Kolme koodia: ohjelmistokehittäjän portfolio

Marika Röyhkiö

Haaga-Helia ammattikorkeakoulu

Tradenomi

AMK-Opinnäytetyö

2025

Tiivistelmä

Tekijä(t)
Marika Röyhkiö

Tutkinto
Tradenomi

Raportin/Opinnäytetyön nimi
Kolme koodia: ohjelmistokehittäjän portfolio

Sivu- ja liitesivumäärä
30 + 0

Tässä opinnäytetyössä esitellään kolme ohjelmistokehitysprojektia, jotka toteutettiin erilaisissa konteksteissa: opintokurssin, vapaa-ajan itsenäisen kehitystyön sekä asiakastyön kautta. Töiden valitseminen portfolioon perustui kuitenkin pääasiallisesti projekteissa käytettyihin teknologioihin, sekä tunnistettuihin työskentelymalleihin. Töiden erilaiset kontekstit ja lähtökohdat otettiin huomioon projektien valinnassa sekä niiden analysoinnissa, mutta ne eivät ole opinnäytetyön kannalta olennaisin osa,

Työn tavoitteena on tarkastella näiden projektien kautta niissä käytettyjä ohjelmointiteknologioita ja työskentelymalleja, vertailla niiden hyötyjä ja haasteita sekä reflektoida tekijän ammatillista kehittymistä. Projektien analyysejä työstäessä tekijä pääsi palaamaan teknologioiden dokumentaation pariin sekä syvemmin tarkastelemaan omia ratkaisujaan ja projektien toimivuutta, joka edesauttoi ammatillista kasvua.

Opinnäytetyön tietoperustassa syvennytään neljään keskeiseen teknologiaan: TypeScriptiin, Python-ohjelmointikieleen, Flask-kehykseen sekä Git versionhallintaan. Lisäksi käsitellään ohjelmistokehityksen yleisimpiä työskentelymalleja, kuten vesiputousmallia, ketteriä menetelmiä ja BT-mallia. Näiden pohjalta arvioidaan, miten eri valinnat vaikuttivat projektien etenemiseen, lopputulokseen ja tekijän oppimiskokemuksiin. Lähteinä on käytetty pääasiallisesti eri teknologioiden virallisia dokumentaatioita, jotta analyysissä voidaan reflektoida projektien työstämistä eri teknologioiden perusteisiin.

Työskentelymallien osalta tietoperustassa hyödynnetään erilaisia ohjelmistoalan ammattilaisten julkaisuja. Työskentelymallien seuraaminen projekteissa ei ollut täysin oppikirjan mukaista, ja analyysissä keskitytäänkin projektien työstämiseen niiltä osin, kuin se johonkin ohjelmistoalan työskentelymalliin sopii. Lisäksi arvioidaan, kuinka toimivia nämä työskentelymallien piirteet olivat jokaisen projektin kohdalla.

Tulosten perusteella ketterä kehitys osoittautui joustavimmaksi työskentelymalliksi erityisesti asiakasprojekteissa, kun taas vesiputousmalli soveltui rajatumpiin ja ennalta määriteltyihin tarpeisiin. Teknologiavalinnoilla oli merkittävä vaikutus projektien onnistumiseen ja kehitystyön tehokkuuteen. Analyysissä käydään läpi haasteita sekä kehityskohteita, joihin tekijä törmäsi teknologioiden osalta projektien toteutuksen aikana. Opinnäytetyö antaakin kokonaiskuvan ohjelmistokehittäjän osaamisen rakentumisesta käytännön projektien kautta, ja tekijän on mahdollista hyödyntää oppimaansa esimerkiksi omien ammatillisten vahvuuksiensa tunnistamisessa sekä mahdollisesti tulevaisuuden työnhakuun liittyvissä tilanteissa.

Asiasanat
Ohjelmistokehitys, portfolio, Typescript, Python,

Sisällys

1 Johdanto

Tässä opinnäytetyössä käsittelen kolmea ohjelmistoprojektia, jotka olen itse henkilökohtaisesti kehittänyt. Tavoitteena on käydä läpi projekteissa käytettyjä teknologioita ja työskentelymalleja. Työssä myös vertaillaan projekteihin valikoituneita teknologioita, niiden valintaperusteita sekä mahdollisia haasteita niihin liittyen. Samalla tavalla käsitellään työskentelymalleja sekä niiden sopivuutta projekteihin.

Käyn myös läpi projektien lähtökohtia, ja niiden mahdollisia vaikutuksia kehitystyöhön ja lopputulokseen. Portfolioon valitut ohjelmistot toteutettiin osana opintoihini liittyviä kursseja, tai täysin itsenäisesti vapaa-ajan projekteina. Projekteissa, niiden etenemisessä sekä lopputuloksissa on huomattavissa eroja, joita voidaan reflektoida lähtökohtiin ja työskentely-ympäristöihin nähden. Tämä ei kuitenkaan ole opinnäytetyön pääasiallinen näkökulma, mutta koin sen tärkeäksi osaksi, kun haluttiin analysoida projektityöskentelyn kokonaisuutta.

Opinnäytetyön tavoitteena on reflektoida omaa oppimista ja kokemuksia ohjelmistoalan teknologioiden ja työskentelymallien kanssa. Projektien parissa kohtasin haasteita sekä teknologiavalintojen että työskentelymallien parissa, ja niiden reflektoimisen tavoitteena on auttaa minua kehittymään ammatillisesti ja antamaan minulle paremmat lähtökohdat siirtyä työelämään omien vahvuuksien ja heikkouksien tunnistamisen kautta. Opinnäytetyön työstäminen myös antoi hyvän syyn palata projektien ja eri teknologioiden dokumentaation pariin, joka vielä syvensi osaamistani niiden saralla.

Opinnäytetyön tietoperustana on käytetty pääasiallisesti projekteissa hyödynnettyjen teknologioiden ja työskentelymallien virallista dokumentaatiota. Tietoperustaa rakentaessa vierailin myös lukuisilla ohjelmistoalan ammattilaisten hyödyntämillä foorumeilla, joihin tutustuin projektien kehitystyön aikana. Tätä kautta pystyin helpommin palauttamaan mieleeni kehitysprosessia ja työskentelyn etenemistä. En kuitenkaan hyödyntänyt näitä foorumikeskusteluita tai blogitekstejä varsinaisina lähteinä opinnäytetyölleni.

Alla avattuna keskeisiä käsitteitä, jotka esiintyvät opinnäytetyön myöhemmissä kappaleissa.

Komponentti – Ohjelmistokehityksessä on yleisesti ymmärretty, että komponentit ovat itsenäisiä ohjelmiston osia, joita voidaan yhdistää keskenään ohjelmistosysteemin luomiseksi (Sommerville, 2016, 467).

Käyttöliittymä – Se osa ohjelmistoa, laitetta tai sovellusta, joiden kautta käyttäjä seuraa ja ohjaa ohjelman toimintaa (Sanastokeskus, s.a.)

Backend – Sisältää kaikki sovelluksen tai ohjelmistokokonaisuuden osat, jotka siirtävät, käsittelevät ja vastaanottavat sen tietoja. Näihin osiin sisältyvät esimerkiksi tietokantayhteydet, lomakkeiden ja tiedostojen käsittelyt sekä kirjautumistoiminnot (Bautomo, s.a.a.)

Frontend – Verkkosivuston tai ohjelmiston selainpuoli. Frontend on se verkkosivun tai ohjelmistokokonaisuuden osa, jonka kanssa loppukäyttäjä on tekemisissä. Se sisältää sivuston tai sovelluksen käyttöliitymän (Bautomo, s.a.b.)

localStorage – localStorage objektin avulla ohjelma voi tallentaa tietoja paikallisesti selaimen muistiin. Tieto on käytettävissä aina, kun sivusto avataan, vaikka selain olisi suljettu välillä (Sofela, 2023).

Admin – Järjestelmänvalvoja tai hallintakäyttäjä. Ohjelmistokehityksessä admin-tili tarjoaa käyttöoikeudet sovelluksen tai järjestelmän hallintaan, kuten käyttäjien hallintaan ja asetusten muokkaamiseen. Admin-käyttäjillä on tyypillisesti korkeammat käyttöoikeudet kuin tavallisilla käyttäjillä (Lanchec, 2024).

Responsiivisuus – Responsiivisuus ohjelmistokehityksessä tarkoittaa sivujen ja ohjelmien suunnittelua niin, että ne skaalautuvat erilaisiin selaimiin, kuten mobiililaitteisiin, tabletteihin ja tietokoneisiin (Carver, 2014).

API rajapinta – Välittäjä, jonka kautta kaksi eri sovellusta voivat kommunikoida ja vuorovaikuttaa keskenään (SAP, s.a.).

Verkkoharavointi – Tekniikka, jossa ohjelma kerää dataa toisen ohjelman tuottamasta sisällöstä, mahdollisista tekijänoikeuksista välittämättä (Cloudfare, s.a.).

2 Tietoperusta

Tietoperusta käsittelee ohjelmointiteknologioita, joita opinnäytetyöhön valituissa projekteissa on käytetty, sekä ohjelmistokehityksen työskentelymalleja, joiden avulla projektit on toteutettu. Lisäksi se tarkastelee projektien teknologioiden ja työskentelymallien välisiä yhtäläisyyksiä ja eroavaisuuksia, sekä näiden tiimoilta tapahtuneiden päätösten taustoja.

Ohjelmointiteknologiat ja ohjelmistokehityksen työskentelymallit muodostavat perustan kaikille portfolion projekteille, ja niiden valinta tietoperustan keskittymiskohteiksi perustuu siihen, että jokaisessa portfolion projektissa teknologia ja työskentelymalli määrittelivät pitkälti projektin kulun, vaatimukset ja haasteet. Niiden vaikutukset ovat myös selkeästi havaittavissa projektien lopputuloksissa. Tietoperustan tehtävänä on tukea analyysia ja vertailua tarjoamalla teoreettinen ja käytännönläheinen pohja projektien arviointiin. Tässä luvussa keskitytään erityisesti teknologioihin ja malleihin, jotka olivat projektien toteutusvaiheessa ratkaisevissa rooleissa.

Teknologioiden osalta esittelyssä ovat TypeScript, Flask, Python ja Git-versionhallinta. Nämä kielet ja kehykset olivat käytössä joko yksin tai yhdessä kaikissa kolmessa portfolioprojektissa. Tarkastelu painottuu niiden käyttöön ohjelmistokehityksessä ja siihen, millaisia vahvuuksia ja rajoitteita niillä oli eri käyttötarkoituksissa. Tietoperustan osuudessa kiinnitetään huomiota teknologioiden ominaisuuksiin sekä niiden vaikutuksiin projektien teknisessä toteutuksessa. Tavoitteena on muodostaa käsitys siitä, miten valinnat vaikuttivat lopputuloksiin sekä kehitysprojektin sujuvuuteen.

Työskentelymallien osalta tietoperustassa käsitellään vesiputousmallia, ketterää kehitystä ja BT-mallia. Mallit tarjoavat erilaisia tapoja jäsentää ja toteuttaa ohjelmistoprojekteja, ja ne vaikuttavat suuresti siihen, millaisia vaiheita projektin toteutus pitää sisällään ja miten joustavasti kehitystyö voi vastata tuotteelle asetettujen vaatimusten ja tarpeiden muuttumiseen. Esittelyssä huomioidaan erityisesti mallien teoreettinen tausta ja käytännön soveltaminen, sekä niiden yhteys portfolion projektien työskentelytapoihin. Tietoperustan tavoitteena on antaa lukijalle kokonaiskuva niistä valinnoista ja rakenteista, joihin projektit nojautuvat, ja jotka mahdollistavat niiden vertailevan tarkastelun opinnäytetyön myöhemmissä luvuissa.

