoitteet. (Scrum.org 2024.)

Scrumissa on kolme keskeistä roolia:

- Product Owner eli tuotteen omistaja. Omistajalla on visio tuotteesta sekä määrittää tuotteen tavoitteet.
- Scrum Master eli scrum mestari varmistaa tuotannon onnistumisen, työtehtävien jaon ja toimii siltana tuotteen omistajan ja kehitystiimin välillä.
- Development Team eli kehitystiimi kehittää itse tuotteen.

3.2.3 Waterfall

Waterfall eli vesiputousmallin kehitti vuonna 1970 Winston W. Royce, ja se on aiemmin ollut todella suosittu lähestymistapa ohjelmistokehitykseen. Useimmat muut ohjelmistokehityksen mallit ovat johdannaisia vesiputousmallista. Vesiputousmalli on suoraviivainen ja sen vaiheet suoritetaan perättäin vaiheissa, jotka on suoritettava tietyssä järjestyksessä ja nämä vaiheet ovat

- Vaatimukset määritellään projektin alussa, jolloin asetetaan selkeät tavoitteet ja vaatimukset kehitystyölle.
- Suunnittelu sisältää sovelluksen arkkitehtuurin, ulkoasun ja toiminnallisuuden suunnittelun ennen kehitysvaihetta.
- Kehitys tarkoittaa itse sovelluksen rakentamista suunnitelman mukaisesti vaiheittain.
- Testaus varmistaa sovelluksen toimivuuden ja laadun ennen käyttöönottoa.

- Käyttöönotto tapahtuu, kun sovellus on valmis ja hyväksytty käyttöön.
- Ylläpito käyttöönoton jälkeen sovellusta päivitetään ja ylläpidetään.

Klassinen vesiputousmalli on simppeli ja idealistinen, ja se eroaa ketteristä menetelmistä siten, että se vaatii tarkan etukäteen suunnittelun toisin kuin ketterät menetelmät, jotka toistavat vesiputousmallin askelia sprinteissä täten mukautuen muutokseen. Vesiputousmalli sopii hyvin tarkasti määriteltyihin ja muuttumattomiin projekteihin. (Pal 2024.)

3.2.4 DevOps

DevOpsin ajatus on yhdistää ohjelmistokehitys (Development – Dev) ja IT-infrastruktuurin hallinta (Operations – Ops) jatkuvan yhteistyön ja automaation avulla. Perinteisesti nämä kaksi toimintoa ovat toimineet erillisinä osina organisaatioissa, mikä on johtanut ristiriitoihin, hidasteisiin ja tehottomuuteen. DevOps ratkaisee tämän korostamalla tiivistä yhteistyötä, prosessien automatisointia sekä nopeaa palautteenantoa. (Eficode 2013.)

DevOpsin lyhyt historia:

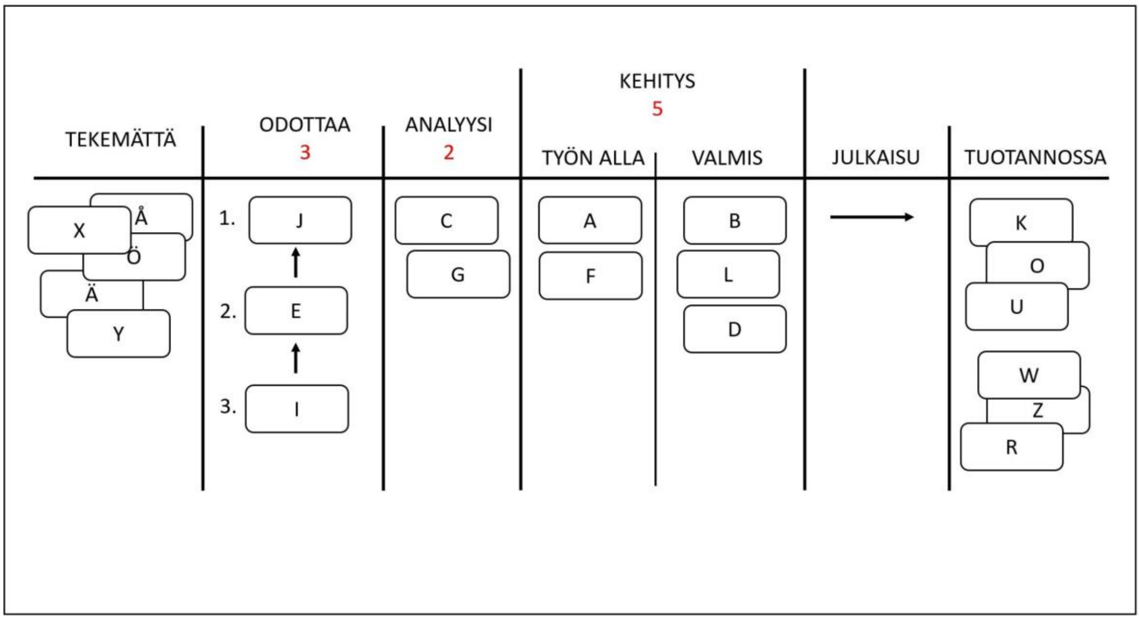
- 2007: Patrick Debois huomasi kehitys- ja operaatiotiimien huonon yhteistyön, mikä johti DevOps-ajattelun syntyyn.
- 2008: Andrew Shafer järjesti Agile-konferenssissa keskustelun ketterästä infrastruktuurista, jossa Debois tapasi hänet ja he keskustelivat DevOpsin tarpeesta. Tästä alkoi DevOps.
- 2009: Pidettiin ensimmäinen DevOpsDays-konferenssi, jossa termi "DevOps" sai alkunsa. DevOpsin käytännöt alkoivat saada suosiota.
- 2010: Continuous Delivery -kirja julkaistiin, esittäen automaation ja yhteistyön merkityksen ohjelmistojen jatkuvassa toimituksessa.
- 2013: Gene Kim, Kevin Behr ja George Spafford julkaisivat The Phoenix Project -kirjan, joka käsittelee lean-periaatteiden soveltamista ohjelmistokehityksessä.
- 2015: DORA (DevOps Research and Assessment) perustettiin ja se julkaisi State of DevOps -raportteja, jotka osoittivat DevOpsin positiivisen vaikutuksen organisaatioiden suorituskykyyn.
- 2016: Julkaistiin ”The DevOps Handbook” joka tarjosi käytännön ohjeet DevOpsiin.

- 2019: DevOps oli jo levinnyt laajasti ja yli 60 DevOpsDays-tapahtumaa järjestettiin 21 maassa, mikä osoitti DevOpsin merkittävän vaikutuksen ohjelmistokehityksessä. (Odazie & Iheanacho 2023.)

3.2.5 Kanban

Kanban on suosittu menetelmä Agile- ja DevOps-kehityksessä. Se perustuu reaaliaikaiseen viestintään ja työn läpinäkyvyyteen. Kanbanissa käytetään Kanban-taulua, jossa taululla näkyvät kaikki tehtävät ja niiden eteneminen, jolloin tiimi voi seurata työtilannetta helposti. (Radigan 2025.)

Kuvassa 2 näkyy esimerkki Kanban-taulusta, jossa kirjaimet kuvaavat työn osia ja sarakkeet ohjelmistokehityksen vaiheita.



Kuva 2. Kanban-taulu

3.2.6 Lean

Lean on myös ohjelmistokehityksessä käytetty ketterä metodologia, jonka tarkoitus on maksimoida arvo asiakkaalle ja poistaa kaikki ylimääräinen kehitysprosessista. Leanin ideaa voisi kuvata yhdellä lauseella ”Keskitytään työn vähentämiseen samalla kun kasvatetaan arvoa”. Työn vähentämisellä tarkoitetaan kaikkea turhaa työtä ja arvolla sitä kaikkea, mistä asiakas maksaa.

Ketteränä menetelmänä myös Lean pyörii sykleissä, joissa etsitään jatkuvasti parempia tapoja suorittaa tehtäviä. Näissä sykleissä on neljä vaihetta:

- Identification eli Tunnistus – Tässä vaiheessa tutkitaan mahdollisuuksia parantaa työnkulkua.
- Planning eli Suunnittelu – Mietitään miten tunnistettua ongelmaa voisi kehittää.
- Execution eli Suoritus – Toteutetaan itse muutokset, kun kehitysidea on valmis.
- Reviewing eli Seuranta – Tutkitaan miten tehdyt muutokset ovat vaikuttaneet prosessiin.

Ajatuksen mukaan näitä tekemällä ja jatkuvalla työn kehityksellä maksimoidaan työn tehokkuus eli toisin sanoen arvo. (Siddiqui 2021.)

3.2.7 Extreme Programming

Extreme Programming eli XP on Kent Beckin vuonna 1996 kehittämä Agile kehys ohjelmistokehitykseen. XP tähtää laadukkaaseen koodiin ja parantamaan kehitystiimin elämänlaatua.

Sen arvoja ovat:

- Communication eli viestintä: Extreme Programming painottaa nimenomaan kasvotusten keskustelua, koska ohjelmointi on pohjimmiltaan joukkueurheilua ja tiedon pitää liikkua koko joukkueelle.
- Simplicity eli yksinkertaisuus: XP:ssä vältetään ylimääräistä työtä ja kysytään että ”Mikä on yksinkertaisin ratkaisu?” ja toimitaan sen mukaan. Tämän takia projekti pysyy helpommin käytettävänä ja hoidettavana.
- Feedback eli palaute: Jatkuvan palautteen ansiosta tiimit pystyvät mukauttamaan työskentelytapojaan ja tavoitteitaan, niin kuin ketterissä menetelmissä on tapana.
- Courage eli rohkeus: Beckin mukaan tarvitaan rohkeutta puuttua tiimin tehokkuutta haittaaviin ongelmiin, koittaa uusia käytäntöjä sekä vastaanottaa palautetta, vaikka se ei olisi mieleistä.

- Respect eli arvostus: Tiimin keskinäinen arvostus on olennaista kommunikaation, palautteen toimivuuden ja yhdessä yksinkertaisten ratkaisuiden löytämisen takia. (Agile Alliance 2025.)

4 Shop Easyn projektinhallinta

4.1 Projektivisio

Shop Easylle perustetuilta nettisivuilta löytyy seuraavanlainen kuvaus perustettavasta yrityksestä ja sille kehitettävästä sovelluksesta:

> Kivijalkamyymälöiden, kauppa-, ja ostoskeskusten tulevaisuus on täällä.
>
> Me teemme vaateostoksista vaivatonta.
>
> Yhdistämme kivijalkamyymälöiden valikoimat, jotta kuluttaja-asiakkaat löytävät heille sopivimmat tuotteet nopeasti ja helposti yhdestä paikasta.
>
> Vaatekaupoille tarjoamme markkinointialustan, joka kohtaa kuluttaja-asiakkaan tarpeet.
>
> Nyt jokainen myymälä voi olla tasa-arvoinen ja nähty, sijaitsi se missä tahansa. (Tsokkinen 2025.)

