Riikka Kautonen, Artem Poliakov, Jere Pyörökivi

Attendance Checker -projekti

Projektisuunnitelma

Ohjelmistotuotantoprojekti 1

Tieto- ja viestintätekniikka, Ohjelmistotuotanto

Metropolia Ammattikorkeakoulu

19.1.2026

Sisältö

1 Johdanto

Tämä dokumentti on suunnitelma *Ohjelmistotuotantoprojekti 1* –kurssilla toteutettavalle projektille, jossa harjoitellaan erityisesti ohjelmistotuotantoa DevOps-toimintamallia hyödyntäen. Projektissa suunnitellaan ja toteutetaan sovellus, jolla opettajat voivat merkitä ylös koululuokkien ja erillisten kurssien opiskelijoiden läsnäolot opetuskertojen mukaan. Näin sovelluksen kautta pystyy seuraamaan luokan tai kurssin sekä yksittäisten opiskelijoiden läsnäoloja helposti ja tehokkaasti. Sovelluksen ja täten myös projektin työnimi on *Attendance Checker*, mutta se ei välttämättä ole sovelluksen lopullinen nimi.

Projekti on tarkoitettu kaikille, joita kiinnostaa tutustua ohjelmistotuotantoon ja sen eri vaiheisiin sekä DevOps-toimintamalliin ja muihin sovellettaviin metodeihin. Tämä projekti on pääasiassa myös oppimisprojekti, eli se on tarkoitettu *Ohjelmistotuotantoprojekti 1* –kurssia ohjaavalle opettajalle esittelemään projektin jäsenten työskentelyä ja oppimista. Projekti on siis myös tarkoitettu projektitiimin jäsenille, jotta he pääsevät oppimaan nykyaikaista ohjelmistotuotantoa käytännössä.

Projektissa käsiteltäviä ja harjoiteltavia aiheita ja osa-alueita ohjelmistotuotannon näkökulmasta ovat ohjelmiston vaatimusten analysointi, ohjelmiston arkkitehtuurin suunnittelu erilaisten UML- ja ER-kaavioiden avulla, ohjelmiston implementaatio Java-koodauskieltä käyttäen, ohjelmiston testaus ja laadunvarmistus mm. yksikkötesteillä, DevOps-automatisointi eli toistettavien tehtävien muuttaminen automaattisiksi, sekä ohjelmiston versiohallinta ja sen tuotannon dokumentointi.

2 Projektin tavoitteet

Projektin päätavoitteena on oppia, dokumentoida ja esitellä nykyaikaisen DevOps-toimintamallia käyttävän ohjelmistosuunnittelun perusteet, jotta

projektitiimin jäsenet ja muut projektiin tutustuvat pystyvät tulevaisuudessa muissa ohjelmistotuotantoprojekteissa hyödyntää osaamistaan ja sujuvoittaa projektien kulkua.

Muita projektin tavoitteita ovat:

- Tuottaa sovellus, jolla opettajat voivat seurata luokkien ja kurssien sekä opiskelijoiden läsnäoloja.
- Dokumentoida ohjelmistontuotannon vaiheita ja käyttää versiohallintaa selkeästi ja hyödyllisesti.
- Automatisoida ohjelmiston tuotantoon ja sovelluksen ylläpitoon liittyviä toistuvia tehtäviä.
- Oppia ohjelmiston pakkaus ja käyttöönotto Dockerilla.
- Kirjoittaa koodia, joka on selkeää, luettavaa ja testattavaa.
- Testata koodia ja parantaa koodin laatu.

3 Rajaus ja tuotokset

Projektissa keskitytään ymmärtämään, miten ohjelmistoja rakennetaan, testataan, pakataan, otetaan käyttöön ja ylläpidetään DevOps-toimintamallin perusteella. Projektissa opetellaan ohjelmiston arkkitehtuurin suunnittelua erityisesti UML- ja ER-kaavioiden avulla, koodataan suhteellisen yksinkertainen sovellus sekä opetellaan Jenkins:n ja Docker:n avulla automatisoimaan testausta, pakkausta, käyttöönottoa ja ylläpitoa.