2.1 Ohjelmointiteknologiat

Tässä kappaleessa käsittelen keskeisiä teknologioita, joiden avulla toteutin portfoliossa olevat ohjelmistoprojektit. Kaikissa projekteissa ei ole hyödynnetty kaikkia teknologioita, mutta kokonaisuutena nämä teknologiat kattavat portfolion projektien keskeisimmät toteutustavat.

Teknologioina on esitelty ohjelmointikieliä, sovelluskehys sekä versionhallinta, joita hyödynsin projekteissa. TypeScript ja Python ovat projekteissa pääasiallisesti käytetyt ohjelmointikielet, ja Flask

on sovelluskehys, jota hyödynsin yhdessä Python ohjelmointikielen kanssa. Git-versionhallintaa hyödynnettiin myös kaikissa projekteissa. Käsittelen myöhemmissä kappaleissa syvemmin jokaisessa projektissa käytettyjä teknologioita. Nämä yksityiskohtaisemmat kuvaukset kuitenkin koskevat lähinnä teknologioita, jotka eivät olleet ratkaisevassa asemassa projekteja toteuttaessa.

2.1.1 TypeScript

TypeScript on Microsoftin kehittämä ohjelmointikieli, joka laajentaa JavaScriptiä staattisella tyypityksellä. Sen tavoitteena on parantaa koodin luettavuutta, ylläpidettävyyttä ja virheiden tunnistettavuutta kehitysvaiheessa. (TypeScript Team, 2025.) TypeScript tarjoaa JavaScriptia paremman tuen suurille projekteille, koska sen staattinen tyypitys löytää koodista virheet nopeasti, mikä ehkäisee ohjelman ajonaikaisia virheitä. TypeScript myös helpottaa koodin jäsentelyä ja rakennetta, helpottaen täten suurten ohjelmakokonaisuuksien hallintaa. TypeScriptissä on myös enemmän sisäänrakennettuja työkaluja koodaamisen helpottamiseksi, ja se tukee kehittyneitä ominaisuuksia kuten luokkia, luokkien välistä ominaisuuksien perimistä sekä rajapintoja (Nitin ja Puja, 2025). Näiden kautta TypeScript parantaa koodin uudelleenkäytettävyyttä.

TypeScriptin edut korostuvat erityisesti tiimeissä tehtävissä projekteissa, joissa useampi kehittäjä työskentelee saman koodin parissa. Kielen selkeä tyypitys ja automaattinen täydentäminen helpottavat koodin lukemista ja vähentävät väärinkäsityksiä. Lisäksi TypeScript on helppo integroida suosittuihin kehitysympäristöihin, kuten Visual Studio Codeen, jossa sen ominaisuudet tulevat tehokkaasti esiin. Tämä tekee siitä erityisen houkuttelevan valinnan projekteihin, joissa ylläpidettävyys ja pitkäaikainen laajennettavuus ovat keskeisiä vaatimuksia.

2.1.2 Flask

Flask on kevyt Python-pohjainen verkkosovelluskehys, joka tarjoaa joustavuutta ja helppokäyttöisyyttä erityisesti pienissä ja keskisuurissa projekteissa (Pallets Projects, 2010a). Sen keskeiset ominaisuudet sopivat hyvin mikropalveluarkkitehtuurien kehittämiseen, nopeaan prototypointiin sekä kokeiluihin. Sen pohjarakenne tukee hyvin kolmansien osapuolien laajennuksia. (Pallets Projects, 2010b).

Flaskissa on sisäänrakennettu kehityspalvelin ja tuki URL-reitityksille, mikä tekee siitä erinomaisen alustan backend-sovelluksen rakentamiselle. Sen templatemoottori Jinja2 mahdollistaa tehokkaan frontendin ja backendin yhdistämisen, jolloin käyttäjille voidaan tarjota dynaamisia ja pitkälle personoituja verkkosivunäkymiä. Tämä tekee Flaskista erityisen käyttökelpoisen tilanteissa, joissa sovelluksen ulkoasun ja toiminnallisuuden välillä tarvitaan joustavaa vuorovaikutusta.

Yksi Flaskin suurimmista vahvuuksista on sen muokattavuus. Kehittäjä voi valita juuri ne komponentit ja kirjastot, joita kehitettävä projekti tarvitsee, ja näin välttää tarpeettomia riippuvuuksia. Tämä voi myös tietyissä tilanteissa lisätä kehitystyön haastavuutta, sillä suuremmissa projekteissa Flask voi vaatia enemmän manuaalista työtä verrattuna valmiimpiin kehyksiin, kuten esimerkiksi Djangoon.

2.1.3 Python ja sen käyttö ohjelmistokehityksessä

Python on monipuolinen ja laajasti käytetty ohjelmointikieli, joka soveltuu sekä verkkosovelluksiin että datan käsittelyyn. Sen suosio perustuu selkeään syntaksiin ja laajaan kirjastoekosysteemiin. (Python Software Foundation, 2001). Sen helposti luettava syntaksi tekee siitä sopivan työvälineen sekä aloitteleville ohjelmoijille että kokeneille kehittäjille. Pythonin laaja kirjastoekosysteemi (esim. Flask, Django, Pandas, NumPy) sekä helppo integroituvuus muihin järjestelmiin ja ohjelmointikieliin kuuluvat sen keskeisiin etuihin.

Python soveltuu erinomaisesti niin verkkosovellusten kehitykseen (Flask, Django) kuin datatieteisiin, koneoppimiseen, tekoälytyöskentelyyn ja automaatioon. Verkkosovelluksissa Pythonin joustavuus näkyy siinä, että sen avulla voidaan toteuttaa sekä frontend- että backend-toimintoja. Lisäksi sen yhteensopivuus erilaisten tietokantajärjestelmien kanssa tekee siitä tehokkaan työkalun täysimittaisten sovellusten rakentamiseen.

Opinnäytetyön projekteissa Python oli erityisesti käytössä backend-kehityksessä. Se tarjosi vakaan ja joustavan alustan lomakkeiden käsittelyyn, tietokantayhteyksien rakentamiseen ja reititysten hallintaan. Pythonin käyttö edesauttoi tehokasta ja modulaarista koodausta, mikä helpotti ohjelmien ylläpitoa ja jatkokehitystä. Lisäksi sen dokumentaatio ja yhteisötuki auttoivat ratkaisemaan ongelmatilanteita tehokkaasti.

2.1.4 Git-versionhallinta

Git on hajautettu versionhallintajärjestelmä, jota käytetään laajasti ohjelmistokehityksessä koodin muutosten seuraamiseen, hallintaan ja yhteiskehittämiseen. Se mahdollistaa useiden kehittäjien työskentelyn saman projektin parissa ilman, että yksittäisten muutosten tallentaminen tai yhdistäminen aiheuttaa ristiriitoja. Git tallentaa koodin muutokset selkeästi vaiheittain, mikä mahdollistaa aiempiin versioihin palaamisen ja virheiden jäljittämisen tehokkaasti (Git, s.a.).

Kaikissa opinnäytetyön projekteissa Git toimi keskeisenä työkaluna kehitystyön hallinnassa. Vaikka projekteja työsti vain yksi kehittäjä, versionhallinta mahdollisti työvaiheiden selkeän dokumentoinnin ja kehityksen etenemisen seuraamisen. Erityisesti tilanteissa, joissa kokeiltiin uusia

ominaisuuksia tai tehtiin suurempia rakenteellisia muutoksia, Gitin avulla oli mahdollista säilyttää vakaa versio toimivasta kokonaisuudesta ja kehittää uusia ominaisuuksia omilla haaroillaan.

Gitin hyödyntäminen paransi myös ohjelmistokehityksen laatua. Se tuki systemaattista työskentelyä ja edisti omien ratkaisujen dokumentointia. Lisäksi GitHubin kaltaisten pilvipohjaisten Git-palveluiden avulla projektit olivat varmuuskopioituja ja tarvittaessa helposti jaettavissa. Git toimi paitsi teknisenä apuvälineenä, myös oppimisvälineenä, jonka kautta syveni ymmärrys siitä, miten ohjelmistoprojektit dokumentoidaan ja hallitaan ammattimaisessa kehitystyössä.

2.2 Ohjelmistokehityksen työskentelymallit

Tässä osiossa tarkastelen ohjelmistokehityksen työskentelymalleja, joita hyödynsin portfolioprojekteissa. Työskentelymallit ohjaavat kehitystyön etenemistä, vaiheiden rakennetta sekä sitä, miten suunnittelu, toteutus ja testaus jäsentyvät projektin aikana. Ne vaikuttavat ratkaisevasti siihen, miten projektin vaatimuksiin vastataan, ja kuinka ketterästi kehitystyö voi reagoida yllättäviinkin muutoksiin.

Portfolion projekteissa käytin erilaisia työskentelymalleja riippuen projektin lähtökohdista, tavoitteista ja kontekstista. En soveltanut kaikissa projekteissa samaa työskentelytapaa, mutta kokonaisuutena valitut mallit kuvastavat yleisimpiä ohjelmistokehityksessä käytettyjä lähestymistapoja. Niiden avulla voidaan tarkastella, millaisia eroja eri malleissa ilmenee muun muassa joustavuudessa, aikataulutuksessa ja vaatimusten hallinnassa.

Seuraavissa alaluvuissa esittelen tarkemmin vesiputousmallin, ketterän kehityksen ja BT-mallin keskeisiä periaatteita. Lisäksi tarkastelen, miten ne suhteutuvat toteutettuihin projekteihin ja millaisia kokemuksia sain niiden soveltamisesta käytännössä.

2.2.1 Vesiputousmalli

Vesiputousmalli on lineaarinen ohjelmistokehitysmalli, jossa kehitys etenee vaiheittain ennalta määriteltyjen askelten mukaisesti. Jokainen vaihe aloitetaan vasta, kun edellinen on saatu päätökseen, eikä edellisiin vaiheisiin palata (Atlassian, s.a.). Tämä malli soveltuu erityisesti projekteihin, joissa vaatimukset ovat selkeät ja muuttumattomat (Kirvan, Lutkevich & Lewis 2024.)

Vesiputousmallin mukaisessa työskentelyssä ensimmäinen vaihe on vaatimusmäärittely. Vaatimusmäärittelyvaiheessa selvitetään, mitä järjestelmän tai ohjelmiston tulee tehdä. Tässä vaiheessa kerätään ja dokumentoidaan kaikki vaatimukset kehitystyön suunnittelun selkeyttämiseksi.

Seuraava vaihe on suunnittelu, jossa määritellään järjestelmän arkkitehtuuri, tekniset ratkaisut sekä toimintalogiikka. Tämä vaihe luo pohjan projektin toteutukselle, jossa suoritetaan varsinainen ohjelmointi ja järjestelmän rakentaminen.

Kun toteutus on valmis, siirrytään testausvaiheeseen, jossa varmistetaan, että järjestelmä toimii suunnitellusti ja täyttää sille asetetut vaatimukset. Viimeinen vaihe on käyttöönotto, jossa valmis järjestelmä siirretään tuotantoon ja toimitetaan käyttäjille (Kirwan, Lewis & Lutkewich, 2024)

2.2.2 Ketterä kehitys

Ketterät menetelmät, kuten Scrum ja Kanban, korostavat iteratiivista kehitystä, tiivistä yhteistyötä ja nopeaa palautteen saamista. Ketterä kehitys soveltuu erityisesti projekteihin, joissa vaatimukset voivat muuttua kehitysprosessin aikana. (Agile Alliance, s.a.).