Projektin visiona on kehittää hyperlokaali kauppa-alusta Shop Easy, joka yhdistää eri kauppojen tuotteet ja tarjoukset yhteen sovellukseen sekä mahdollistaa niiden vertailun. Näiden lisäksi sovelluksessa voi tehdä käyttäjän, hakea tuotteita liikkeistä ja varata niitä, selata mukana olevia kauppoja ja kauppakeskuksia kartalta ja hausta, ja paljon muita pienempiä ominaisuuksia. Prototyypin valmistuttua on tarkoituksena perustaa yritys, joka alkaa markkinoimaan tuotetta.

4.2 Projektitiimi

Projektissa on mukana neljä henkilöä: työn tilaaja (Product Owner), josta puhutaan työssä tilaajana, tämän insinöörityön kirjoittaja projektipäällikkönä (Scrum Master), johon viitataan kirjoittajana ja kaksi muuta ohjelmistotuotannon opiskelijaa Metropoliasta kehittäjinä (Development Team). Heitä kutsutaan tässä työssä nimillä kehittäjä 1 ja kehittäjä 2.

Tilaajalla on idea sovelluksesta, sen toiminnoista ja muotoilusta, mutta muuten sovittiin, että projektipäälliköllä on suhteellisen vapaat kädet projektin toteutuksen kanssa.

Projektipäällikön tehtävänä on kehityksen lisäksi projektin hallinta sekä projektissa käytettävien teknologioiden ja kehitysmetodien valinta ja käyttö, joita tarkastellaan tässä insinöörityössä tarkemmin.

Kehittäjä 1 keskittyy enemmikseen Backendin ja kehittäjä 2 enemmän taas Frontendin kehitykseen.

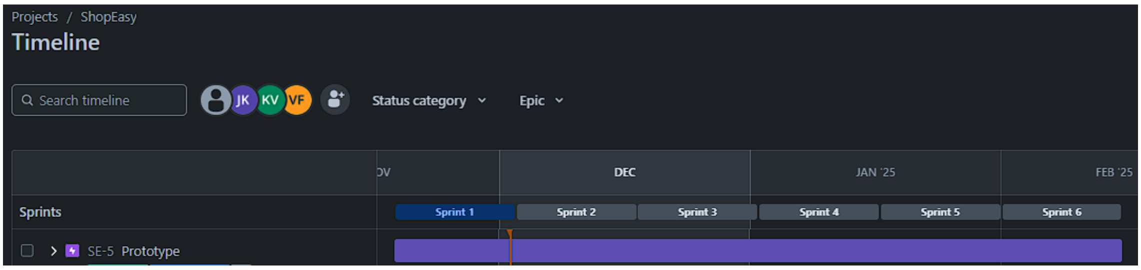
4.3 Projektin aikataulu ja budjetti

Kuten Nick Hodges (2015) kertoo artikkelissaan, ohjelmistokehityksessä oikea vastaus kysymykseen ”Kauanko menee saada tämä tehtyä?” on ”En tiedä.”, mutta kysymykseen pitää silti antaa joku arvio. Tämä kuvastaa ohjelmistokehityksen keston arviointiin liittyvää vaikeutta ja sitä, kuinka tärkeää on hallita odotuksia kehitysprojektin aikana.

Alkupalaverissa sovittiin prototyypin alkuperäiseksi aikatauluksi noin kolme kuukautta. Arvioitu aika perustui vain projektipäällikön ja toisen kehittäjän insinöörityöiden kirjoittamisen arvioituun kestoon ja siihen, että uskottiin, että tässä ajassa saadaan aikaan jo alustava versio sovelluksesta. Täysimittaisen sovelluksen kehitykseen tämä ei tietenkään ole riittävä aika, ja prototyyppiin tulevat ominaisuudet todennäköisesti muuttuvat kehityksen aikana. Jos annettu aika ei riitä kehitykseen, jatketaan projektia mahdollisesti siten, että kehitystiimi jatkaa projektissa harjoittelulla tai jää töihin, jos yritys perustetaan.

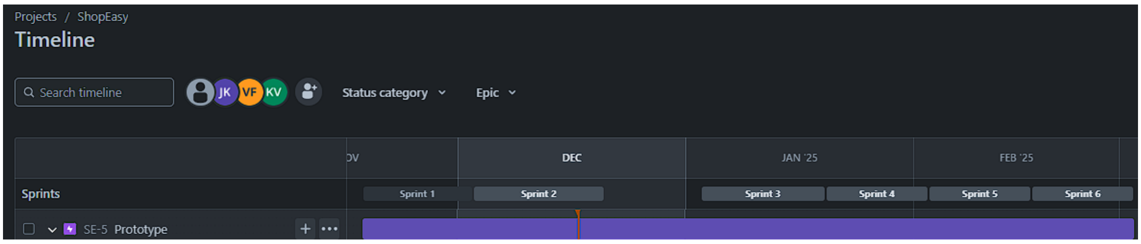
Budjettia sovelluksen kehitykseen ei ole sovittu, mutta tilaaja on tarjoutunut maksamaan mahdolliset kulut, kuten API- ja palvelinkustannukset sekä tilavuokrat tarvittaessa. Projektinjohtajalle maksetaan prototyypin kehityksen ajalta 500 euroa kuussa, mutta maksimissaan kolmen kuukauden ajalta.

Kuva 3 näyttää projektin alkuperäisen suunnitellun aikataulun ja sprintit Jiran aikajanalla.



Kuva 3. Alustava idea projektin aikataulusta Jirassa

Aikataulu kuitenkin muuttui nopeasti siten, että sprinttejä venytettiin hieman ja siihen lisättiin parin viikon mittainen joululoma (kuva 4).



Kuva 4. Projektin päivitetty aikataulu

4.4 Projektin riskianalyysi

Varsinaista riskianalyysia ei ole tehty, koska projektin aikataulua, budjettia tai oikeastaan edes tavoitteita ei ole tarkkaan määritelty. Projekti elää ja kehittyy muutosten mukana. Toki riskinä on aina, että projekti venyy aivan liian pitkäksi: joku lopettaa työt tai projekti osoittautuu liian haastavaksi pienelle opiskelijoista koostuvalle kehitystiimille (kts. liite 1 Projektisuunnitelma).

4.5 Projektimetodologia

Agile eli ketterä kehitys valittiin sen oletetun projektiin sopivuuden, joustavuuden ja yleisyyden takia projektin kehitysmetodiksi.

Scrum taas siksi, että projektipäällikkö on ennen projektia toiminut Scrum Masterina useasti ja Scrum metodologiana on ollut hyvin toimiva.

Alkupalaverissa keskusteltiin, että prototyyppi koitettaisiin kehittää kolmen kuukauden aikana. Kehitystiimin kanssa sovittiin, että pidetään kuusi noin kahden viikon mittaista sprinttiä. Sprintit on koitettu ajoittaa siten, että ne loppuvat sunnuntaihin ja alkavat maanantaisin ja väliin lisättiin kahden viikon joululoma, joten kaikki sprintit eivät ole aivan samanpituisia.

4.6 Projektissa käytetyt projektinhallintasovellukset

Erilaisia ohjelmistotuotantoon sopivia projektinhallinnan työkaluja on lukemattomia, mutta tässä työssä käydään läpi tarkemmin niitä, joita on hyödynnetty työn kohteena olevassa projektissa.

4.6.1 Jira

Jira on varsinkin ohjelmistokehityksessä käytetty projektinhallintasovellus, joka tukee hyvin Scrum ja Kanban-menetelmiä. Jirassa projekteille luodaan ’Backlog’, johon listataan projektin osat sekä issuet ja työssä käytetään näistä termiä tiketti. Näitä tikettejä sitten lisätään sprintteihin ja niille asetetaan vastuuhenkilöt. Jirassa tiketti on yksittäinen tehtävä, ongelma, ominaisuus tai jokin muu tehtävä työvaihe. Tikettejä sitten liikutellaan esimerkiksi ”To Do”-, ”In Progress”- ja ”Done”-tauluissa tiketin silloisen tilanteen mukaan.

Jirassa on eriarvoisia tikettejä:

- Epic on suurempi kokonaisuus, johon sisältyy useita pienempiä tikettejä.

- Story on käyttäjätarina, joka kuvaa ominaisuuden, sisältää pienempiä taskejä.
- Task on yleisesti pienempi tehtävä.
- Bug on virhe, joka vaatii korjausta.

Näillä saadaan yhdellä silmäyksellä todella tehokkaasti seurattua ja kokonaisvaltainen kuva projekteista. Pääprojektinhallintasovellukseksi Jira valikoitui saatujen suositusten ja sen Scrum kehitykseen sopivuuden takia.

4.6.2 Discord

Discord on viestintäalusta. Discord suunniteltiin aluksi pelaajille, mutta on levinnyt yleisempään käyttöön opiskelijoille, yhteisöille, tiimeille, ja yrityksillekin. Discordissa voi käydä teksti-, video- tai äänikeskusteluja ja se soveltuu myös pienempien tiedostojen jakoon. Discordiin luodaan palvelin yhteisölle ja palvelimilla on kanavia eri aiheille ja tavoille kommunikoida. Discord valikoitui projektin kommunikaatiokanavaksi sen yleisyyden, helppokäyttöisyyden ja tiedostojen jako mahdollisuuksien takia.

4.6.3 WhatsApp

WhatsApp on viestisovellus pääosin puhelimilla, mutta sillä on myös web-sovellus. WhatsAppia käytetään projektin osapuolten väliseen keskusteluun ja tapaamisista ym. sopimiseen. Projektin työkaluksi se valittiin sen yleisyyden ja helppouden takia.

4.6.4 Figma

Figma on pilvipohjainen käyttöliittymä- ja prototyyppisuunnittelutyökalu, jolla voidaan luoda visuaalisia ja lievästi toiminnallisiakin käyttöliittymiä eri alustoille.

Figmassa voidaan suunnitella ja luoda käyttöliittymiä samanaikaisesti usean ihmisen toimesta ja tämä tekee siitä helpon työkalun tiimityöhön.