Projektissa tuotetaan Java-koodauskielellä ohjelmoitu sovellus, jolla opettajat voivat seurata luokkien ja kurssien sekä niihin kuuluvien opiskelijoiden läsnäoloja. Sovelluksen täytyy olla toimiva, mutta, koska projektin päätavoitteena ei ole tuottaa monipuolista ja toiminnallisuuksiltaan runsasta sovellusta, sen ei tarvitse olla erityisen innovatiivinen ja visuaalisesti laadukas. Ohjelmiston koodin täytyy kuitenkin olla siistiä, luettavaa ja testattavaa.

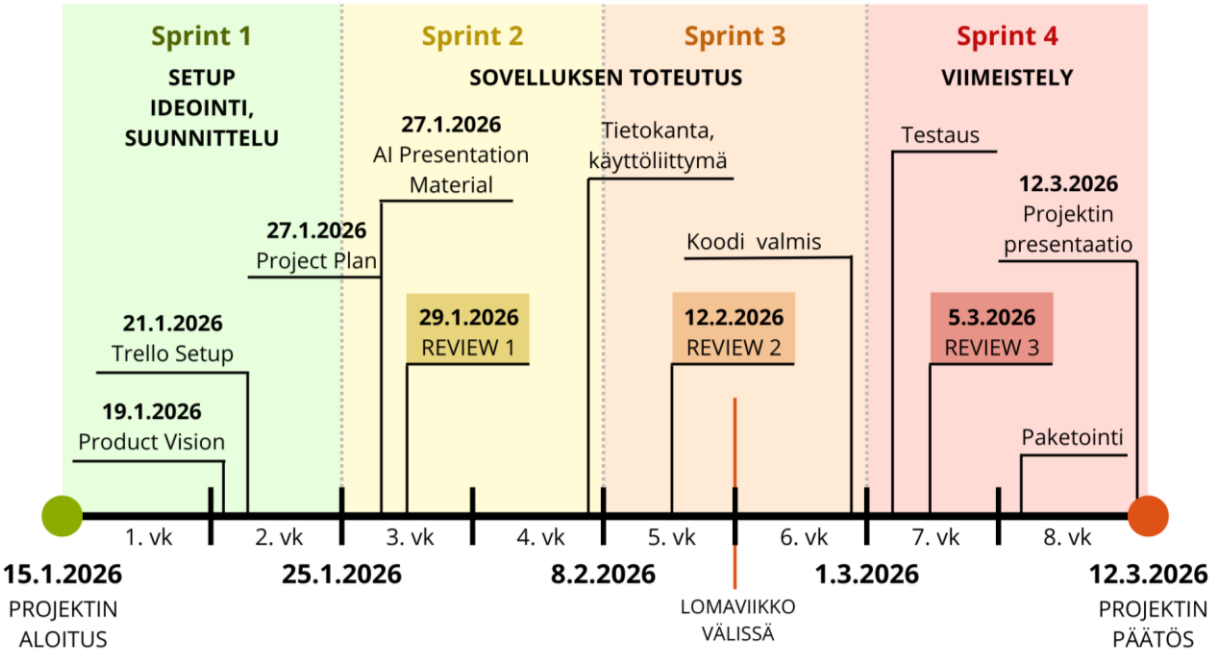
Projektissa tuotetaan useita UML- ja ER-kaavioita sekä muita ohjelmistoa kuvaavia kaavioita. Tuotetut kaaviota säilytetään projektin GitHub-repositoriossa omassa kansiossaan.

Projektissa ei esitellä Java-koodauskieltä ja sen ominaisuuksia sekä ohjelmiston koodin toimintaa tarkasti. Ohjelmiston koodia pyritään kommentoimaan ja dokumentoimaan GitHub-repositorioon, mutta toimintaa ei muuten esitellä tarkemmin, sillä projektin päätavoitteena ei ole opetella ja tutustua Java-ohjelmointiin.

4 Projektin aikajana

Projekti rakentuu neljästä vaiheesta, eli sprintistä, joissa jokaisessa on päätavoitteet. Kuvassa 1 on esitelty projektin alustava aikajana, jossa näkyy nämä neljä sprinttiä ja niiden sisältöä sekä projektin aloitus- ja päättymispäivämäärät. Väliin on myös sijoitettu sprintin viimeisen viikon päättymispäivämäärä. Kolmella ensimmäisellä vaiheella on omat katsaussessiot (review), jotka ovat myös merkitty kuvaan.