Ketterän kehityksen periaatteissa korostetaan joustavuutta ja asiakaslähtöisyyttä. Yksi keskeinen ajatus on, että asiakasyhteistyö on tärkeämpää kuin muodolliset sopimusneuvottelut, sillä jatkuva vuoropuhelu varmistaa, että kehitystyö todella vastaa projektin todellisia tarpeita. Lisäksi ketterässä kehityksessä painotetaan toimivaa ohjelmistoa enemmän kuin laajaa dokumentaatiota, sillä tavoitteena on saada aikaan konkreettisia ja toimivia ratkaisuja mahdollisimman nopeasti.

Lopuksi, ketterän kehityksen periaatteiden mukaan muutoksiin sopeutuminen on tärkeämpää, kuin ennalta määritellyn suunnitelman tiukka noudattaminen, koska projektin vaatimukset ja olosuhteet voivat muuttua kehitysprosessin aikana (Elmhurst University, s.a.).

2.2.3 BT-malli

BT-malli eli Business Technology-malli on kehitetty yli 50 Pohjoismaisen yrityksen, julkisen sektorin toimijan ja yliopiston yhteistyönä, ja se on ollut käytettävissä vuodesta 2009 lähtien. (BT-malli s.a.a). Se on avoin johtamisen viitekehys, joka tarjoaa kokoelman parhaita käytäntöjä informaatioteknologian suunnitteluun, rakentamiseen ja johtamiseen. Se on kehitetty tukemaan teknologian hyödyntämistä liiketoiminnan tarpeiden mukaisesti. BT-malli yhdistää liiketoiminta- ja teknologiaosaamista, ja sen tarkoitus on varmistaa, että teknologiaprojektit tukevat strategisia tavoitteita.

BT-malli koostuu useista osa-alueista, jotka kattavat esimerkiksi palvelunhallinnan, kehitystyön ohjauksen, portfolionhallinnan sekä teknologia-arkkitehtuurin. Se painottaa jatkuvaa oppimista ja kehitystyön arvontuottoa. BT-malli ei ole suoranaisesti yksityiskohtainen ohjelmointikehys, vaan sen avulla voidaan varmistaa, että teknologia tukee toiminnan tavoitteita (BT-malli s.a.)

Vaikka BT-malli ei ollut keskeisessä käytössä omissa portfolioprojekteissani, sen periaatteet ovat läsnä erityisesti portfolion kolmannen projektin taustalla, jossa toteutin verkkosivuston oikealle

yritysasiakkaalle. Projektin suunnittelu, dokumentointi ja aikataulutus noudattivat osittain BT-mallin mukaista ajattelua.

3 Portfolion esittely ja tuotosten analyysi

Tässä luvussa esitellään kolme ohjelmistoprojektia, jotka muodostavat opinnäytetyöni portfolion. Projektit syntyivät eri konteksteissa: yksi toteutettiin osana opintokurssia, toinen vapaa-ajan itseopiskeluprojektina ja kolmas osana työharjoittelua todelliselle asiakkaalle. Projektien valintaan vaikutti niiden vaihteleva toteutustapa, valmiusaste, teknologinen monipuolisuus sekä se, että ne kuvaavat hyvin kehityspolkuani ohjelmistokehittäjänä.

Kukin projekti esitellään omassa alaluvussaan, ja jokaisessa tarkastellaan projektin taustaa, teknologisia valintoja, toteutusta, haasteita sekä oppimiskokemuksia. Tämän lisäksi tarkastellaan myös työskentelytapoja: miten projektit on suunniteltu ja jäsennelty, millaisia menetelmiä niiden kehityksessä on hyödynnetty, ja miten nämä ovat vaikuttaneet lopputulokseen. Analyysi ei keskity pelkästään tekniseen toteutukseen, vaan tarkastelee myös kehitysprosessin kokonaisuutta kehittäjän näkökulmasta.

Luvun tavoitteena on paitsi dokumentoida toteutetut projektit, myös reflektoida omaa osaamistani sekä työskentelytapojeni kehittymistä. Projektit toimivat ikään kuin peileinä, jotka mahdollistavat omia vahvuuksiani, kehityskohtia, sekä ammatillista kasvuani. Luvun lopussa on projektien välistä vertailevaa analyysiä, joka auttaa tunnistamaan yhtäläisyyksiä ja eroavaisuuksia teknologioissa, työskentelymalleissa ja oppimisessa.

3.1 Projekti 1: Hetkien Arkisto-ohjelma

Hetkien Arkisto on sovellus, jonka avulla käyttäjä voi luoda päiväkirjamaisia merkintöjä, joihin hän voi lisätä kuvia sekä teema- ja fiilistunnisteita. Lisäksi käyttäjä pystyy seuraamaan teemojen ja fiillisten trendejä kuukausitasolla merkitsemiensä tunnisteiden perusteella. Trendien kuvaamiseen käytetään graafeja.

Projektiin kuuluu useita komponentteja, kuten merkintöjen lisäyslomake, teemanhallinta, kalenterinäkymä sekä visuaaliset analyysityökalut. Sovellus käyttää kalenterityylistä käyttöliittymää, johon käyttäjä voi lisätä merkintöjä, liittää niihin teemoja ja fiiliksiä sekä tarkastella kuukausittaista analyysia tekemisistään. Ohjelma tallentaa datan localStorageen, joka takaa tietojen säilyvyyden ilman erillistä backend-palvelinta.

Projekti syntyi seminaarityönä opintoihini kuuluvalla 'Ohjelmistokehityksen teknologioita (ajankohtaiset teknologiat)'- kurssia. Kurssin tavoitteena oli oppia soveltamaan ammattimaisen ohjelmistokehityksen teknologioita ja työkaluja (Haaga Helia, s.a.). SCRUM-menetelmää ja TypeScriptia käsiteltiin kurssin aikana, jonka vuoksi ne valikoituivat osaksi seminaarityötä, jonka tarkoituksena oli syvemmin tutustua johonkin käsiteltyyn teknologiaan.

3.1.1 Teknologiat ja työskentelymallit

Käytin projektin toteuttamiseen useita teknologioita. Sovelluksen pääkehyksen toteutin käyttäen Reactia. React mahdollistaa komponenttipohjaisen kehityksen ja tarjoaa tehokkaan tilanhallinnan (React, s.a.). Reactin avulla pystyin toteuttamaan ohjelmalle dynaamisen ja responsiivisen käyttöliittymän, jossa käyttäjä voi lisätä, muokata ja tarkastella merkintöjään helposti.

Projektin pääohjelmointikieli oli TypeScript, joka valikoitui käytettäväksi opintokurssin asettamien vaatimusten kautta. TypeScriptin käyttäminen Reactin yhteydessä toi mukanaan mahdollisuuksia luoda uudelleenkäytettäviä ja skaalautuvia komponentteja, joiden tyypityksen avulla oli helpompi ennaltaehkäistä virheitä. Tämä oli tärkeää erityisesti silloin, kun käyttöliittymään lisättiin monimutkaisempia toimintoja, kuten graafien visualisointia ja tunnisteiden käsittelyä.

Käyttäjän tekemien merkintöjen datan visualisointiin käytin Chart.js kirjastoa. Sen avulla toteutin merkintöihin lisättyjen fiilisten ja teemojen jakaumakaaviot. Chart.js mahdollistaa dynaamiset ja responsiiviset graafit, jotka päivittyvät käyttäjän syöttämien tietojen perusteella ja näyttävät merkintöjen väliset määrälliset suhteet (Chart.js, 2025). Tämän avulla käyttäjä pystyy analysoimaan menneitä merkintöjään ja niiden muutoksia reaaliajassa.

Ohjelman käyttöliittymän visuaalisen ilmeen toteutin hyödyntäen CSS:ää. CSS:n avulla varmistin responsiivisuuden eri laitteilla. Huomioin erityisesti käytettävyyden parantamisen värikontrasteilla, intuitiivisilla painikkeilla ja responsiivisella asettelulla. Lisäksi lisäsin animaatioita ja varjostuksia parantamaan visuaalista selkeyttä ja käyttäjäkokemusta.

Tietojen säilyttämiseen käytin LocalStoragea ilman erillistä backend-ratkaisua. Tämä mahdollistaa käyttäjän datan tallentamisen ilman erillistä kirjautumista tai tietokantayhteyttä (Long, 2024).

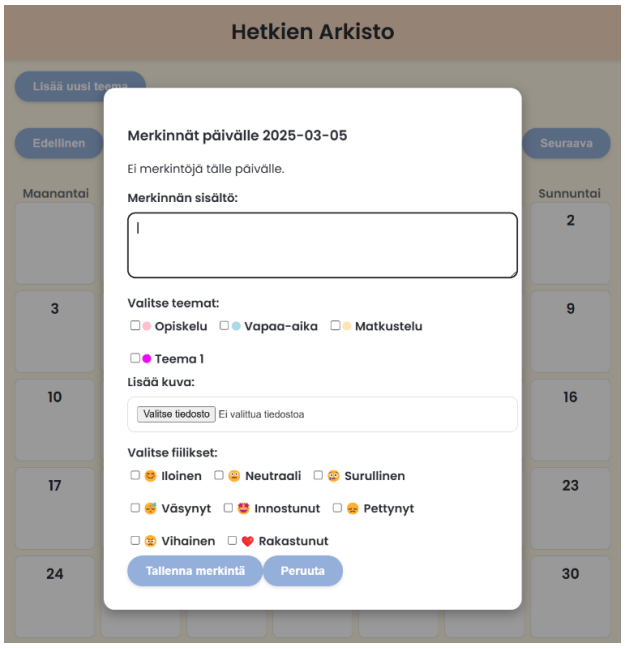
Lisäksi projektissa käytettiin Git-versionhallintaa, jonka avulla kehitystyötä oli helpompi hallita eri vaiheissa. Git mahdollisti myös koodin varmuuskopioinnin ja eri kehitysvaiheiden seuraamisen, joka vähensi riskiä menettää merkittäviä osia toteutetusta työstä. Vaikka projekti toteutettiin yksin, versionhallinta toimi hyödyllisenä apuvälineenä muutosten dokumentoinnissa ja testauksen hallinnassa.

Työskentelymalli, jota sovelsin projektin toteutuksessa, on ketterä SCRUM-malli. Jaoin työskentelyäni osiin, ja joka osion valmistuttua testasin osuuden toimivuuden ja miten hyvin se vastasi projektisuunnitelmaa. Tämä työskentelymalli mahdollisti sovelluksen jatkuvan parantamisen, ja ongelmien huomaamisen mahdollisimman aikaisessa vaiheessa (Schwaber & Sutherland, 2020).