4.6.5 Git, GitHub, GitHub Actions

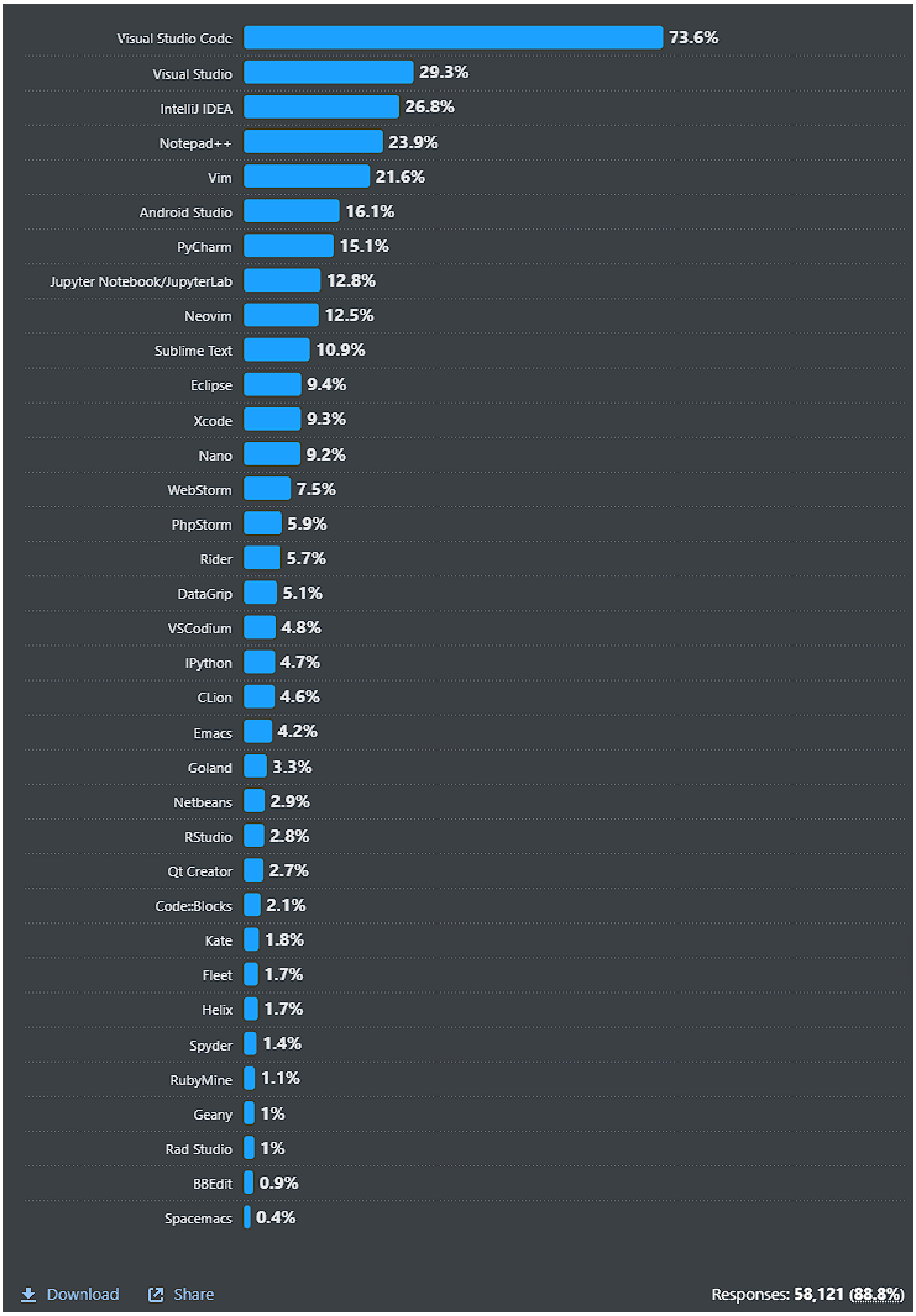
Git on ohjelmointiprojekteissa käytetty versionhallintajärjestelmä, jossa projektien lähdekoodin muutoksia ja versioita voi seurata ja hallita. GitHub taas on todella suosittu pilvipohjainen graafinen käyttöliittymä. (Chaudhary 2022.) Git:iä käyttävään GitHubiin tallennetaan lähdekoodi ja sieltä voi seurata koodin versiohistoriaa, luoda ja yhdistää projektin haaroja, keskustella, tehdä yhteistyötä ja käyttää erinäisiä työkaluja. GitHub valikoitui mukaan sen lähes pakollisuuden takia ohjelmistokehityksessä.

GitHub Actions on GitHubin tarjoama työkalu, jolla voidaan automatisoida monia ohjelmistokehityksen tehtäviä. Sitä voi käyttää CI/CD eli Continuous Integration (Jatkuva Integraatio) ja Continuous Delivery (Jatkuva Toimitus) -toiminnallisuuksiin, ja se toimii saumattomasti GitHub-repositorioiden kanssa. GitHub Actions mahdollistaa koodin testaamisen, rakentamisen, ja julkaisemisen automaattisesti määritettyjen sääntöjen mukaan. Projektissa se on mukana sen helppouden ja käytännöllisyyden takia GitHubin palveluna.

4.7 Projektissa käytetyt teknologiat

4.7.1 VSCode

Kehitysympäristöksi valittiin VSCode-projektissa yhteisesti käytettäväksi. IDE (engl. Integrated Development Enviroment), eli integroitu kehitysympäristö on sovellus, joka mahdollistaa erilaisten ohjelmoinnin osa-alueiden yhdistämisen yhteen paikkaan. Yleisiä toiminnallisuuksia ovat lähdekoodin muokkaaminen, suoritettavien tiedostojen rakentaminen, virheenkorjaus sekä muita koodausta helpottavia ominaisuuksia kuten automaattinen täydennys. (Codecademy Team 2024.) VSCode on Microsoftin kehittämä monia muita kehitysympäristöjä kevyempi, mutta monipuolinen, todella suosittu avoimen lähdekoodin kehitysympäristö, kuten kuvasta 5 näkyy.



Kuva 5. Stackoverflown kysely kehitysympäristöistä vuodelta 2024.

4.7.2 React

React on Metan (ent. Facebook) kehittämä avoimen lähdekoodin JavaScript-kirjasto. React on todella suosittu webkehityksessä sen nopeuden, tehokkuuden ja komponenttipohjaisen lähestymistavan takia.

4.7.3 React Native

React Native on Reactiin pohjautuva kehitysalusta, jolla voidaan kehittää natiiveja sovelluksia mobiililaitteille (Android, iOS) sekä verkkosovelluksia React Native Webin avulla. Yksi koodipohja ja kolme kohdeympäristöä tekevät siitä erittäin käytännöllisen verrattuna perinteisiin mobiilikehitysmenetelmiin, joissa sovellus täytyy kehittää erikseen jokaiselle alustalle.

React Native -sovellukset kirjoitetaan JavaScriptillä tai TypeScriptillä, ja ne käännetään natiivikoodiksi, Kotliniksi Androidille ja Swiftiksi iOS:lle sekä React Native Web mahdollistaa samojen komponenttien käytön verkkosovelluksissa, jolloin selainversio toteutuu ilman erillistä koodipohjaa. Myös React Nativea kehittää Meta yhdessä avoimen lähdekoodin yhteisön kanssa.

4.7.4 NativeWind

NativeWind on tyylittely työkalu. Se antaa mahdollisuuden käyttää TailWind CSS:n komponentteja React Native -sovelluksissa.

4.7.5 MongoDB

MongoDB on NoSQL (Not Only Structured Query Language) -tietokantajärjestelmä, joka mahdollistaa tietojen tallentamisen joustavasti ja skaalautuvasti. Siinä ei käytetä taulukoita kuten SQL-tietokannoissa, vaan MongoDB:n omaa BSON (Binary JSON) -muotoisia dokumentteja, joihin tallennetaan tiedot ja ne haetaan key-value-pareina.

4.7.6 GraphQL

GraphQL on vaihtoehto API-kyselyihin. Sen kehitti Facebook (Meta) vuonna 2012, ja se julkaistiin avoimena lähdekoodina 2015. Se on joustavampi ja tehokkaampi kuin perinteinen REST-arkkitehtuuri, koska sillä on mahdollista laatia tarkempia ja monimutkaisempia kyselyitä, joten turhaa tiedonhakua ei synny. GraphQL:ssä on myös vain yksi päätepiste, jonka kautta kaikki kyselyt (Query) tai datan lisäykset ja muokkaukset eli mutaatiot (Mutation) suoritetaan.

4.7.7 Jest

Jest on Metan avoimen lähdekoodin JavaScript testauskehys. Se on todella suosittu testaustyökalu varsinkin React- ja React Native -projekteissa. Se mahdollistaa testauksen yksinkertaisesti ja tehokkaasti. Jestillä tehtiin prototyypin yksikkötestit.

4.7.8 Expo

Expo on alusta, joka tarjoaa valmiita työkaluja, kirjastoja ja kehitysympäristön React Native -projektien kehitykseen. Se nopeuttaa ja helpottaa React Native -mobiilisovellusten kehitystä huomattavasti.

Expolla on Expo GO -sovellus, jolla sovelluksen voi avata saman verkon yli Metro Bundlerin kautta, jonka voi käynnistää suoraan IDE:ssä. Expo GO:ta käyttäessä kaikkia kirjastoja tai natiiveja komponentteja ei voi käyttää, mutta se on todella käytännöllinen pienempien projektien kehitykseen.

Kun Expo GO ei enää täytä tarpeita, voidaan siirtyä Development Buildiin (Custom Dev Client). Tämä toimii jokseenkin samalla tavalla kuin Expo GO, mutta nyt ladataan Expo.dev-sivulta oma .apk tai .aab Androidille tai .ipa iOS:lle. iOS-version lataus on huomattavasti monimutkaisempaa kuin Androidilla, joten tätä ei pystytty prototyypin kehityksen aikana tekemään. Se vaatii Apple-laitteen, Apple Developer -tilin sekä maksullisen Apple Developer Program -jäsenyyden.

Development Buildissa voi käyttää kaikkia natiiveja komponentteja ja kirjastoja, mutta miinuksena sovelluksen buildaaminen vie aikaa, erityisesti ilman Expon maksullista EAS Build -palvelua. Kumpikin Expo GO ja Expo Development Build tukevat Reactin Fast Refreshiä, eli sovellusta ei tarvitse käynnistää uudestaan, kun koodiin tehdään muutoksia, vaan nämä näkyvät heti mobiililaitteessa tai selaimessa.

Kun sovellus on valmis jaettavaksi, voidaan tehdä Standalone Preview ja Production Buildit. Preview Build on tarkoitettu sisäiseen jakoon, esimerkiksi TestFlightiin tai Google Playn sisäiseen testaukseen. Production Build on sovelluskauppoihin ladattava versio. Preview build on tarkoitettu sovelluksen kehityksen sisäiseen jakoon ja Production Build on sovelluskauppoihin ladattava versio sovelluksesta.

4.7.9 ESLint

ESLint on työkalu, joka tarkistaa ja parantaa koodin laatua sekä varoittaa virheistä määritettyjen asetusten ja hyvien käytänteiden mukaan esimerkiksi poistamalla käyttämätöntä koodia tai varoittamalla epäjohdonmukaisesta koodista.