Ensimmäisen sprintin tehtäville on annettu deadlinet, mutta myöhempien sprinttien tavoitteille ei ole vielä asetettu määräaikoja, joten ne ovat kuvassa suuntaa antavasti. Projektin kulkua suunnitellaan tarkemmin Trello-sivustolle. Jokaisella vaiheella on oma taulunsa, ja vaiheen sisältämät tehtävät lisätään tauluun aina vaiheen alkaessa. Tehtäville myös asetetaan silloin määräajat.



Kuva 1. Projektin alustava aikajana.

Tavoitteina ensimmäisessä vaiheessa (sprint 1) on koota projektitiimit, laittaa pystyyn tiimin projektinhallinta sekä versiohallinta, valita projektin aihe ja suunnitella toteutettavaa sovellusta luomalla sille käyttäjätarinoita. Vaiheessa siis perustetaan projektille Trello-sivu ja GitHub-repositorio sekä suunnitellaan sovelluksen toteutusta. Vaiheeseen kuuluu myös materiaalin kerääminen ja kokoaminen AI-aiheista esitystä varten sekä projektisuunnitelman kirjoittaminen.

Toisessa vaiheessa (sprint 2) tavoitteina on suunnitella ja toteuttaa ohjelmiston tietokanta, aloittaa sovelluksen käyttöliittymän koodaus sekä kirjoittaa sovelluksen backend- ja frontend-koodia yksikkötestien kautta. Kolmannessa vaiheessa (sprint 3) jatketaan koodausta. Sovelluksen backend ja kriittiset ominaisuudet on tavoitteena saada valmiiksi tässä sprintissä. Tavoitteena on myös automatisoida ohjelmistoa.

Neljännessä ja viimeisessä vaiheessa (sprint 4) on tavoitteena viimeistellä toteutettava ohjelmisto ja sovellus. Vaiheessa tehdään ohjelmistolle testausta,

ja ohjelmisto paketoidaan ja käyttöönotetaan Dockerilla. Tämä vaihe päättyy ohjelmiston ja sovelluksen esittelyyn, jota varten kootaan esitys.

5 Resurssien jako

Projektitiimi koostuu kolmesta jäsenestä. Tiimin jäsenet ovat Riikka Kautonen, Artem Poliakov ja Jere Pyörökivi. Jokaisella jäsenellä on yhtä suuret vastuualueet, mutta vastuutehtävät ja työt jaetaan aina vaiheiden mukaan jäsenten kiinnostusten ja osaamisen perusteella. Tehtävät pyritään jakamaan tasaisesti kaikkien jäsenten kesken. Projektitiimillä ei ole varsinaista johtajaa, vaan jokaisen sprintin, eli vaiheen, aikana yksi jäsen toimii ikään kuin Scrum Masterina, ja huolehtii tehtävien laatimisesta ja niiden palautuksista. Jokainen jäsen toimiin Scrum Masterina vähintään yhden kerran.

Työkaluna projektinhallintaan käytetään Trello-verkkosivustoa. Sinne suunnitellaan jokaisen sprintin sisältö, eli projektivaiheen, tehtävät. Tehtäville asetetaan määräajat ja niistä vastaavat henkilöt sekä mahdollisesti pilkotaan tehtävät pienempiin osiin tarkistuslistojen avulla. Tehtäviä siirretään listoista toiseen niiden tilojen perusteella, kunnes ne ovat suoritettuja. Versionhallinnan työkaluna käytetään GitHub-sivustoa, jonne tiimille luodaan repositorio koodin ja tiedostojen säilyttämistä varten.

Tuotettavan sovelluksen koodin kirjoittamiseen ja ohjelmointiin käytetään erilaisia ohjelmistonkehitysympäristöjä. Ensisijaisesti projektissa käytetään IntelliJ IDEA -ohjelmistonkehitysympäristöä, jolla kirjoitetaan Java-koodauskielellä ohjelmia. Jenkis-ohjelmaa ja -serveriä käytetään ohjelmiston tehtävien automatisointiin, ja Docker-ohjelmaa käytetään ohjelmiston pakkaamiseen ja käyttöönottoon. Ohjelmiston tietokanta toteutetaan MariaDB:llä, joka on relaatiotietokantajärjestelmä. UML-kaavioiden rakentamiseen käytetään StarUML-työpöytäsovellusta.