3.1.2 Toteutusprosessi ja haasteet

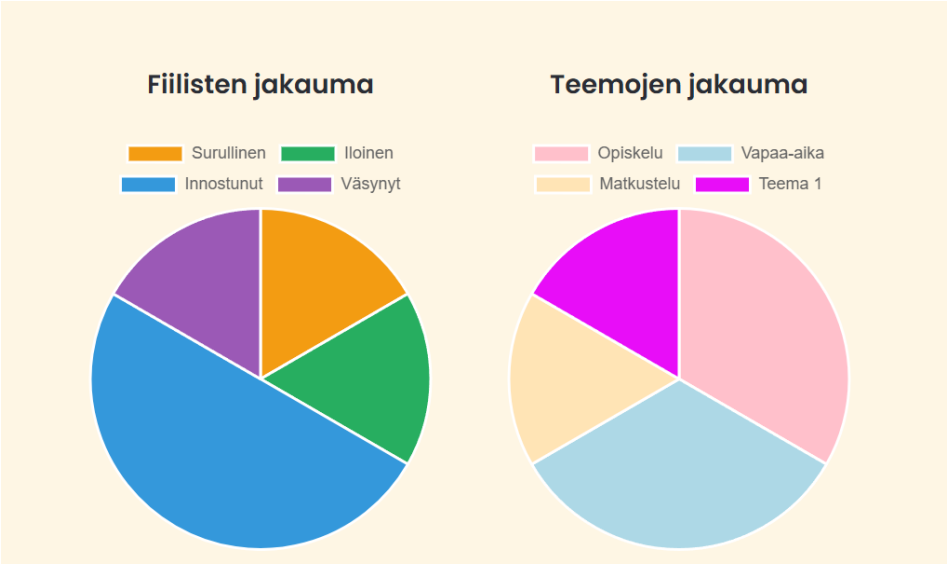
Aloitin projektin toteutuksen päätoiminnallisuuksien suunnittelulla ja komponenttien rakenteiden määrittelyllä. Ensimmäinen vaihe sisälsi kuvassa 1 esitetyn kalenterinäkymän rungon toteutuksen, joka mahdollistaa päiväkohtaiset merkinnät sekä merkintälomakkeen rakentamisen. Kuvassa 2 näkyy, että merkintälomake mahdollistaa merkintöjen lisäämisen kalenteriin. Merkintään voidaan tekstisisällön ohella lisätä merkintään sopiva teema, fiilis sekä kuva.

Kuva 1. Hetkien Arkisto-ohjelman kalenterinäkymä.



Kuva 2. Hetkein arkisto-ohjelman merkintälomake, jonka avulla käyttäjä voi lisätä merkintöjä sovelluksen kalenteriin.

Projektin toisessa vaiheessa lisäsin projektiin datan visualisoinnin, joka näkyy kuvassa 3. Se mahdollistaa merkintöihin lisättyjen fiilisten ja teemojen esittämisen piirakkakaaviossa. Kaavio hyödyntää kalenterimerkintöihin lisättyjä teema- ja fiilistunnisteita. Kaavion avulla käyttäjä voi tarkastella, miten eri teemat ja tunteet jakautuvat tietyllä ajanjaksolla.



Kuva 3. Hetkien Arkisto-ohjelman datan visualisointinäkymä, josta käyttäjä voi tarkastella tekemiensä merkintöjen fiilis- ja teema-tunnisteita ja niiden esiintymistä.

Lisäksi lisäsin teemojen hallinnan. Tämän avulla käyttäjä pystyy lisäämään uusia teematunnisteita sovellukseen. Mahdollistamalla omien teematunnisteiden lisäämisen varmistetaan, että käyttäjä pystyy kustomoimaan ohjelmaa paremmin omiin tarpeisiinsa.

Projektin kolmannessa osassa keskityin ohjelman visuaaliseen ilmeeseen ja responsiivisuuteen. Kokeilin useita erilaisia tyylejä, ja pyysin kahta ulkopuolista käyttäjää testaamaan sovelluksen intuitiivisuutta. Käyttäjäpalautteen avulla pystyin korjaamaan pieniä yksityiskohtia, joiden avulla ohjelmaa olisi helpompi käyttää. Hioin myös aiempien vaiheiden ominaisuuksia ja lisäsin koodiin kommentteja eri komponenttien toiminnasta helpottaakseni seminaarityön esittelyä.

Ratkaisu localStoragen käytöstä tietokantaintegraation sijasta syntyi lähinnä ajanhallinnallisista syistä. Ohjelman visuaalisen ilmeen luominen ja viilaaminen vei niin paljon aikaa, että opintokurssin työmäärän ja seminaarityölle annettuun aikarajaan nähden ei olisi ollut ideaalia lähteä enää toteuttamaan kirjautumista tai tietokantayhteyttä.

Projektin suunnitteluvaiheessa totesin kurssin sisällön pohjalta, että SCRUMin tyyppinen työskentelymalli on sopiva projektin toteuttamiseen. En seurannut tarkasti mitään SCRUMin ohjekirjaa, mutta toteutin kehityksen sprinteissä ja katselmoin itsenäisesti jokaisen sprintin tuloksia

korjatakseni kehityksessä ilmenneitä ongelmia ja virheitä. Ohjelmallani oli myös SCRUMille ominainen tuotteen kehitysjono, jota jalostin pilkkomalla projektin toteutusta pienempiin, tarkempiin osiin. Tämä helpotti ohjelman toteutusta.

Projektin haasteet liittyivät erityisesti Chart.js:n käytön opetteluun, sekä responsiivisuuteen. Ne olivat osa-alueita, jotka eivät olleet minulle entuudestaan tuttuja, joten suuri osa työskentelyajasta kului yritykseen ja erehdykseen. Minun oli myös jatkuvasti työskennellessäni pidettävä mielessä projektille annettu suositeltu työtuntimäärä, jotta en lähtenyt toteuttamaan mitään liian monimutkaista.

Koska kyseessä oli opintokurssiin kuuluva seminaarityö, oli saatavilla koko ajan opettajien ohjausta ja apua, mikäli ongelmia esiintyi. Itse en hyödyntänyt tätä resurssia, vaan keskityin ratkaisemaan ongelmat itsenäisesti. Jälkikäteen ajateltuna olisin voinut menetellä tässä toisin, ja uskon että ohjelmasta olisi tullut parempi ja laajempi, mikäli olisin pyytänyt ohjausta matalammalla kynnyksellä. Vaikka en hyödyntänyt kaikkia minulle tarjottuja resursseja, on projektin lopputuloksena kuitenkin toimiva sovellus, joka vastaa sille asettamiini vaatimuksiin ja suunnitelmaan hyvin. Myös kurssista saamani arvosana(4) on viite siihen, että sovellus on hyvin onnistunut ja tarkoituksenmukainen.

Jatkokehitysideoina on erityisesti tietokantayhteyden ja kirjautumisominaisuuden lisääminen. Näin käyttäjäkohtaisten tietojen tallentaminen olisi pysyvämpää ja turvallisempaa. Lisäksi jatkosuunnitelmana voisi olla sovelluksen yhteisöllistäminen; mahdollisuus jakaa omia merkintöjä muiden käyttäjien kanssa, tai luoda yhteisiä merkintöjä useamman käyttäjän kesken.

3.2 Projekti 2: Ohjelma häiden vieraslistan ja RSVP-ilmoittautumisten hallinnointiin

Tämä projekti on häiden vieraslistan hallintaan tarkoitettu sovellus, jonka toteutin Flask-kehyksen avulla. Sovellus käyttää SQLite-tietokantaa vieraiden tietojen tallentamiseen ja hallintaan. Sovelluksen avulla käyttäjä voi tehdä RSVP-ilmoittautumisen lisätietoineen sekä ladata kuvia. Lisäksi päätoiminnallisuuksiin kuuluu käyttäjien tunnistautuminen, admin-puolen hallinta, vieraslistan päivittäminen sekä tietojen lisääminen tietokantaan.

Tämä projekti on portfolion projekteista keskeneräisin. Kaikki ominaisuudet, joita ohjelmaan alun perin suunnittelin, eivät tulleet toteutettua. Projekti kuitenkin valikoitui mukaan portfolioon, koska sen pohja ja raamit ovat tarkoitusta palvelevat ja työskentely oli johdonmukaista koko projektin työstön ajan. Myös projektin perusominaisuudet tulivat toteutettua. Puutteita on suurimmaksi osaksi sivuston visuaalisessa ilmeessä ja ominaisuuksien hienosäädössä.

3.2.1 Teknologiat ja työskentelymallit

Käytin projektin toteutukseen Flaskia, joka on kevyt Pythonilla toteutettu web-kehys. Se tarjoaa yksinkertaisen tavan rakentaa palvelinpohjaisia sovelluksia. Flaskin avulla pystyin nopeasti ja helposti määrittelemään reititykset sekä käsittelemään lomakkeita.

Ohjelman tietokantayhteydessä käytin SQLitea. Se oli helppo integroida Flask-sovellukseen, ja se toimii erityisen hyvin pienissä ja keskikokoisissa projekteissa. SQLite mahdollistaa kevyen datan hallinnan ilman erillistä tietokantapalvelinta, mikä tekee sen käytöstä yksinkertaista ja tehokasta (SQLite, 2025). Python myös tarjoaa valmiin sqlite3-moduulin, jonka kanssa voi käyttää mitä tahansa SQLite-tietokantaa ilman erillisiä asennuksia (Dyori, 2021). Tämä yhteensopivuus Pythonin ja Flaskin kanssa oli tärkeä tekijä ohjelman tietokantaratkaisua päätettäessä.

Käytin ohjelman toteutuksessa myös Flaskin hyödyntämää Jinja2 templating-järjestelmää. Se mahdollistaa dynaamisten verkkosivujen generoinnin, koska se kääntää ja välimuistittaa templatet Python koodiksi. Tämä nopeuttaa koodin suorittamista (Pallets Projects, 2007). Jinja2 avulla pystyin tuomaan tietokannasta haettuja tietoja suoraan käyttöliittymään, kuten RSVP-vastausten listaukseen ja vieraslistan hallintaan.

Sovelluksen visuaalisen ilmeen määrittelyyn käytin CSS-tyylitiedostokieltä. CSS ohjaa, miten ohjelman eri erin eri elementit näytetään eri medioissa. Pyrin käyttöliittymän toteutuksessa mahdollisimman responsiiviseen ja käyttäjäystävälliseen lopputulokseen. Hyödynsin CSS-tyylejä selkeyden ja esteettisyyden parantamiseen, sekä painikkeiden, navigaation ja lomakkeiden muotoilun käytettävyyteen ja saavutettavuuteen.