4.7.10 Prettier

Prettier on automaattinen koodin muotoilutyökalu, jolla koodi pysyy määriteltyjen sääntöjen mukaan johdonmukaisena ja samalla tavalla formatoituna kaikilla ja kaikkialla.

Prettierissä ja ESLintissä on risteäviä ominaisuuksia, mutta ne pääasiassa hoitavat eri asiaa. Prettier keskittyy enemmän koodin ulkonäköön ja ESLint taas koodin sisältöön ja johdonmukaisuuteen.

4.8 Projektin seuranta ja seurantamenetelmät

Tapaamiset hoidettiin kehitystiimin kesken lähitapaamisina tai Discordissa ja tilaajan kanssa Zoomissa. Scrumia toteutettiin seuraavasti:

Sprint Planning eli sprintin suunnittelu tapaaminen:

Spirintin alussa projektipäälliköllä oli palaveri ensin tilaajan kanssa, jossa keskusteltiin viime sprintin tavoitteiden saavuttamisesta sekä käytiin sovelluksen toimintaa sekä muita projektiin liittyviä asioita läpi. Lisäksi sovittiin yhdessä, mitä tulevassa sprintissä on tavoitteena saada tehtyä sekä mahdolliset muut asiat. Tämän jälkeen oli kehitystiimin kanssa tapaaminen, jossa keskusteltiin tarkemmin siitä, miten ja millä aikataululla halutut asiat saadaan hoidettua.

Daily Scrum eli päivittäinen Scrum:

Päivittäin pidettävät tapaamiset olivat klo 10 Discord kanavalla. Näissä tapaamisissa oli tarkoitus käydä pikaisesti läpi, mitä kukakin on tehnyt, mahdolliset ongelmat ja kysymykset, sekä sitä, mitä kukakin tekee seuraavaksi. Keskiviikkoisin oli lähipäivä, jolloin kehitystiimi tapasi jollain Metropolian toimipisteellä ja työskenteli ja keskusteli projektiin liittyvistä asioista yhdessä.

Weekly Scrum eli viikottainen Scrum:

Sprintin ensimmäisen viikon perjantaina pidettiin viikon lopetustapaaminen. Tapaamisessa käytiin läpi tarkemmin, miten viikon ja sprintin tavoitteita on saavutettu ja miten ensi viikolla jatketaan.

Sprint Review Meeting eli sprintin arviointitapaaminen:

Sprintin viimeisenä perjantaina käytiin läpi, miten sprintin tavoitteet saavutettiin, sekä ongelmat, kysymykset, ideat ja muut asiat, minkä lisäksi suunniteltiin jo seuraavaa sprinttiä.

Sprint Retrospective Meeting eli sprintin jälkitarkastelutapaamisia ei pidetty. Projektin edistystä seurattiin pidetyissä tapaamisissa, niiden dokumentaatiossa, Discordissa sekä Jirassa.

5 Sprinttisuunnitelma

5.1 Alkupalaveri 18.11.

Alkupalaverissa tavattiin Kampissa. Tapaamisessa oli projektin tilaaja ja kaksi muuta Metropolian opiskelijaa, jotka suorittavat harjoittelua sekä insinöörityötä. Alkupalaverissa tiimi tutustui toisiinsa, ja tilaaja kävi läpi projektin visiota prototyypin osalta, ja mitä kaikkea sen pitäisi pitää sisällään sekä omia markkinointi- ja kehitysideoitaan prototyypin valmistuttua. Kirjoittaja kyseli kehittäjien mielipiteitä kehitysmetodeihin ja käytettäviin sovelluksiin sekä alusti projektinhallintasovelluksia.

Shop Easyn prototyypin kehityksen kestoksi arvioitiin noin kolme kuukautta eli kuusi kahden viikon sprinttiä. Tämä vaihteli haluttujen toimintojen ja niiden toteutuksen onnistumisen mukaan.

Tapaamiset sovittiin tilaajan kanssa jokaisen sprintin alkuun ja/tai loppuun ja tapaamisissa oli tarkoitus käydä läpi sitä, mitä ja miten sprintin tavoitteet on saavutettu sekä keskustella mahdollisista ongelmista ja suunnitella jatkoa.

Kehittäjien kanssa sovittiin, että sprinttien alussa ja vähintään kerran viikossa tavataan porukalla. Sprintin alussa suunniteltiin seuraava kahden viikon jakso ja jokaiselle työtehtävät. Töitä seurattiin päivittäin tai vuoropäivittäin henkilökohtaisesti ja tarvittaessa yhdessä lyhyehköissä palavereissa. Jokainen myös päivitti Jiraa edistymisensä mukaan, jota projektinjohtaja valvoi.

5.2 Sprint 1 17.11.–2.12.

Ensimmäisessä sprintissä oli tarkoituksena suunnitella prototyypin design, luoda projektin pohja, valita käytettävät teknologiat sekä tehdä sovellukselle runko, joka mahdollistaa navigoinnin eri sivujen välillä.

5.3 Sprint 2 2.12.–20.12.

Toisen sprintin tavoitteena oli saada tehtyä karttaan sekä etusivulle toiminnallisuutta dummyshoppien ja -tuotteiden kanssa. Tavoitteena oli myös saada asetukset ja profiilisivu tehtyä ja niille toimintoja sekä uusi kotisivu tilaajan vision mukaan sekä tyylitellä jo tehtyä sovellusta. Sprintin jälkeen sovittiin joululoma ajalle 21.12.–3.1.

5.4 Sprint 3 3.1.–19.1.

Kolmannen Sprintin tavoitteena oli kehittää uusi versio etusivusta, joka on kompromissi kahdesta aikaisemmasta versiosta sekä tyylitellä koko sovellus samalla tavalla. Kun tämä oli tehty tilaaja lähettää projektin palvelumuotoilijalle, jolta tulee kehitysideoita projektiin. Lisäksi oli tarkoitus saada tietokantaan kauppoja ja tuotteita sekä näille omat sivut.

5.5 Sprint 4 20.1–2.2.

Neljännen sprintin tavoitteena oli saada tuotteiden kategoria- ja tekstihaku tehtyä, karttaan lisättyä kauppakeskukset ja niiden sivut, lisättyä kauppoihin reititys käyttäjän sijainnista, tuote- ja kauppasivuja kehitettyä lisää sekä luotua uudet sivut eri profiili- ja asetussivujen painikkeille. Lisäksi nämä kaikki oli tarkoitus tyylitellä ja optimoida. Lisäksi oli tarkoitus opiskella Google Places API:a, jota käytettäisiin kauppakeskusten, kauppojen ja reittien tietojen hakemiseen.

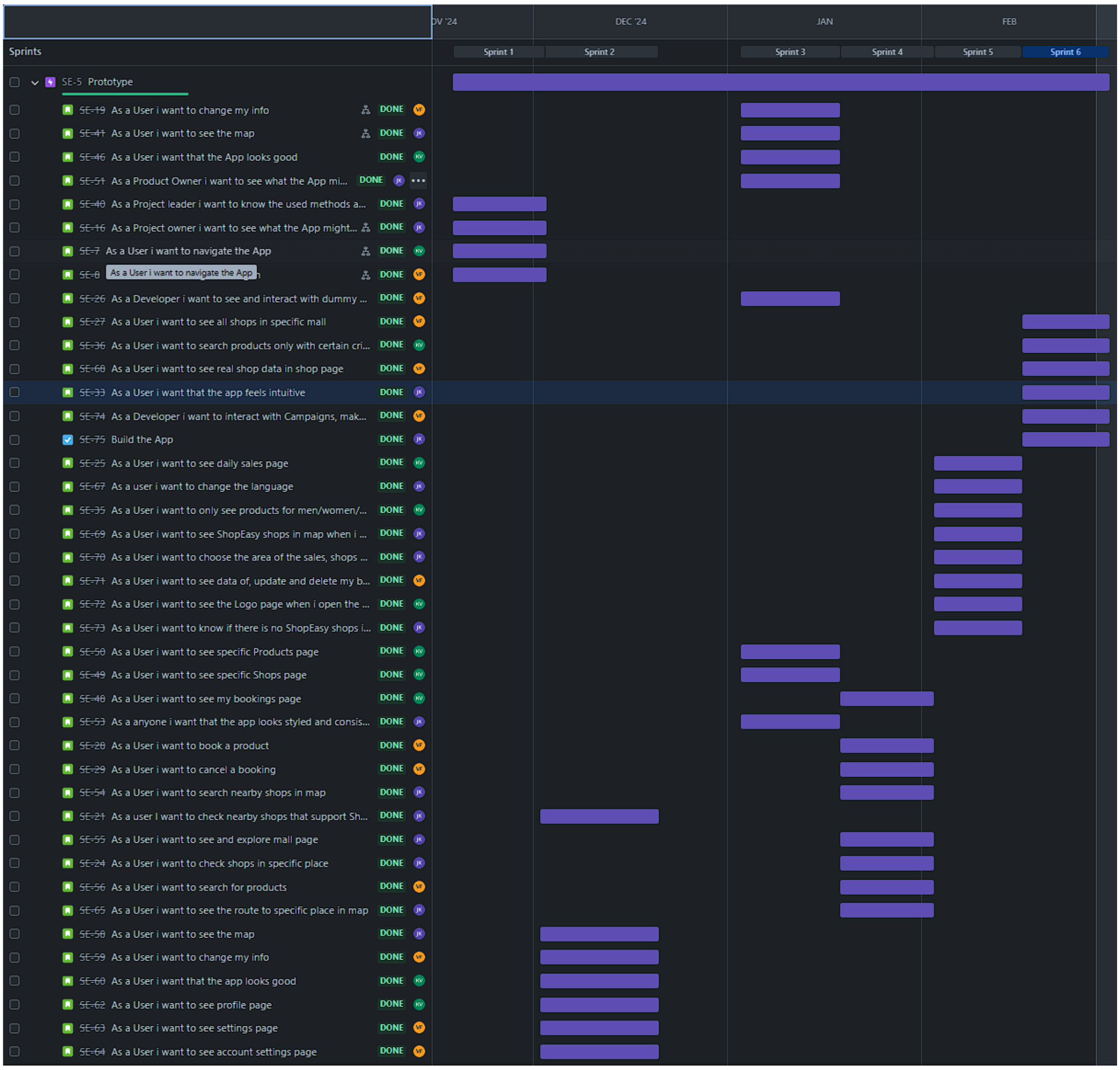
5.6 Sprint 5 3.2–16.2.

Viidennen sprintin tavoitteena oli saada varaus ja hakusivut viimeisteltyä, kielistä ainakin englanti ja ruotsi lisättyä sovellukseen sekä viimeistellä keskeneräisiä kokonaisuuksia ja yhtenäistää sovelluksen ulkonäköä. Lisäksi oli tarkoitus tehdä erinäisiä tilaajan pyytämiä lisäyksiä.