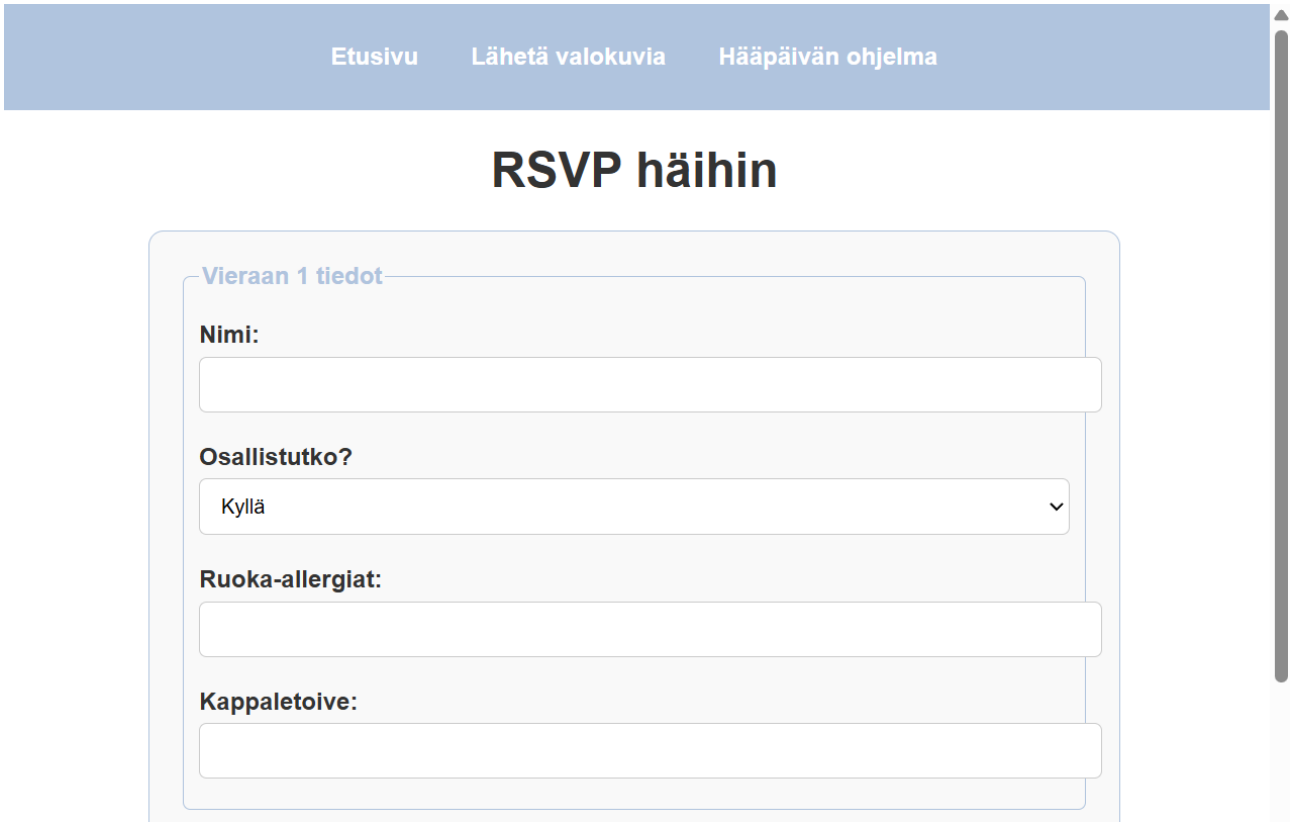
Lisäksi hioin sivujen ja lomakekenttien dynaamisuutta JavaScriptia hyödyntämällä. JavaScript on dynaaminen ohjelmointikieli, jonka avulla voidaan lisätä verkkosivujen vuorovaikutteisuutta (MDN, 2025). Paransin sen avulla ohjelman käyttäjäkokemusta esimerkiksi siten, että RSVP-lomakkeen ruoka-allergiakentät ja kappaletoiveiden syöttö näytetään käyttäjälle vain, mikäli hän on aiemmin lomakkeella jo valinnut RSVP-vastauksekseen osallistuvansa juhliin.

Projektin kehityksen työskentelymallina hyödynsin pääasiallisesti vesiputousmallin kaavaa, joskin en täysin puhtaasti. Koin vesiputousmallin eduksi sen selkeän rakenteen ja vaiheiden järjestelmällisyyden. Projektin vaatimukset olivat hyvin selkeät alusta saakka, ja oli hyvin epätodennäköistä, että ne muuttuisivat projektin aikana. Tiedossa oli, että vierailta tarvittiin tieto, osallistuvatko he tapahtumaan, sekä osallistuvilta vierailta tietyt lisätiedot.

3.2.2 Toteutusprosessi ja haasteet

Projektin tavoitteiden toteamisen ja tarkentamisen jälkeen projektin toteutus eteni vaiheittain. Ensimmäisessä vaiheessa suunnittelin ja alustin tietokannan. Tietokannan rakenteen määrittelyn jälkeen lisäsin mukaan alustavat vierastiedot. Tietokantaratkaisuksi valikoitui SQLite, ja sen luomiseen sekä täyttämiseen hyödynsin init.db.py-skriptiä.

Seuraavaksi toteutin Flask-sovelluksen perustoiminnallisuudet. Tässä vaiheessa rakensin sovelluksen sivupohjat ja reitit ykset. Lisäksi kehitin dynaamisen RSVP-lomakkeen, jonka avulla käyttäjät voivat ilmoittautua tapahtumaan sekä ilmoittaa mahdollisista ruoka-allergioista ja esittää musiikkitoiveita lisättäväksi juhlien soittolistalle (kuva 4). Lopuksi lisäsin sovellukseen valokuvien latausominaisuuden. Tämä mahdollistaa käyttäjien lataamien kuvien tallentamisen palvelimelle, ja tarjoaa osallistujille mahdollisuuden jakaa hääkuvia keskenään.



Kuva 4. Hääsivuston RSVP lomake, jonka täyttämällä ja lähettämällä kutsuvieras voi ilmoittaa kutsujalle pääsevänsä paikalle, sekä kutsujan tarvitsemat lisätiedot.

Näiden ominaisuuksien jälkeen oli tarkoitus vielä toteuttaa käyttäjätunnistus, jotta vain valtuutetut käyttäjät voisivat hallinnoida vieraslistaa. Tällä oli tarkoitus lisätä sovelluksen turvallisuutta. Lisäksi oli tarkoitus lisätä hääpäivän ohjelma, aikataulu ja istumajärjestys niille tarkoitetuille sivuille. Tämän tavoitteena oli saada kaikki vieraiden tarvitsema tieto yhteen lähteeseen.

Projektin kehittämisen aikana kohtasin kuitenkin haasteita erityisesti sisäänkirjautumisominaisuuden ja admin-oikeuksien kanssa. Vasta kehityksen ollessa hyvin pitkällä huomasin, että Flask ei välttämättä ollutkaan paras vaihtoehto ohjelmalle, jossa tarvitaan erillinen admin-interface. Esimerkiksi Django olisi tarjonnut tähän valmiimman pohjan.

Projektin lopputulos on tarkoituksenmukainen, muttei täysin toimiva tai valmis sovellus. Sovelluksesta jäi puuttuman haluttuja ominaisuuksia jotka olisivat tehneet siitä käyttökelpoisemman. Pohja on kuitenkin toimiva ja kehityskelpoinen, eikä siinä ole isompia virheitä, jotka estäisivät tai edes hankaloittaisivat jatkokehitystä. Vaikka esimerkiksi aiemmin viittasin Djangon valmiiseen admin-rajapintaan, myös Flask tarjoaa tähän vaihtoehtoja, joten ei ole tarpeen aloittaa projektia alusta. Pohjatyön käytettävien teknologioiden suhteen olisi kuitenkin voinut ehdottomasti tehdä paremmin, jolloin tällaiselta vaihtoehtoisten ratkaisujen mietinnältä olisi säästytty ja kehitys olisi todennäköisesti edennyt pidemmälle.

Projektin parissa työskennellessäni opin kuitenkin paljon seuraavista asioista. Opin hyödyntämään Flask-kehystä web-sovelluksen rakentamisessa, erityisesti reititysten hallinnassa, tietokantakyselyiden toteutuksessa sekä templating-järjestelmän käytössä. Lisäksi sain kokemusta SQLite-tietokannan käytöstä, erityisesti tietokantataulujen välisten suhteiden hallinnasta ja tietojen tehokkaasta hakemisesta. Opin myös hyödyntämään JavaScriptia lomakkeiden parantamisessa, esimerkiksi RSVP-lomakkeen dynaamisessa muokkaamisessa käyttäjän tekeminen valintojen perusteella.

Sovellusta voisi edelleen kehittää useilla tavoilla. Kehitysideat veisivät projektia eteenpäin ja tekisivät siitä tehokkaamman ja paremmin tarkoitustaan palvelevan. Yksi merkittävä kehitysaskel olisi tietokantaratkaisun päivittäminen PostgreSQL.ään, joka tarjoaisi paremman suorituskyvyn ja mahdollistaisi laajennettavuuden sekä suuremman tehokkuuden (Airbyte, 2024). Lisäksi sovellukseen voisi lisätä sisäänkirjautumisominaisuuden, jotta adminilla olisi mahdollisuus hallita tiettyjä sovelluksen osia, kuten vierastietokantaa, suoraan käyttöliittymän kautta.

RSVP-lomaketta voisi myös kehittää. Käyttäjälle voisi esimerkiksi tarjota mahdollisuuden muokata aiemmin antamiaan tietoja tiettyyn aikarajaan saakka ennen tapahtumaa. Tämä ominaisuus olisi yhteydessä sisäänkirjautumisjärjestelmään, sillä eri tasoisilla käyttäjillä tulisi olla erilaiset käyttöoikeudet.

3.3 Projekti 3: Verkkosivu tatuointialan yritykselle

Tämän projektin toteutin opintoihini kuuluvana työharjoitteluna. Kyseessä on verkkosivusto tatuointialan yrittäjälle. Tavoitteena oli luoda verkkosivu, jonka kautta yrittäjän asiakkaat voivat lähettää ajanvaraustiedusteluja, tarkastella ja varata itselleen tatuoijan luomia valmiita tatuointimalleja, sekä

tarkastella tatuoijan portfoliota. Tavoitteena oli luoda sivu, joka toimisi sekä yhteydenottoalustana että täydennyksenä yrittäjän portfoliolle etenkin asiakkaille, jotka eivät käytä sosiaalista mediaa.

Projekti valikoitui mukaan tähän portfolioon, koska se on opiskeluaikaisista projekteistani pisimmälle ja huolellisimmin kehitetty. Lisäksi halusin portfolioon mukaan projektin, joka on toteutettu täysin itsenäisesti mutta asetetussa aikataulussa ja osana opintoja. Lisäksi tämä projekti tuo hyvää lisää työskentelymallien analysointiin, sillä se on toteutettu yhteistyössä asiakkaan kanssa, jolloin työskentely oli rakenteellisempaa ja seurasi tarkemmin tiettyjä ohjelmistokehityksen työskentelymalleja.

3.3.1 Teknologiat ja työskentelymallit

Käytin tässä projektissa useita teknologioita, joiden parissa olin työskennellyt jo aiemmin. Tämän projektin parissa tutustuin kuitenkin tarkemmin SQLAlchemy ORMin kanssa työskentelyyn. Se mahdollista tietokannan käsittelyn Python-olioilla ilman suoria SQL-kyselyitä. SQLAlchemy valikoitui mukaan projektiin, koska se on yhteensopiva Flaskin kanssa ja tarjoaa lisää joustavuutta ja hallittavuutta tietokantojen käsittelyyn. Muita tässä projektissa käytettyjä teknologioita ovat jo aiemmin esitellyt:

- Flask(python)
- SQLite
- HTML & Jinja2
- CSS
- JavaScript

Tässä projektissa hyödynsin kaikkein selvimmin ketterää kehitysmallia. Minulla oli projektille kirjallinen, pienempiin osiin jaettu suunnitelma ja aikataulu. Toteutin projektin osa kerrallaan, testaten jokaisen pienemmän kokonaisuuden toimivuuden ennen seuraaviin siirtymistä. Täten pystyn antamaan itselleni jatkuvasti palautetta, kehittämään ja muuttamaan suunnitelmia ja aikatauluja tarpeen mukaan, ja pitämään asiakasta mukana kehityksessä säännöllisillä tapaamisilla.

Asiakkaalta tuli joitain kertoja projektin aikana jatkokehitysideoita sekä lisäyksiä ja muutoksia alkuperäiseen tarvekartoitukseen. Projektin alkuaikana tapasimme noin kahden viikon välein, jotta pystyin pitämään yllä selkeää työsuunnitelmaa siitä huolimatta, että asiakkaan mieli ja tarpeet muuttuivat kehityksen edetessä. Projektin edetessä pidemmälle, suunnittelusta kehitysvaiheeseen, asiakastapaamiset harvenivat. Muun koulutyön ohessa kehitystyö vei aikaa, emmekä asiakkaan kanssa kokeneet tarpeelliseksi palaveerata jokaisen painikkeen koodaamisesta. Asiakas antoi minulle kehittäjänä paljon luovia vapauksia sivuston toteuttamisessa, joten pystyin toteuttamaan SCRUM tyylistä kehitystä omassa yhden hengen tiimissäni.