5.7 Sprint 6 17.2–2.3.

Kuudennen eli prototyypin kehityksen viimeisen sprintin tavoitteena oli saada sovellus näyttämään hyvältä mahdollisia rahoittajia ja yrityksiä varten. Sprintissä oli tarkoitus keskittyä isoihin kokonaisuuksiin, kuten karttaan ja hakuun siten, että ne toimivat halutulla tavalla. Lisäksi tavoitteena oli muuten tyylitellä sovellus yhtenäiseksi ja hyvän näköiseksi sekä korjata bugeja.

Kuvassa 6 näkyy prototyypin kehityksen aikajana ja tiketit Jirassa kuudennen sprintin jälkeen. Kuvassa näkyy, että kaikki prototyyppiin halutut ominaisuudet saatiin tehtyä, ja suurin osa vielä sille määrätyssä ajassa.



Kuva 6. Prototyypin aikataulu ja tiketit Jirassa.

Tiketeissä oli osittain päällekkäisyyksiä ja osaa tehtiin yhdessä, mutta suurin piirtein prototyypin työmäärä ja tiketit jakautuivat tasan kaikkien kesken.

5.8 Loppupalaveri 4.3.

Projektin loppupalaveri pidettiin Kampissa tilaajan ja IT-tiimin kesken. Yhdessä keskusteltiin siitä, että kirjoittaja ja kehittäjä 1 jatkavat projektissa, kunhan rahoitusasiat ovat kunnossa.

6 Projektin tulokset

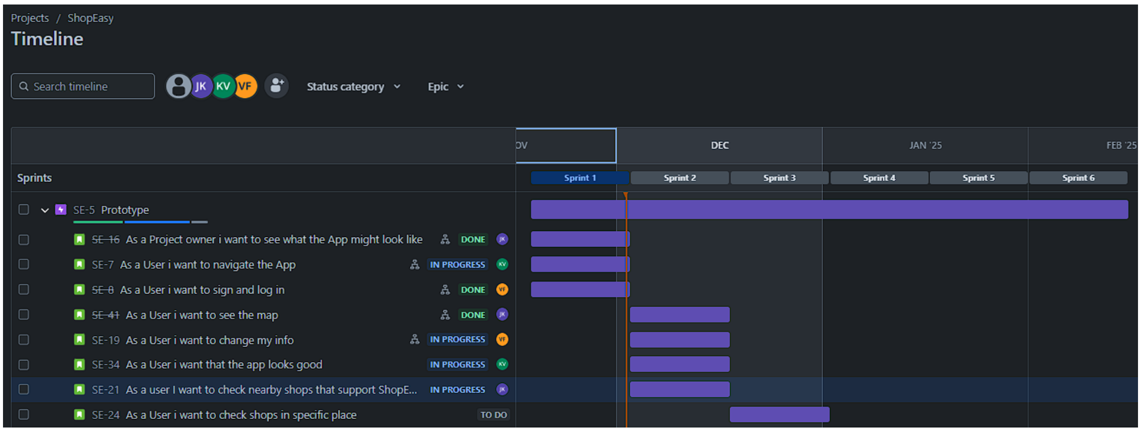
Projekti onnistui todella hyvin, kun miettii sille alussa määriteltyjä tavoitteita. Prototyyppi saatiin valmiiksi eikä sen suurempia haasteita juuri ollut. Tiimityö ja tiedonvaihto sujui erinomaisesti koko prototyypin kehityksen ajan. Projektin tuloksena saatiin rahoituksenkeruu tarpeisiin todella hyvin sopiva prototyyppi Shop Easyn mobiilisovelluksesta.

Sprinttien tuloksissa on merkitty tiketeiksi isompia kokonaisuuksia, kuten vaikka kartan toiminnallisuus tai kirjautuminen. Esimerkiksi kartta on voitu saada Sprintissä X valmiiksi, mutta sitä on voitu muokata muuttuneiden ideoiden takia sprintissä Y, se on kuitenkin merkattu tehdyksi sprinttiin X. Tiketit jaettiin aina vastuuhenkilölle, mutta tikettejä on tehty myös yhdessä.

6.1 Sprint 1 tulokset

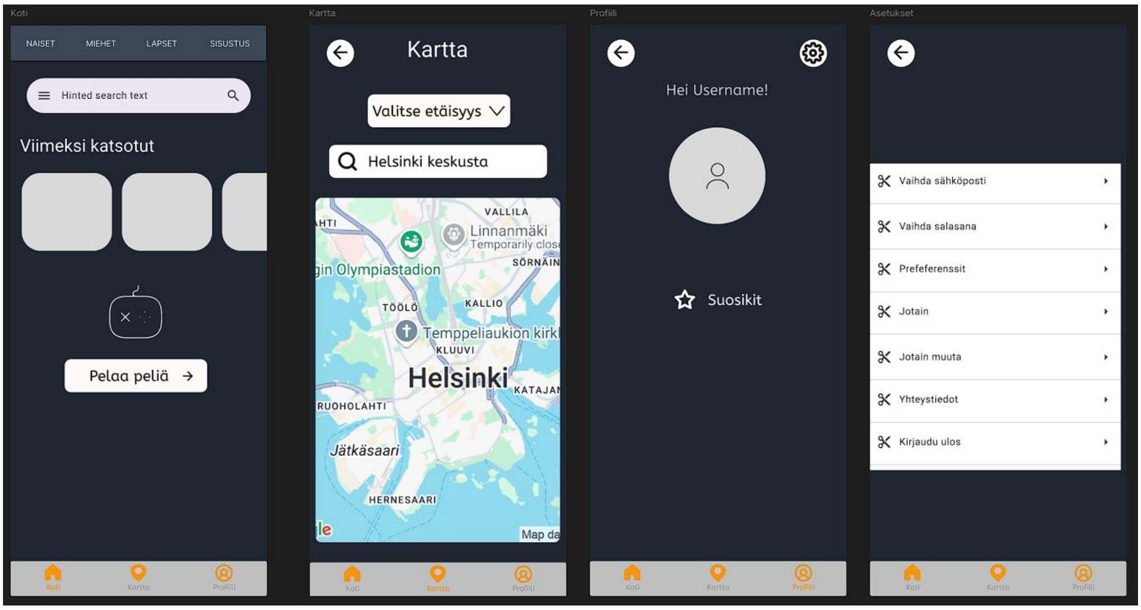
Sprintille asetetuttuja tikettejä suoritettu 4/4.

Ensimmäisessä sprintissä kirjoittaja tutki ja valitsi käytettävät teknologiat sekä lisäsi projektiin ESLintin, Prettierin, NativeWindin, GitHub Actions -workflow'n, Jestin ja alusti Jiran sekä Discord-palvelimen. Kirjoittaja myös toteutti kartta-sivun ja kartan sekä piti palaverit. Kuva 7 esittää ensimmäisen sprintin työnjakoa.



Kuva 7. Jira ensimmäisen Sprintin viimeisenä päivänä.

Projektin suunnittelua varten luotiin Figma projekti ja GitHub repositoryn tiimille. Kuva 8 näyttää alkuperäisen Figmalla luodun idean sovelluksen sivujen rakenteesta.



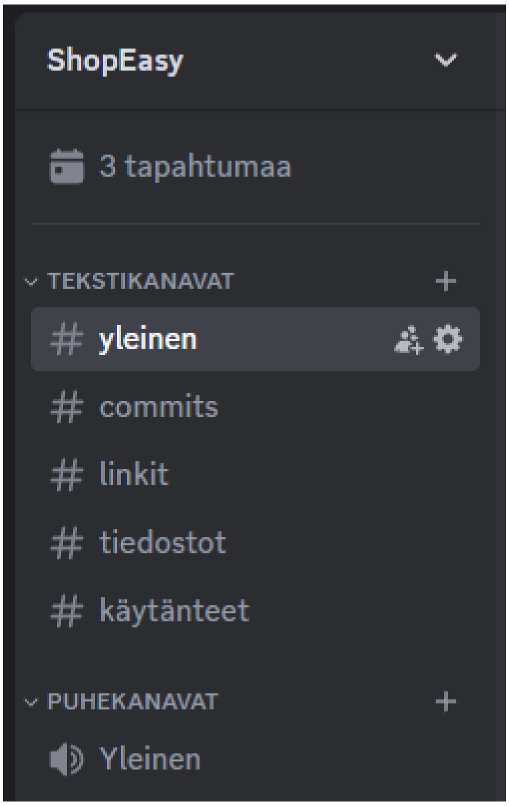
Kuva 8. Figmalla luotu ensimmäinen tyylittelemätön idea prototyypistä.

Figmassa luotiin prototyypin henkilökäyttäjien puoleisen pohjan alustava design, jonka pohjalta jaettiin työtehtävät siten, että kehittäjä 2 aloitti frontendin kehityksen, pohjasovelluksen eri sivuja sekä niiden navigaatiota ja kehittäjä 1 backendiä, eli alkoi suunnittelemaan kirjautumisen toiminnallisuutta ja tietokantaa. Projektipäällikkö keskittyi käytettävien teknologioiden tutkimiseen ja prototyypin Backlogin kasaamiseen, eli prototyypin haluttujen toimintojen listaamiseen sekä sprinttien ja työnjaon suunnitteluun Jiraan.

Projektin alussa määriteltiin käytettäväksi yhteisesti käytettävät formatteri ja lintteri projektiin, Prettier ja ESLint vastaavasti. Projektiin integroitiin myös GitHub Actions Workflow automaattista projektin testausta varten sekä testauskirjasto Jest.