3.3.2 Toteutusprosessi ja haasteet

Projektin kehityksen ensimmäisessä vaiheessa pidimme asiakkaan kanssa aloituspalaverin. Tässä palaverissa kävimme läpi, millaisia alustoja hänellä on entuudestaan käytössä, ja millaisia tarpeita hänellä on nettisivuille näiden alustojen ulkopuolelle. Tarkastelimme muiden saman alan yrittäjien sivustoja, joista hän löysi itseään miellyttäviä ominaisuuksia ja piirteitä, joko toiminnallisuuden tai visuaalisuuden kannalta. Kävimme läpi sisältöä, jota hän sivuilleen toivoo, sekä visuaalisesta ilmeestä ne osat, joista hänellä oli vahva näkemys (väripaletti, fontit). Rajasimme, mitä on mahdollista toteuttaa tietyssä aikaikkunassa oman taitotasoni huomioiden. Suunnittelimme myös projektin ja tulevien tapaamisten aikataulua. Tässä vaiheessa kävimme myös läpi mahdollisuuden käyttää projektia opinnäytetyössäni.

Kehityksen toinen vaihe koostui teknologioihin tutustumisesta. Olin käyttänyt suurinta osaa teknologioista jo aiemmin, joka olikin suurin syy siihen, että ne valikoituivat osaksi tätä projektia. Halusin kuitenkin syventää osaamistani näistä teknologioista, joten tutkin teknologioiden dokumentaatioita, sekä ohjelmistokehittäjien forumeita, joista löysin hyviä esimerkkejä käyttötapauksista, joita pystyin myöhemmin hyödyntämään omassa työssäni.

Aloitin projektin toteutuksen luomalla Flask-sovelluksen pohjaksi nettisivulle. Loin ohjelmaan sivut ominaisuuksille, joita halusin sinne lisätä sekä navigaation sivujen välille. Ensimmäinen ominaisuus, jonka kehitin, oli ajanvarauslomake (Kuva 5). Lomake on hyvin kattava, sillä tatuoija tarvitsee asiakkaalta paljon tietoja arvioidakseen sopivan ajan tatuoinnin toteuttamiselle. Lomake lähettää tiedot tatuoijan sähköpostiin, ja sähköpostiviestin perusteella tatuoija pystyy ottamaan asiakkaaseen yhteyttä ja sopimaan budjetista, konsultaatiosta ja varattavasta ajasta.

Kuva 5. Yhteydenottolomake tatuointialan yrittäjän verkkosivulta.

Hyödynsin ajanvarauslomaketta myös sivuston seuraavan osan toteutuksessa. Sivulla on galleria tatuoijan suunnittelemia valmiita mallikuvia, joita asiakkaat voivat tarkastella, mikäli heillä ei ole omaa tarkkaa ideaa, millaisen tatuoinnin he haluavat. Koska tatuoijalla on periaate, että hän tatuoi jokaisen mallin vain kerran, pystyy valmiita malleja varaamaan galleriasta. Asiakas voi valita mallin, ja varauspainiketta painamalla siirtyä ajanvarauslomakkeelle. Ohjelma siirtää galleriasta valitun kuvan suoraan ajanvarauslomakkeen ’inspiraatiokuva’ kohtaan. Tämän jälkeen lomakkeen lähettäminen poistaa vapaan mallin galleriasta, jotta usea ihminen ei voi varata samaa mallia.

Seuraavaksi lähdin kehittämään sivuston portfolio-osuutta. Tavoitteena oli luoda kuvagalleria, jossa kuvien asettelu muuttuu responsiivisesti selailuun käytettävän laiteen näytön koon mukaan. Galleriassa näkyvä kuva on ’kansikuva’ kansiolle, joka aukeaa näytölle suurempana, kun kuva valitaan galleriasta. Näin yhdestä tatuoinnista koottua kuvakansiota pystyy selaamaan suurempina kuvina, joista näkyvät tarkemmin tatuoinnin yksityiskohdat. Käytin tämän ominaisuuden toteuttamiseen model dialog-ominaisuutta, jolla kuvakansio aukesi auki olevan sivun päälle, eikä ollut tarvetta navigoida sivujen välillä kuvia tarkastellakseen.

Nettisivun etusivulle tavoitteena oli lisätä kuvakaruselli, joka haki yrittäjän instagramista viimeisimmät julkaisut, ja toimisi samalla linkkinä, jonka kautta sivulla vierailleva asiakas voisi siirtyä

Instagramiin. Tarkoituksena oli käyttää Metan API rajapintaa yhdistämään Instagram sivustolle, mutta kävi ilmi että sen käyttöönotto oli monimutkaisempaa kuin projektille asetettu aikaikkuna ja kehittäjän taitotaso sallivat. Tarkastelin vaihtoehtoisia toteutustapoja, mutta löytämissäni vaihtoehdoissa oli riskinä, että Meta arvioisi ne haitalliseksi verkkoharavoinniksi, ja yrittäjälle tulisi ongelmia tilinsä käyttöoikeuksien kanssa. Järjestimme tapaamisen asian tiimoilta asiakkaan kanssa, sillä tämä ominaisuus oli alkujaan hänelle tärkeä sivun toteutuksessa. Käytyämme läpi realiteetit Metan APIn suhteen, haasteet ja mahdolliset riskit vaihtoehtoisten ratkaisujen käytössä, päädyimme jättämään ominaisuuden pois. Se on kuitenkin yhä jatkokehityssuunnitelmissa, ja sivulle lisättävien ominaisuuksien listalla, mutta sen tärkeysastetta madallettiin.

Kun sain sivuston perusominaisuudet toimiviksi, siirryin kehittämään sivuston visuaalista ilmettä. Olimme asiakkaan kanssa sopineet käytettävästä väripaletista sekä fonteista. Hän toimitti minulle kuvia käytettäväksi esimerkiksi sivuston headerissa(Kuva 6). Lisäksi hyödynsin CSS:ää pienissä yksityiskohdissa, kuten hover-efekteissä sivuston painikkeiden yhteydessä. Näiden efektien ulkopuolella sivuston tyyli on hyvin yksinkertainen. Halusimme että sivun tyyli on selkeä ja helposti navigoitavissa. Tästä huolimatta käyttöliittymän toteutus vei suuren osan kehitysprosessiin käytetystä ajasta.

Kuva 6. Valmiiksi muotoiltu ja reititetty header tatuointialan yrittäjän verkkosivuilta, johon on lisätty mukaan yrittäjän taidetta.

Suurimmat haasteet projektin yhteydessä olivat aikataulutus sekä käyttöliittymän visuaalisen ilmeen suunnittelu. Kirjallisesta suunnitelmasta huolimatta projektin edetessä oli haastavaa löytää ajankohtia palavereille asiakkaan kanssa. Lisäksi muun opiskelutyön ohella kehitystyölle ajan löytämine oli ajoittain vaikeaa, etenkin kun kohtasin ohjelmoinnissa vaikeuksia, joiden ratkaisu vaati lisää pohjatyötä ja paneutumista teknologioiden dokumentaatioihin.

Sivuston visuaalisen ilmeen suunnittelu oli haasteellista, koska en ole opinnoissani syventynyt siihen erityisemmin. Tämä oli ensimmäinen suurempi projekti, johon minun täytyi suunnitella ja toteuttaa yhtenäinen, asiakkaan toiveiden mukainen ilme. Kokemattomuuteni vuoksi oli vaikea hahmottaa, mikä on mahdollista toteuttaa. Lisäksi tekemiseni suunnittelun osalta perustui puhtaasti omiin kokemuksiini erilaisten verkkosivujen käyttäjänä. Oli siis haastavaa arvioida, olivatko ratkaisuni sivuston asettelusta vain omia kokemuksiani vai oikeasti intuitiivisesti ja selkeästi käytettävissä.

4 Projektien vertailu ja johtopäätökset

Tähän opinnäytetyöhön sisällytetyt projektit tarjoavat hyvän katsauksen erilaisiin ohjelmistokehityksen teknologioihin ja työskentelymalleihin. Vaikka jokaisella projektilla oli omat tavoitteensa ja haasteensa, niitä yhdistävät ohjelmistokehityksessä yleisesti hyödynnetyt teknologiset ja metodologiset pohjat. Lisäksi jokainen projekti toteutettiin erilaisista lähtökohdista yhden projektin ollessa osa opintokurssia, toisen oma, henkilökohtainen projekti ja kolmannen asiakastyönä toteutettava projekti. Näiden lähtökohtien vaikutukset näkyvät myös selkeästi projektien työskentelyn etenemisessä ja lopputuloksissa.

Näitä projekteja ja niihin liittyviä tekijöitä tarkastelemalla voidaan analysoida teknologioiden ja työskentelymallien vaikutuksia ohjelmistokehitysprosessiin ja omaan työskentelyyni, sekä arvioida, mitkä ratkaisut osoittautuivat tehokkaimmiksi ja mitkä kaipaisivat kehitystä tulevaisuudessa.

4.1 Teknologioiden yhtäläisyydet ja erot

Ensimmäinen projekti, Hetkien Arkisto-ohjelma, jonka toteutin seminaarityönä opintokurssille, hyödynsi pääasiassa TypeScriptiä ja Reactia. Keskeiseksi haasteeksi nousi erityisesti datan visualisointi Chart.js:n avulla sekä responsiivisen ja visuaalisesti miellyttävän käyttöliittymän suunnittelu ja toteutus. Projekti onnistui kuitenkin hyvin, ja sen lopputulos oli toimiva, helposti laajennettavissa oleva sovellus.

TypeScriptin käyttö tarjosi paremman koodin ylläpidettävyyden, mutta samalla lisäsi alkuvaiheen kehitykseen kulunutta aikaa sillä se oli minulle kehittäjänä täysin uusi teknologia. Teknologiavalinta tuli kuitenkin opintokurssin puolelta, sillä seminaarityön ideana oli nimenomaan tutustua tarkemmin johonkin kurssilla käsiteltyyn aiheeseen tai teknologiaan, ja toteuttaa jonkinlainen esiteltävä työ valitulla teknologialla. Opintokurssin aikataulutus ja vaatimukset asettivatkin projektille tiukat raamit, joiden sisällä toimia, ja tämän vuoksi osa alkuperäisistä tavoitteista jäi saavuttamatta. Projektin toteutukseen oli kuitenkin saatavilla ohjausta koko sen toteutuksen ajan, ja kurssin ohessa saatu vertaistuki muilta opiskelijoilta, sekä ohjaus opettajilta ohjasivat projektin toteutumista ja muotoutumista sen lopulliseen tulokseen.

Toinen projekti, vapaa-ajallani itsenäisesti toteutettu häiden vieraslistan hallintaan tarkoitettu sovellus, perustui Flaskiin ja SQLite-tietokantaan. Toteutuksen edetessä huomasin, että Flask ei välttämättä ollut paras mahdollinen kehys sisäänkirjautumis- ja admin-toimintojen toteuttamiseen. Tämä rajoite nousi esiin erityisesti siinä vaiheessa, kun aloin perehtyä käyttöoikeuksien hallintaan.