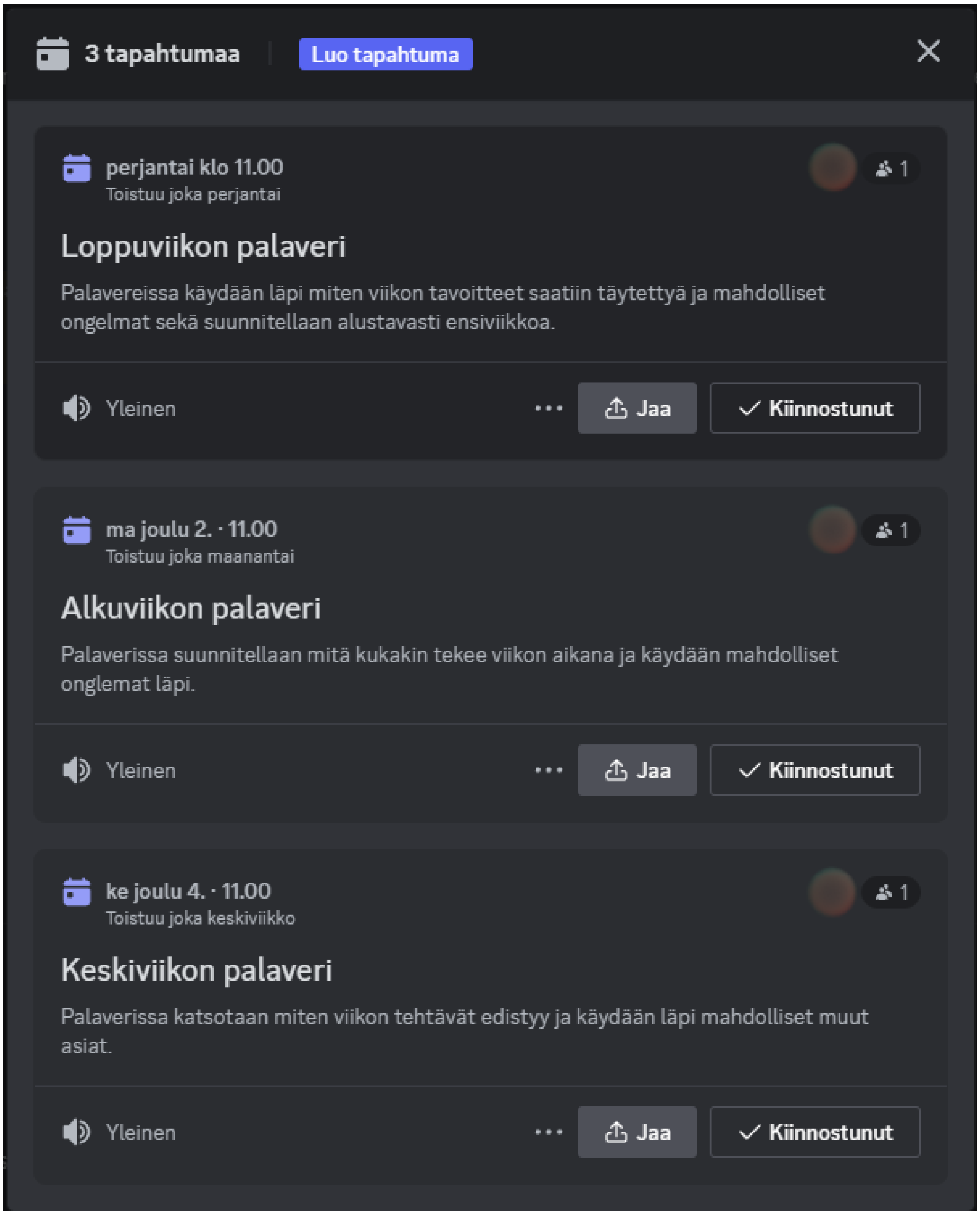
Kirjoittajalla oli haasteita saada NativeWind, TailWind kirjasto React Nativelle, toimimaan projektissa. Monista yrityksistä huolimatta koodi ei jostain syystä saanut NativeWindin tyylejä latautumaan sovellukseen, vaikka mitään virheilmoituksia ei tullut. Myöskään Tailwind-rn, joka on vaihtoehto NativeWindille, ei toiminut. Sitten kuitenkin neljännellä asennuksella NativeWind alkoi toimimaan ja päästiin aloittamaan projektin tyylittely.

Ensimmäisessä sprintissä kirjoittaja myös teki kehitystiimille Discord-palvelimen. Kuva 9 esittää palvelimen kanavat ja tekstikanavat oli tarkoitettu nimensä mukaiseen sisältöön, jotta selkeys säilyy ja asiat löytyvät nopeasti tarvittaessa.



Kuva 9. Discord-palvelimen rakenne.

Kuva 10 näyttää, kun palvelimelle myös luotiin toistuvat tapahtumat viikoittaisille palavereille, jotka pidettiin aluksi maanantaisin, tiistaisin ja perjantaisin kello 11 ja myöhemmin joka päivä kello 10. Maanantaisin käytiin läpi ja suunniteltiin viikon tehtävät ja tavoitteet kaikille, mahdolliset ongelmat ja mahdolliset muut asiat. Keskiviikkoisin oli nopea tilannekatsaus ja perjantaisin käytiin läpi, mitä viikolla saatiin aikaan ja miten se onnistui. Palavereissa käytiin myös läpi mahdolliset ongelmat ja keskusteltiin alustavasti seuraavan viikon tehtävistä.



Kuva 10. Discordin palaveritapahtumat.

Näiden lisäksi sovittiin, että projektinjohtaja pitää tilaajan kanssa oman palaverin aina sprintin alussa, jossa käydään sprintin tapahtumat ja käydään läpi seuraavan sprintin tavoitteita.

Tiimityö lähti hyvin käyntiin ja sovitut asiat piti. Kaikki sprintille asetetut tavoitteet saavutettiin.

6.2 Sprint 2 tulokset

Sprintille asetetuttuja tikettejä suoritettu 7/7.

Kirjoittajan toisen sprintin tavoitteena oli saada kartta integroitua mukaan sovellukseen. Tämä onnistuikin hyvin MapView-natiivikomponenttia käyttäen. Natiivikomponentin käyttäminen tosin rikkoi Web-version, mutta tämä ei ollut ongelma, koska sovellus oli tarkoitus kehittää tässä vaiheessa vain mobiilialustoille. Sprintin lopussa kirjoittaja myös integroi Google Maps API:n sovellukseen, jotta saatiin toimiva hakutoiminto karttaan. Lisäksi kirjoittaja tyylitteli sovellusta noudattaen vakioita teemavärejä, jotka vaihtuvat puhelimen asetusten mukaan.

Toisen Sprintin alussa tilaajan kanssa pidetyssä palaverissa keskusteltiin suurimmilta osin sovelluksen ulkonäöstä. Kehitystiimi ei ollut aiemmin aivan ymmärtänyt tilaajan visiota ja tätä alettiin ratkaisemaan. Kuvassa 11 nähdään kirjoittajan tekemä tilaajan esimerkkisivun mukainen kotisivu, jossa on halutun väriteeman mukaisia animoituja kategoriakuplia ja tuotehakupalkki sekä kuvassa 12 kehitystiimin versio etusivusta.

Kuva 11. Tilaajan vision mukainen etusivu.

Kuva 12. Kehitystiimin vision mukainen etusivu.

Kehitystiimin kanssa keskusteltaessa kävi kuitenkin nopeasti ilmi ongelmia tilaajan vision mukaisessa toteutuksessa. Haluttujen värien takia sovellus ei olisi niin selkeä ja saavutettavuus kärsisi. Värit eivät myöskään sopineet oikein tummaan eikä vaaleaan teemaan, jotka puhelinsovelluksista usein löytyy. Myös epäselvä rakenne, liikkuvat osat sekä haluttu joka suuntainen skrollaus vaikeuttaisivat sovelluksen opittavuutta ja todennäköisesti vähentäisi käyttäjien kiinnostusta sovellukseen. Kehitystiimin idea etusivusta taas oli hiukan tylsä ja persoonaton.

Sprintin 2 lopetuspalaverissa keskusteliin tilaajan kanssa näistä asioista ja päästiin hyvin yhteisymmärrykseen siitä, että liikkuvat elementit sovelluksessa tulee poistaa, mutta haluttu ruskea, harmaa ja beige väriteema, kuplat ja pehmeät muodot tulisi säilyttää. Kompromissina näiden kahden välille suunniteltiin versio, jossa on päällekkäin karusellimaisia rullia, joissa on kuplan muotoisia laatikoita tuotteille ja kaupoille, joita voi selata sivusuuntaan. Kupliin laitetaan kuvia tuotteista ja niihin tulee teemanväriset reunat, kuplamuotoja lisätään muuallekin sovellukseen, ja tilaaja suunnittelee haluamansa värit sovelluksen tummaan tilaan.

Sprintin 2 aikana kehittäjä 1 ja kehittäjä 2 saivat tehtyä Profiili-, Asetukset- ja Tilin asetukset siten, että kehittäjä 1 teki suurimmilta osin toiminnallisuuden kuten profiilin tietojen vaihdon, ja kehittäjä 2 tyylitteli uudet sivut ja muuta sovellusta.

Sprintti onnistui hyvin muilta osin, mutta projektipäälliköllä oli ongelmia, koska jokin projektin riippuvuus (Dependency) koitti jatkuvasti asentaa Reactin version 19.0.0 vaikka tällaista ei ole edes olemassa, saati sitten projektin riippuvuudet määrittävässä package.json-tiedostossa. Vaikka projektin riippuvuudet sisältävät node_modules-kansio ja package-lock.json poistettiin ja vaikka package.json issakin oli oikea versio, piti React 18.3.1 asentaa manuaalisesti, jotta riippuvuuskonflikti saatiin korjattua.

Palaverit pidettiin maanantaisin, keskiviikkoisin ja perjantaisin, ja ne kestivät 15–60 minuuttia. Niissä keskusteltiin siitä, miten on sujunut, mitä tehdään, mitä tullaan tekemään, ratkaistiin ongelmia sekä sovittiin joululomat.

Kaikkia sprintin tavoitteita ei saavutettu. Sprintti silti edisti projektia reilusti. Dummy-kaupat ja -tuotteet siirrettiin seuraavaan sprinttiin.

6.3 Sprint 3 tulokset

Sprintille asetetuttuja tikettejä suoritettu 8/8.

Kolmannen sprintin alussa tapaamiset vaihdettiin maanantai, tiistai ja keskiviikko kello yhdestätoista pidettäväksi jokainen arkipäivä kello kymmenen. Maanantaisin oli alkuviikon palaveri, jossa suunniteltiin viikkoa, keskiviikkoisin pidettiin lähipäivä jollain Metropolian toimipisteellä ja perjantaisin tarkasteltiin viikon edistymistä ja suunniteltiin seuraavaa viikkoa. Tiistaisin ja torstaisin oli nopea tilannekatsaus, jossa käytiin läpi mahdolliset ongelmat.

Sprintin päätavoitteena oli tyylitellä sovellus yhtenäisesti ja luoda uusia sivuja sekä toiminnallisuus tuotteille backendiin. Kaikki tavoitteet saavutettiin ja sovellus alkoi näyttää tässä vaiheessa jo asialliselta (kuvat 13 ja 14).

Kuva 13. Sovelluksen uusi etusivu tummalla teemalla.

Kuva 14. Sovelluksen uusi etusivu vaalealla teemalla.

Sprintin aikana tuotteille ja kaupoille luotiin omat sivut, joihin lisättiin toiminnallisuuksia ja tyylittelyjä. Karttaan lisättiin hakutoiminto, joka mahdollistaa käyttäjän paikantamisen sekä vaatekauppojen etsimisen tietyn säteen sisältä Googlen tietokannasta tai kovakoodatuista Shop Easyn omista kaupoista. Sovellusta paranneltiin tyylillisesti, toiminnallisuuksia kehitettiin, ja koodia siistittiin koko projektista tämän sprintin aikana.

Kolmannen sprintin aikana tilaaja oli alkanut ottamaan yhteyttä ostoskeskuksiin ja kauppoihin sekä luonut Shop Easylle nettisivut osoitteeseen shopeasy.fi. Liikkeet olivat osoittaneet kiinnostusta projektia kohtaan. Seuraavaksi oli tarkoitus aloittaa rahoituksen kerääminen perustettavalle startupille.

6.4 Sprint 4 tulokset

Sprintille asetetuttuja tikettejä suoritettu 8/8.

Heti sprintin alussa kirjoittaja syventyi opiskelemaan Google Places API:a. Tämän jälkeen karttasivu kehittyi näyttämään eri paikkoja eri väreillä, ja kauppakeskukset saivat omat sivut, johon haettiin kauppakeskuksen kuvat ja tiedot Googlesta. Kehittäjä 1 teki kuvakarusellin tuote- ja kauppakeskussivuille, ja kehittäjä 2 teki kauppasivun sekä tekstipohjaista sekä kategoriaoilla haun toiminnallisuutta.

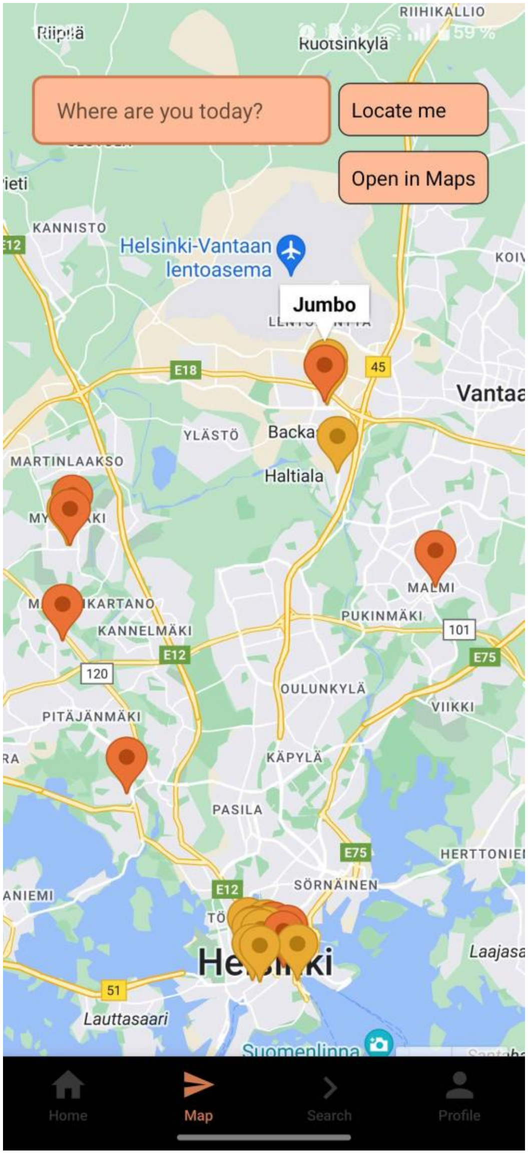
Sprintin aikana saatiin aikaan paljon, kartta edistyi huomattavasti ja siihen lisättiin esimerkiksi reitin esitys paikannetun käyttäjän luota haluttuun kauppaan tai kauppakeskukseen ja läheisten kauppojen haku ja esitys kartalla. Myös tuote- ja hakusivut saivat toiminnallisuutta ja tyylit sekä pienempiä muita muutoksia ympäri sovellusta. Neljännessä sprintissä lisättiin myös huomattava määrä yksikkötestejä koodiin.

6.5 Sprint 5 tulokset

Sprintille asetetuttuja tikettejä suoritettu 6/8.

Viidennessä sprintissä saatiin tehtyä kategorioita ja tuotteita niille, sekä niiden haku ja tuotteiden varaus. Lisäksi kirjoittajan toimesta sovellukseen lisättiin vaihtoehdoksi englannin ja ruotsin kieli koko sovellukseen, ja etusivulle tuli myös sijaintivalitsin. Lisäksi sovellusta viimeisteltiin, yhtenäistettiin ja lisättiin tilaajan pyytämiä asioita sovellukseen kuten logosivu, joka aukeaa hetkeksi sovelluksen auetessa sekä monia muita pienempiä muutoksia.

Myös karttasivua (kuva 15) paranneltiin huomattavasti ja siitä poistettiin aiemmin tehty reitin esitys, ja toiminto siirrettiin kokonaan käyttäjän järjestelmän karttasovellukseen. Kuvassa näkyy kartassa tummemman ruskealla Shop Easyn tietokannasta haettuja kauppakeskuksia, joiden tiedot on haettu Google Places APIn kautta Googlesta löytyvillä tiedoilla ja samalla tavalla haetut normaaleita kauppoja vaaleammalla ruskealla. Paikan pinniä (markeria) ja nimikylttiä (calloutia) painamalla sovellus vie kaupan tai kauppakeskuksen omalle sivulle (kuva 16).



Kuva 15. Kartta-sivu viidennessä sprintissä.

Kuva 16. Kauppakeskus-sivu viidennessä sprintissä.

Kartta tarkastaa sovelluksen kartan koon (Viewport) ja hakee kaupat ja kauppakeskuksen hiukan sitä pienemmältä alueelta. Jos käyttäjä menee alueelle, jossa tietokannasta ei löydy yhtään Shop Easyn kauppaa, tulee ilmoitus ”Hups! Emme ole täällä vielä, mutta laajennumme kokoajan.” Viewportin zoomia on rajoitettu kumpaankin suuntaan sopivalle etäisyydelle ja paikan pinniä painamalla voi avata paikan joko valitsemassaan karttasovelluksessa tai Google Mapsissa Androidilla ja Apple Mapsilla IOSilla.

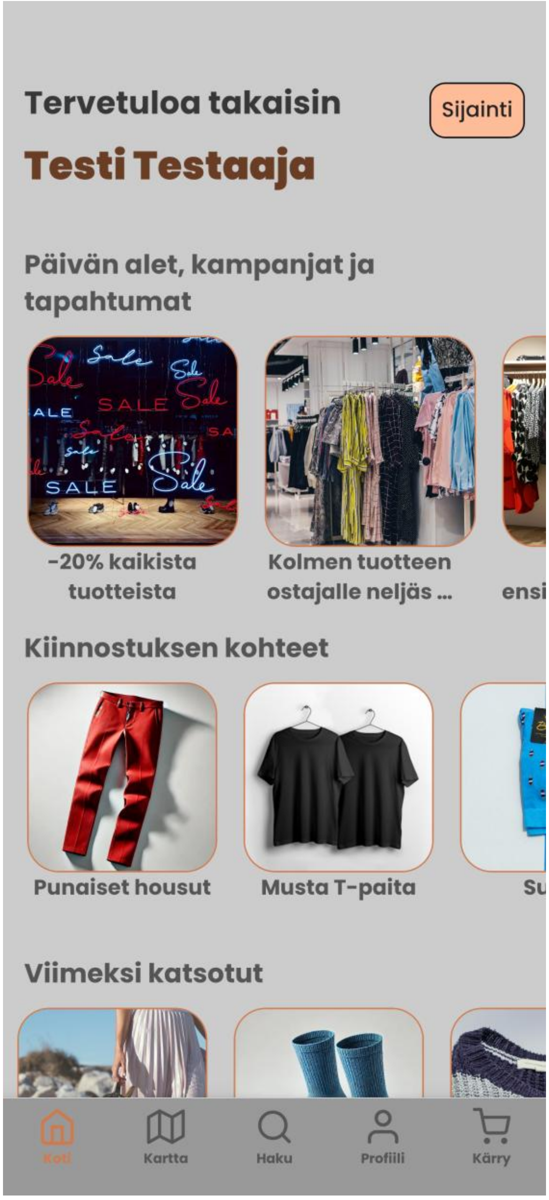
Viidennen sprintin loppupuolella alettiin tilaajan ja projektipäällikön kesken keskustella tulevan yrityksen jatkosta. Prototyypin kehityksen loputtua 2.3. jäätäisiin rahoituksen keräys tauolle. Sovelluksesta oli tehtävä siis hyvännäköinen rahoittajia ja yrityksiä varten, ja loppuaika kehityksestä käytettiin tähän. Priorisoidaan suuremmat kokonaisuudet kuten haku ja kartta ja niiden toiminnallisuus, sekä pienemmät tyylilliset asiat ja yhtenäistetään sovellus. Itse kaiken toiminnallisuus jää ulkonäön varjoon, mutta asiat kehitetään sitten, kunhan rahoitus yritykselle on kerätty.

Kaikkia sprintin tavoitteita ei saavutettu ja viimeiseen sprinttiin siirrettiin haun valmistuminen ja kauppakeskusten kauppalistaus.

6.6 Sprint 6 tulokset

Sprintille asetetuttuja tikettejä suoritettu: 6/6.

Kuudennessa sprintissä tehtiin hakusivua ja sen toiminnallisuutta, lisättiin tuotteita ja niille kuvia sekä tehtiin alkukyselykaavake ja muita parannuksia ympäri sovellusta. Varsinaisia sprintin tavoitteita ei ollut kuin sovelluksen viimeistely rahoituskierrokselle sopivaksi ja saada sovellus rakennettua. Kuvissa 17 ja 18 näkyy etu- ja hakusivu vaalealla teemalla kuudennen sprintin loputtua.



Kuva 11. Shop Easyn etusivu.

Kuva 12. Shop Easyn hakusivu.

7 Pohdinta

Koko projektin ajan kehitystyö sujui erinomaisesti. Lopputuloksena saatu sovellus vastasi sitä mitä tilaaja halusi, ja kaikki alussa Jiran backlogiin listatut, sovellukseen mukaan halutut ominaisuudet saatiin prototyyppiin yrityksen rahoituksenkeruuta edellyttävälle tasolle valmiiksi. Projektin budjettia ei ylitetty, koska sitä ei ollut määritelty, ja prototyypin kehityksen aikaiset kulut käsittivät oikeastaan vain alku- ja loppupalaverien virkistykset ja ruuat ynnä muut, jotka tilaaja kustansi. Projektille ei alussa asetettu tiukkaa aikaraamia vaan sen tavoite oli, että kehityksen lopussa olisi tuotettu sovellus, jota pystyy markkinoida eteenpäin, ja tämä saavutettiin reilussa kolmessa kuukaudessa. Kommunikaatio projektin aikana toimi todella onnistuneesti koko tiimin kesken.