SQLite tarjosi kevyen ratkaisun tietokannan hallintaan, mutta se ei mahdollistanut skaalautuvuuden kannalta parhaita käytäntöjä, mikä saattaa muodostua haasteeksi, jos päädyn vielä

jatkokehittämään sovellusta. Projektin lopputulos on käyttökelpoinen, mutta se jäi osittain keskeneräiseksi, ja sen jatkokehityksessä tulen harkitsemaan esimerkiksi PostgreSQL-tietokantaa sekä mahdollisesti Djangon tarjoamia hallintatyökaluja.

Ensimmäiseen, opintokurssille toteutettuun sovellukseen verrattuna, tämä sivusto syntyi huomattavasti nopeammin. Flaskin ja admin-ominaisuuksien yhteensopivuusongelmista huolimatta kehitystyö oli suoraviivaisempaa, koska ominaisuudet, joita sivustolle kehitin, olivat minulle entuudestaan tuttuja, vaikka en suoranaisesti Flaskia ollut käyttänyt aiemmin. Tietokantayhteydet ja erilaiset lomakkeet ovat tulleet minulle hyvin tutuiksi opintojeni kautta, joten niiden toteuttaminen ei luonut haasteita. Lisäksi Python on minulle ohjelmointikielenä tuttu.

Kolmannen projektin, Verkkosivuston tatuointialan yrittäjälle, toteutin Flaskin, SQLAlchemyn, HTML:n, CSS:n ja JavaScriptin avulla. Haasteita tuotti erityisesti aikataulutus sekä API-rajapinnan hyödyntäminen Instagram-sisällön lataamisessa, jonka lopulta jouduin jättämään pois projektin tässä vaiheessa. Käytettävyystestit ja asiakaspalautteen perusteella tehdyt muutokset paransivat verkkosivuston lopputulosta merkittävästi.

Suurimmaksi osaksi kolmannessa projektissani käytetyt teknologiat olivat minulle entuudestaan tuttuja. Se onkin teknologiavalinnoiltaan hyvin samankaltainen kuin portfolion toinen projekti. Projekti on kuitenkin yksityiskohtaisempi ja teknologioita ja niiden ominaisuuksia hyödynnettiin huomattavasti paremmin ja tehokkaammin.

Projektien teknologiat eroavat toisistaan, mutta myös jakavat samoja perusperiaatteita. Esimerkiksi TypeScriptin käyttö Hetkien Arkisto -projektissa tarjosi vahvan tuen rakenteelliselle ja virheettömälle frontend-kehitykselle (TypeScript Team, 2025.). Staattinen tyypitys auttoi tunnistamaan virheitä jo kehitysvaiheessa, mikä vaikutti sovelluksen vakauteen ja käytettävyyteen. Tämä ei olisi ollut mahdollista yhtä tehokkaasti pelkällä JavaScriptillä, joka ei tarjoa samaa tarkkuutta virheidenhallintaan.

Toisaalta Flaskin käyttö kahdessa projektissa – RSVP-sovelluksessa ja tatuointialan yrityksen verkkosivustossa – osoitti sen joustavuuden ja nopeuden pienten sovellusten kehittämisessä (Pallets Projects 2010). Flask tarjosi selkeän reititysmallin ja mahdollisuuden yhdistää backend ja frontend loogisesti. Tatuointiyrittäjän portfoliosivustossa SQLAlchemyn ja Flaskin yhdistelmä tehosti tietokannan käsittelyä, mikä osoittaa Pythonin ja sen ekosysteemin vahvuudet (Python Software Foundation 2001).

SQLite toimi nopeasti ja oli helppo integroida molemmissa Flask-projekteissa, mutta sen rajoitteet nousivat esiin jatkokehitystarpeiden kohdalla. Tietoperustan perusteella PostgreSQL olisi tarjonnut

skaalautuvamman vaihtoehdon, ja sen käyttöönotto voisi olla looginen seuraava askel RSVP-sovelluksen jatkokehityksessä.

4.2 Työskentelymallien ja -tapojen eroavaisuudet

Ensimmäisen projektin, Hetkien Arkisto-ohjelman toteuttaminen perustui ketterään kehitykseen, jossa kehitin ja testasin sovelluksen ominaisuuksia vaiheittain. Työskentelymallin seuranta oli kuitenkin melko löyhää, ja mitä pidemmälle projekti eteni, sitä vapaamuotoisemmaksi työskentelyjärjestys muodostui.

Häiden vieraslistan hallintasovelluksessa sovelsin työskentelymallina pääasiassa vesiputousmallia, koska pystyin asettamaan projektille hyvin selvät vaatimukset alusta asti, eikä merkittäviä muutoksia ollut odotettavissa prosessin aikana.

Kolmas projekti, verkkosivusto tatuointialan yrittäjälle, oli selkein esimerkki ketterästä kehityksestä, sillä asiakkaan tarpeet ja toiveet elivät projektin edetessä. Työskentelymallina hyödynsin Scrum-tyyppistä toteutusta, jossa jaoin tehtävät pienempiin osiin ja hyödynsin aktiivisesti tapaamissa kertynyttä asiakaspalautetta. Kehityksen aikana huomasin, että asiakkaan muuttuvat tarpeet vaativat jatkuvaa uudelleenjärjestelyä ja joustavuutta, jonka vuoksi ketterä kehitysmalli oli tässä tapauksessa toimivin ratkaisu.

Työskentelymallien valinnat perustuivat pitkälti projektien luonteeseen. Hetkien Arkisto -projektissa hyödynnetty ketterä, Scrum-tyylinen kehitys tuki jatkuvaa iterointia ja reflektointia (Agile Alliance s.a.). Sprinteissä edennyt kehitys mahdollisti nopean testauksen ja virheiden korjauksen, mikä oli tärkeää erityisesti frontend-komponenttien ja visuaalisuuden parantamisessa. Ketteryyden korostama jatkuva palaute näkyi myös asiakastyössä toteutetussa nettisivussa.

Sen sijaan häitä varten kehitetty RSVP-sivusto toteutettiin pääosin vesiputousmallin mukaisesti, koska vaatimukset olivat tiedossa jo alkuvaiheessa ja pysyivät muuttumattomina (Kirvan, Lutkevich & Lewis 2024.). Tämä mahdollisti järjestelmällisen etenemisen, mutta samalla paljasti mallin jäykkyyden, kun kehityksen aikana nousi esiin uusia ominaisuustarpeita, kuten admin-rajapinta, joita ei ollut aluksi suunniteltu.

Työskentelymallien osalta voidaan huomata, että joustavammat mallit, kuten ketterä kehitys ja sen muunnelmat, ovat tehokkaampia silloin, kun projektin vaatimukset voivat muuttua matkan varrella. Tämä oli selvästi havaittavissa tatuointiyrityksen verkkosivustoprojektissa, jossa asiakaspalautteen pohjalta tehtiin lukuisia suunnanmuutoksia ja tarkennuksia.

4.3 Oma oppiminen ja kehitys prosessin aikana

Tietoperustan teknologioiden ja työskentelymallien tuntemukseni syveni merkittävästi projektien myötä. Esimerkiksi TypeScriptin käyttöönotto Hetkien Arkisto -projektissa tarjosi käytännön ymmärrystä siitä, miten staattinen tyypitys vaikuttaa projektin ylläpidettävyyteen ja virheiden ehkäisyyn (TypeScript Team, 2025.). TypeScriptin opettelu oli haastavaa, mutta sen tarjoamat työkalut, kuten automaattinen täydentäminen ja virhetarkastus, osoittautuivat arvokkaiksi etenkin datan visualisoinnin toteutuksessa. TypeScriptin opiskelu projektia varten myös paransi valmiuksiani kehittää laadukkaita frotend-ratkaisuja.

Flaskin käyttö antoi minulle arvokasta kokemusta web-kehityksen perusteista backendin näkökulmasta. Ymmärsin konkreettisesti, miten reititykset, tietokantayhteydet ja templating-järjestelmät muodostavat web-sovelluksen ytimen. Samaan aikaan hahmotin myös Flaskin rajoitteet ja sen, milloin olisi hyödyllistä valita valmiimpi kehys kuten Django – erityisesti admin-paneelin kaltaisten ominaisuuksien tarpeessa.

Työskentelytapana ketterän kehityksen mallit tuntuivat minulle luonnolliselta valinnalta, koska niitä on painotettu paljon opinnoissani. Tämä tuntui myös selvimmältä valinnalta työharjoitteluprojektiin, koska osasin aavistaa, että asiakkaan kanssa työskennellessä ohjelmiston vaatimukset ja tarpeet elävät toteutuksen aikana, joten työskentelymallin täytyi joustaa tämä huomioon ottaen.

Etenkin kolmannen projektin yhteydessä opin paljon projektien aikatauluttamisen haasteista. Lisäksi opin kuuntelemaan asiakkaalta saatua palautetta ja toivomuksia muutoksista, sekä konsultoimaan asiakasta tavalla, joka helpottaa tarpeiden tärkeysjärjestyksen kehittämistä. Lisäksi opin kantapään kautta, miten tärkeä on tutustua mahdollisiin teknologioihin ennen niiden valitsemista projektin toteutukseen, jotta voi myöhemmin välttyä erilaisilta yhteensopivuusongelmilta. Portfolion projekteja työstäessäni opinkin soveltamaan eri työskentelymalleja tilanteen mukaan. Tietoperustassa esitellyt mallit eivät siis jääneet teoreettisiksi, vaan toimivat käytännön ohjenuorana projektien toteutuksessa ja päätöksenteossa.

4.4 Vastuullisuudesta

Vastuullisuus ohjelmistokehityksessä voi oppimani mukaan näyttäytyä monella tavalla, eikä se aina tarkoita monimutkaisia tietoturvaratkaisuja tai täydellisesti dokumentoituja järjestelmiä. Omien projektieni kohdalla vastuullisuus on näkynyt ennen kaikkea käytännön valinnoissa: siinä, miten suunnittelin ohjelmistot palvelemaan käyttäjiään, sekä siinä, miten rehellisesti arvioin omia rajojani ja oppimiskohteitani. Esimerkiksi ohjelmistojen saavutettavuus ja käytettävyys nousivat toistuvasti esiin valintatilanteissa. Pyrin tekemään ohjelmista mahdollisimman selkeitä, visuaalisesti ymmärrettäviä ja loogisia käyttää. Halusin, että ne vastaisivat hyvin niille asetettuja vaatimuksia, mutta

myös tukisivat omaa oppimistani ja ammattitaitoni kehittymistä ja syventymistä. Koen, että nämä tekijät ovat osa kehittäjän vastuuta, vaikka kyse tässä tapauksessa onkin vain pienistä tai henkilökohtaisista projekteista.

Myös projektien rajaus ja teknologisten valintojen tekeminen suhteessa omiin taitoihin ja käytettävissä olevaan aikaan olivat vastuullisia päätöksiä. En esimerkiksi lähtenyt toteuttamaan kirjautumistoimintoja tai tietoturvaan liittyviä ominaisuuksia tilanteissa, joissa en olisi pystynyt varmistamaan niiden toimivuutta tai turvallisuutta riittävällä varmuudella. Huomattuani, että asetetuissa aikarajoissa ja valitsemillani teknologioilla minun ei olisi mahdollista kehittää niitä sille tasolle, kuin olisi tarvinnut, asetin ne tulevaisuuden kehitysjonoihin, odottamaan että pääsen palaamaan projektien pariin. Tällaisissa kohdissa vastuullisuus tarkoitti rajaamista ja sen hyväksymistä, että kaikkea ei tarvitse tehdä kerralla tai yksin.