Scrumin ja Agilen valinta projektin kehitysmetodologioiksi oli erinomainen valinta, koska nämä sopivat hyvin Shop Easyn pienen tiimin ja suuresti määrittelemättömän projektin tarpeisiin. Kuten Tracy Brower (2019) toteaa artikkelissaan, on Agile muutenkin todella tehokas metodologia ohjelmistokehitykseen ja Scrumia käyttäessä saatiin reagoitua nopeasti muuttuviin tavoitteisiin ja ongelmiin päivittäisissä ja viikoittaisissa palavereissa.

Kirjoittajan rooli Scrum Masterina oli myös onnistunut. Aiempi kokemus ryhmän ja projektien vedosta oli eduksi, mutta tässä projektissa vetovastuu korostui huomattavasti verrattuna aiempiin hankkeisiin johtuen siitä, että nyt vastuu tapaamisista, työnjaosta ja projektin etenemisestä oli oikeasti kirjoittajan vastuulla. Onneksi tilaaja ymmärsi hyvin mahdolliset ristiriidat ja eriävät näkemykset projektin ulkonäöstä ja toiminnallisuudesta ja suostui suurimpaan osaan projektipäällikön ja kehitystiimin muokkausehdotuksista. Kehittäjät taas kuuntelivat tarkkaan kirjoittajan ideoita ja tilaajalta saatuja pyyntöjä sekä ohjeita ja toimivat näiden mukaan, mutta toisaalta taas osasivat toimia itsenäisesti omien kokonaisuuksiensa kanssa kuitenkin mielipiteitä ja neuvoja kuunnellen. Koko projektin ajan prototyypin toiminnallisuuksia tehtiin myös kaikki yhdessä tai pareittain, varsinkin kuin jotkut projektin osat menivät päällekkäin, kuten esimerkiksi hakutoiminto ja tietokanta, jotka pääasiassa kehittivät eri kehittäjät.

Aina päivittäiset tapaamiset eivät välttämättä olleet kovinkaan sisältörikkaita, mutta loi tietyn rutiinin projektin kehitykseen, lähensi tiimiä, ja näin myös pienemmät ajatukset ja ongelmat saatiin heti käsittelyyn. Joskus etätapaamiset eivät välttämättä saattaneet kestää viittä minuuttia pitempää, mutta ajoittain reilusti toista tuntia. Päivittäiset tapaamiset myös varmistivat sen, että kaikki pysyivät koko ajan hyvin perillä siitä, miten projekti etenee, mitä kukakin tekee tai tulee tekemään seuraavaksi. Hyvän kommunikaation ansiosta turhaa työtä tai konflikteja ei juuri syntynyt koko projektin aikana.

Kirjoittajan johtamis- ja projektinhallintataidot kasvoivat merkittävästi projektin aikana. Alku oli hieman hankalaa rajoitetun kokemuksen takia, mutta projektin edetessä kehitys oli suuri. Aluksi palaverit olivat hieman sekaviakin, ja niiden dokumentointi olematonta mutta jo vuodenvaihteessa projektissa käytetyt metodologiat oli opittu paremmin, työkalujen käyttö alkoi olla sujuvaa, tapaamiset järjestellympiä ja kommunikaatio kumpaankin suuntaan parempaa. Projektipäällikkö adaptoitui jatkuviin muutoksiin ja hänen ja koko tiimin työskentelystä tuli sujuvampaa ja tehokkaampaa.

Jirasta kirjoittajalla ei ollut aiempaa kokemusta ennen Shop Easy -projektia, mutta sen käyttöönotto oli suoraviivaista ja Jira antaa todella hyvän kuvan koko projektin kulusta ja nyt sen oppineena tulee varmasti olemaan mukana tulevissa projekteissa.

Jos jotain projektin teossa pitäisi muuttaa, olisi dokumentointi alusta alkaen parempaa. Nytkin se toimi, mutta projektin edetessä huomasin, että myös kattava dokumentaatio esimerkiksi palavereista ja sovituista asioista on tärkeässä asemassa. Lisäksi testausta olisi voinut tehdä enemmän, Jestillä tehtyjen yksikkötestien, integraatiotestien ja käyttäjätestauksen lisäksi ei muuta juuri ehtinyt mukaan sen takia että ehdittiin saamaan kaikki halutut toiminnallisuudet ainakin osittain prototyyppiin. Tässä vaiheessa tuotteen ulkonäöllä oli paljon enemmän vaikutusta kuin itse toiminnallisuudella, koska täydellisesti toimiva tuote ei ollut missään vaiheessa projektin tarkoitus. Hankkeen tarkoituksena oli saada prototyyppi Shop Easyn sovelluksesta, jota pystyy esittelemään ja sillä keräämään rahoitusta tulevaa perusteellista jatkokehitystä varten.

Käyttäjätestausta kirjoittaja teki ahkerasti antaen sovellusta käytettäväksi ystäville, perheenjäsenille sekä muille IT-alalla työskenteleville tai sitä opiskeleville. Käyttäjätestauksesta saatiin suuri määrä arvokasta dataa ja tämä vaikutti huomattavasti sovelluksen lopulliseen toiminnallisuuteen sekä sen käyttöliittymään, mikä selvensi navigointia sekä ulkoasua. Testauksessa kävi ilmi asioita kuten se, että navigaatio sovelluksen sivujen ja toiminallisuuksien välillä ei aina ollut suoraviivaista tai johdonmukaista ja että jotkin asiat oli tehty liian epäselväksi tai vaikeaksi. Sitten jos rahoitus saadaan ja Shop Easy -projekti jatkuu, kokonaisvaltainen testaus otetaan varmasti tiiviimmin mukaan.

Muuta muutettavaa kirjoittaja ei prototyypin kehityksestä keksi, ja projekti oli kokonaisuudessaan onnistunut.

8 Loppumietteet

Shop Easyn kehitys jatkuu prototyypin valmistumisen jälkeen viimeistään tulevana kesänä, ellei uusia muutoksia tule ja kirjoittaja sekä kehittäjä 1 jatkavat projektissa tilaajan kanssa. Tilaaja aloittaa markkinoimaan sovellusta ja keräämään sille rahoitusta. Perustettava yritys jaetaan osallisten ja rahoittajien kesken. Markkinoinnin puolella haasteita tulee varmasti olemaan rahoituksen keruussa, mutta myös asiakkaiden hankinnassa.

Sovelluksen jatkokehityksessä taas on edessä ainakin kassajärjestelmien integraatio sovellukseen sekä saadun datan tekeminen vertailukelpoiseksi keskenään. Tämä voi luoda huomattavia haasteita, mutta tiimin kesken on asennoiduttu tähän, koska nähdään että sovelluksella on suuri potentiaali. Tämän lisäksi optimoitavaa on paljon, tietoturvallisuus pitää ottaa huomioon, testaus, datankeruu ja -käyttö, sovelluksen skaalautuminen tuleviin käyttäjämäärin ja moni muu asia, joita ei vielä edes tiedosteta.

Näistä asioista huolimatta odotetaan innolla tulevaa.

Projekti oli onnistunut vähintään oppimiskokemuksena, ja uskon, että kaikki osalliset ovat tyytyväisiä lopputulokseen ja iloisia matkasta, joka tähän meidät toi.

Lähteet

Agile Alliance. 2024. A short history of Agile. Verkkoaineisto. Agilealliance.org. <https://www.agilealliance.org/a-short-history-of-agile/>. Luettu 15.12.2024.

Agile Alliance. 2025. Extreme Programming. Verkkoaineisto. Agilealliance.org. <https://www.agilealliance.org/glossary/xp/>. Luettu 23.1.2025.

Agile Manifesto. 2001a. Ketterän ohjelmistokehityksen julistus. Verkkoaineisto. Agilemanifesto.org. <https://agilemanifesto.org/iso/fi/manifesto.html>. Luettu 12.12.2024.

Agile Manifesto. 2001b. Julistuksen takana olevat periaatteet. Verkkoaineisto. Agilemanifesto.org. <https://agilemanifesto.org/iso/fi/principles.html>. Luettu 15.12.2024.

Brower, Tracy. 2019. Is Agile Really Worth It? Evidence Says Yes—If You Do These 4 Things. Forbes. <https://www.forbes.com/sites/tracy-brower/2019/10/06/is-agile-really-worth-it-evidence-says-yes-if-you-do-these-4-things/>. 6.10.2019. Luettu 21.4.2025.

Chaudhary, Sristhi. 2022. How GitHub became the largest source code host? Verkkoaineisto. Medium.com. <https://srishtichy.medium.com/how-github-be-came-the-largest-source-code-host-a1c37ea5e5f>. 20.3.2022. Luettu 14.1.2025.

Codecademy Team. 2024. What Is an IDE? Verkkoaineisto. Codecademy.com. <https://www.codecademy.com/article/what-is-an-ide>. Luettu 17.12.2024.

Day, Brett. 2024. What is project management? Verkkoaineisto. Forbes.com. <https://www.forbes.com/advisor/business/what-is-project-management/>. Päivitetty 22.10.2024. Luettu 11.12.2024.

Odazie, Divine & Iheanacho, Amarachi. 2023. A Brief History of DevOps and Its Impact on Software Development. Verkkoaineisto. Everythingdevops.dev. <https://everythingdevops.dev/a-brief-history-of-devops-and-its-impact-on-software-development/>. Luettu 19.2.2025.

Eficode. 2013. Mitä on DevOps? Verkkoaineisto. Eficode.com. <https://www.eficode.com/fi/blog/mita-on-devops>. Päivitetty 15.3.2024. Luettu 19.2.2025.

GeeksForGeeks. 2024. Waterfall Model – Software Engineering. Verkkoaineisto. Geeksfogeeks.org. <https://www.geeksforgeeks.org/waterfall-model/>. Päivitetty 18.10.2024. Luettu 13.1.2025.

Hodges, Nick. 2015. Targets and Estimations in Software Development. Verkkoaineisto. Medium.com. <https://medium.com/nickonsoftware/targets-and-estimations-in-software-deveo-c99c66f20738>. Päivitetty 19.10.2024. Luettu 19.12.2024.

Radigan, Dan. 2025. Kanban. Verkkoaineisto. Atlassian.com. <https://www.atlassian.com/agile/kanban>. Luettu 7.3.2025.

Scrum.org. 2024. What is Scrum? Verkkoaineisto. Scrum.org. <https://www.scrum.org/resources/what-scrum-module>. Luettu 16.12.2024.

Siddiqui, Asim Rais. 2021. Lean Philosophy: The Way Of Business That Gave Rise To Industry Giants. Verkkoaineisto. Forbes.com <https://www.forbes.com/councils/theyec/2021/01/14/lean-philosophy-the-way-of-business-that-gave-rise-to-industry-giants/>. 14.1.2021. Luettu 17.4.2025.

Tsokkinen, Christina. 2025. Shop Easy. Verkkoaineisto. Shopeasy.fi. <https://www.shopeasy.fi/>. Luettu 26.2.2025.

Projektisuunnitelma

Shop Easyn prototyypin kehityksen projektisuunnitelma