Kehitysprojektien aikana pyrin myös välttämään tekoälysisällön tai avustuksen käyttöä. Olen tietoinen erilaisten generatiivisten tekoälyjen aiheuttamista ilmasto- ja ympäristöhaitoista (Zewe, 2025), sekä tiedostan, miten haitallista esimerkiksi valmiin koodin tuottaminen ja käyttäminen voisi olla omalle oppimiselleni. Lisäksi se olisi harhaanjohtavaa sekä opintojeni etenemisen, että asiakkaalle tarjoamani tulosten suhteen.

Lisäksi projektien aikana pyysin palautetta käyttäjiltä ja asiakkaalta, ja tein muutoksia palautteen pohjalta. Vastuullisuus näkyy tässä vuorovaikutuksessa ja halussa kehittää ohjelmistoja oikeita tarpeita varten – ei ainoastaan teknologian oppimisen ilosta. Ohjelmistokehittäjänä näen vastuullisuuden osana jatkuvaa oppimista ja realistista arviointia: mikä toimii, mikä ei, ja miten voin kehittää työtäni niin, että siitä on mahdollisimman paljon hyötyä muille.

5 Yhteenveto

Tässä opinnäytetyössä tarkasteltiin kolmea erilaista ohjelmistoprojektia, jotka toteutettiin eri konteksteissa ja erilaisilla teknologioilla. Projekteja yhdisti tarve valita tarkoituksenmukaiset ohjelmointikielet, työkalut ja kehitysmallit. Valinnat eivät perustuneet pelkästään tekniseen tehokkuuteen, vaan myös omiin oppimistavoitteisiin ja projektien reunaehtoihin, kuten aikarajoihin ja asiakasvaatimuksiin. Tietoperustassa esitellyt teknologiat osoittautuivat käyttökelpoisiksi, mutta niiden rajoitteet nousivat esiin erityisesti projektien jatkokehitysideoita pohdittaessa.

Projektien vertailusta nouseekin esiin muutama keskeinen johtopäätös ohjelmistokehityksen työtavoista ja teknologioista. Ketterä kehitys soveltuu erinomaisesti projekteihin, joissa vaatimukset voivat muuttua tai joihin liittyy asiakasyhteistyötä. Scrum-tyylinen kehitysprosessi mahdollisti verkkosivuprojektissani jatkuvan parantamisen, kun taas vesiputousmalli sopi häiden vieraslistaprojektiin, jossa päätavoitteet olivat hyvin selkeät alusta asti. Tämä vastaa tietoperustan kuvausta mallien vahvuuksista ja haasteista. Tulevaisuudessa voi olla hyödyllistä arvioida, missä määrin voin yhdistää näitä malleja joustavamman kehitysprosessin saavuttamiseksi.

Työskentelymallien valintoja reflektoidessa ei voida kuitenkaan sivuuttaa sitä faktaa, että projektien erilaiset lähtökohdat heijastuivat niissä selkeästi. Opintojeni ohessa toteutettujen projektien työstäminen ketterien kehitysmenetelmien tahdissa tuntui luontaisimmalta valinnalta, koska opinnoissani on käsitelty paljon sen etuja ja toteutusta. Näissä projekteissa oli myös tarkemmat vaatimukset sille, mitä kaikkea ohjelmien piti sisältää. Tämä helpotti työn jakamista pienempiin osiin, eikä projektien toteutus todennäköisesti olisi edes onnistunut vesiputousmallin kaltaisella työjärjestyksellä.

Täysin omatoimisesti ja omiin tarpeisiin suunniteltu ja toteutettu hääsivusto sen sijaan oli vaatimuksiltaan suoraviivaisempi ja yksinkertaisempi. Vapaa-ajan projektina sen toteutuksessa ei tarvinnut ottaa huomioon tiettyihin arvosanatavoitteisiin liittyviä vaatimuksia tai tarkkoja aikatauluja. Toisaalta, koska projektille ei ollut ulkopuolelta asetettuja vaatimuksia, jäi sen sisältö ehkä hieman yksinkertaiseksi. Suunnitteluosuus ei ollut lainkaan niin syväluotaava ja yksityiskohtainen kuin kahdessa muussa projektissa, ja mikäli tarkastelen projektia jälkikäteen, on todennäköistä, että vaatimusmäärittely tulee muuttumaan jonkin verran.

Teknologian valinnalla on suuri merkitys projektin onnistumisen kannalta. Pythonin ja Flaskin käyttö toi esiin backend-kehityksen keskeiset elementit, mutta myös sen, että kevyt kehys ei aina riitä laajempien hallintaominaisuuksien toteutukseen. TypeScript ja React tukivat hyvin frontend-kehitystä, mutta vaativat syvällisempää perehtymistä ja suunnittelua, erityisesti käyttöliittymän responsiivisuuden ja saavutettavuuden kannalta. Tietokantaratkaisujen kohdalla SQLite oli kevyt ja nopea sekä minulle ennalta tuttu vaihtoehto, mutta skaalautuvuutta ja pitkän aikavälin

ylläpidettävyyttä ajatellen PostgreSQL olisi ollut parempi valinta. Nämä havainnot tukivat tietoperustassa esitettyä ajatusta teknologiavalintojen merkityksestä.

Näiden projektien työstäminen toi arvokasta kokemusta eri työskentelymallien ja ohjelmointiteknologioiden käytöstä. Jokaisessa projektissa oli sekä onnistumisia että haasteita, ja niiden analysointi tarjoaa hyödyllisiä oppeja tulevaisuuden ohjelmistokehitysprojekteihini. Jatkossa tulen kiinnittämään erityistä huomiota projektien alkuvaiheen suunnitteluun ja teknologisten valintojen kriittiseen tarkasteluun, jotta kehitysprosessi olisi mahdollisimman sujuva ja tehokas. Lisäksi voisi olla hyödyllistä panostaa enemmän testaus- ja dokumentointikäytäntöihin, jotta kehitysprosessin aikana tehty työ olisi paremmin hyödynnettävissä myös myöhemmissä projekteissa. Näin varmistan, että projektien aikana oppimani asiat siirtyvät käytännön kokemuksista osaksi laajempaa ohjelmistokehitysosaamistani.

Lähteet

Agile Alliance s.a. Agile 101. Luettavissa: https://www.agilealliance.org/agile101/. Luettu: 5. 2. 2025.

Airbyte. 16.9.2024. SQLite Vs PostgreSQL - Key difference. Luettavissa: https://airbyte.com/data-engineering-resources/sqlite-vs-postgresql. Luettu 16.2. 2025.

Atlassian. s.a. Waterfall methodolody: a comprehensive guide. Luettavissa: https://www.atlassian.com/agile/project-management/waterfall-methodology. Luettu 10.4. 2025.

Bautomo. (s.a.a.). https://bautomo.com/sanastoa/backend/

Bautomo. (s.a.b.). https://bautomo.com/sanastoa/frontend/

BTmalli s.a. BTmalli. Luettavissa: https://www.btmalli.fi/. Luettu: 5. 2.2025.

BTmalli s.a. Introduction to Business Technology Standard. Luettavissa: https://www.btmalli.fi/book/introduction/introduction-to-business-technology-standard/. Luettu: 5. 2. 2025.

Carver, Matthew. 2014. The responsive web. Manning publications. New York.

Chart.js. 15. huhtikuu 2025. Doughnuts and pie charts. Luettavissa: https://www.chartjs.org/docs/latest/charts/doughnut.html#pie. Luettu 17.3. 2025.

Cloudflare. s.a. What is data scraping? Luettavissa: https://www.cloudflare.com/learning/bots/what-is-data-scraping/. Luettu 9.4.2025.

Dyori, A. 18.11.2021. How to use an SQLite database in a Flask application. Luettavissa: https://www.digitalocean.com/community/tutorials/how-to-use-an-sqlite-database-in-a-flask-application. Luettu 16.2. 2025.

Elmhurst university. s.a. 5 key concepts to agile project management. Luettavissa: https://www.elmhurst.edu/blog/agile-project-management/. Luettu 17.3. 2025.

Git. s.a. Getting started – About version control. Luettavissa: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. Luettu 10.4.2025.

Haaga Helia. s.a. Ohjelmistokehityksen teknologioita(ajankohtaiset teknologiat). Luettavissa: https://opinto-opas.haaga-helia.fi/course_unit/SOF009AS3A Luettu 19.4.2025.

Kirvan, Paul & Lewis, Sarah & Lutkevich, Ben .2024. What is a Waterfall Model: Definition and guide. Luettavissa: https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model. Luettu 5.2. 2025.

Lanchec, S. 2024. User roles and permissions in software development. Forest Admin. Luettavissa: https://www.forestadmin.com/blog/user-roles-and-permissions-in-software-development/. Luettu 10.4.2025.

Long, B. 9.4. 2024. JavaScript and localStorage in a nutshell. Luettavissa: https://www.tiny.cloud/blog/javascript-localstorage/. Luettu 12. 2. 2025.

MDN. 3.Huhtikuuta 2025. JavaScript. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Luettu 12.2. 2025.

Nitin & Puja. 2025. TypeScript vs JavaScript: Which is Better? Know the Differences. Luettavissa: https://eluminoustechnologies.com/blog/typescript-vs-javascript/. Luettu 10.4.2025.

Pallets Projects. 2010a. Flask Documentation. Luettavissa: https://flask.palletsprojects.com/en/stable/. Luettu: 4. 2. 2025.

Pallets Projects. 2007. Frequently asked questions. Luettavissa: https://jinja.palletsprojects.com/en/stable/faq/#why-is-html-escaping-not-the-default. Luettu 16.2. 2025.

Pallets Projects. 2010b. What Flask Is and What Flask Is Not. Luettavissa: https://flask.palletsprojects.com/en/stable/design/#what-flask-is-what-flask-is-not. Luettu: 4. 2. 2025.

Python Software Foundation. 2001. Python: The Language. Luettavissa: https://www.python.org/doc/essays/blurb/. Luettu: 4. 2. 2025.

React. s.a. Luettavissa: https://react.dev/. Luettu 10. 2. 2025.

Sanastokeskus. s.a. TEPA-termipankki. Luettavissa: https://sanastokeskus.fi/tsk/fi/termitalkoot/haku-266.html?page=get_id&vocabulary_code=TSKTT&id=ID0114. Luettu 10. 2. 2025.

SAP. s.a. What is an API? Luettavissa: https://www.sap.com/finland/products/technology-platform/integration-suite/what-is-api.html. Luettu 10.4. 2025.

Schwaber, K & Sutherland, J. 2020. Scrum opas, Scrumin määritelmä ja pelisäännöt. Luettavissa https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Finnish.pdf. Luettu 12. 2.2025.

Sofela, O. 2023. Web storage: localStorage vs sessionStorage in JavaScript. freeCodeCamp. Luettavissa: https://www.freecodecamp.org/news/web-storage-localstorage-vs-sessionstorage-in-javascript/#heading-what-is-the-local-storage-object. Luettu 21.4.2025.

Sommerville, I. (2016). Software engineering (10th ed.). Pearson Education. Luettavissa: https://www.vlebooks.com/Product/Index/807705?page=0&startBookmarkId=-1. Luettu 16. 3. 2025.

SQLite. 16. Huhtikuuta 2025. About SQLite. Luettavissa: https://www.sqlite.org/about.html. Luettu 16.2. 2025.

TypeScript Team. 2025. Introduction to TypeScript. Luettavissa: https://www.typescript-lang.org/docs/handbook/intro.html. Luettu: 4. 2. 2025.

Zewe, A. 17. Tammikuuta 2025. Explained: Generative AI’s environmental impact. Luettavissa:https://news.mit.edu/2025/explained-generative-ai-environmental-impact-0117. Luettu 22.4.2025.