Apumateriaalina projektissa hyödynnetään kurssin opetusmateriaalia ja ohjeita, jotka koskevat DevOps-toimintamallin sekä eri ohjelmien ja työkalujen käyttöä. Materiaalit ovat kurssin opettajan GitHub-repositorioissa, ja linkit näihin ovat kurssin sivuilla Metropolian OMA-sivustolla. Tähän suunnitelmaan ei laiteta linkkejä materiaaleihin.

6 Riskienhallinta

Tämän projektin aikana mahdollisesti eteen tulevat riskit:

Projekti ei ole valmis aikarajaan mennessä:

Projektin valmistumiseen menee liikaa aikaa ja aika loppuu kesken. Ajan loppumisen riski on hyvin suuri ja voi tapahtua helposti. Kaikkien projektiin osallistuvien pitää siis olla aktiivisia ja tehdä oma osansa. Etenkin Scrum Masterin pitää olla aikatauluista selvillä. Hänen pitää varmistaa, että työt etenevät ja pysytään aikataulussa.

Lisätoiminnot:

Projektiin lisätään toimintoja, joita ei mainita alkuperäisessä suunnitelmassa. Suunnittelemattoman lisääminen on hankalaa, hidasta ja vie aikaa muista osista.

Vähäinen testaus:

Projektia ei testata paljoa, jolloin virheitä jää helposti huomaamatta. Projekti ei toimi ja on keskeneräinen. Testaus on tärkeä osa projektia ja niitä täytyy tehdä. Muuten projetki helposti epäonnistuu täysin. Riski on helppo välttää tekemällä reilusti testauksia ennen julkaisemista.

Saavutettavuus:

Projektin ulkonäkö ei ole optimoitu, esimerkiksi: liian pieni fontti tai projektia on hankala käyttää. Riskin todennäköisyys on kohtalainen. Vaikeasti käytettävä ohjelma vähentää halua käyttäjien tyytyväisyyttä. Ohjelman käyttö voi olla ärsyttävää ja käyttäjä saattaa lopettaa ohjelman kokonaan.

Turvallisuus:

Luvaton henkilö pääsee käsiksi oppilastietoihin. Riski on kohtalainen. Oppilaiden tietoja voidaan käyttää vääriin tarkoituksiin ja se pistää heidät mahdolliseen vaaraan. Tietovuoto tuo myös huonoa mainetta ja vähentää luottamusta. Täytyy varmistaa, että käytetyt turvatoimet toimivat ja projekti toimii odotetusti.

Suorituskyky:

Projekti toimii hitaasti. Riski on kohtalainen. Hidas toiminta vähentää ohjelman käytettävyyttä ja sen käyttöönottoa. Projektia tulee jatkuvasti käydä läpi, jotta mahdollinen hitaus tulee esiin.

7 Testaus ja laadunvarmistus

Projekti käy läpi kaikki tärkeimmät testauksen vaiheet, koska testaus on olennainen osa jokaista kehitysprosessia ja ohjemistotuotantoprojektia. Sen avulla voidaan varmistaa, että ohjelmisto toimii oikein ja vastaa projektin vaatimuksia. Projektissa testausta suoritetaan eri vaiheissa, mikä mahdollistaa virheiden huomaamisen ja korjaamisen ajoissa.

Projektissa käytettävät testausmenetelmät:

- Yksikkötestaus (unit tests), jossa ohjelman yksittäisiä komponentteja testataan erikseen, jotta voidaan varmistaa niiden toiminta.

- Integrointitestauksessa tarkastetaan eri osien yhteistoimintaa ja tiedonsiirron toimivuutta.

- Toiminnallinen testaus suoritetaan, jotta voidaan varmistaa, että ohjelmisto toteuttaa kaikki projektin vaatimuksissa määritellyt toiminnot.

- Projektin viimeisessä vaiheessa on käyttäjätestaus, jonka avulla arvioidaan ohjelman käytettävyyttä ja ymmärrettävyyttä käyttäjän näkökulmasta.

Testausta pidetään onnistuneena, kun kaikki keskeiset virheet on havaittu ja korjattu, ohjelma toimii vakaasti ilman vakavia häiriöitä ja toteuttaa kaikki projektille asetetut vaatimukset. On myös tärkeää, että käyttäjä ymmärtää helposti, miten ohjelmaa käytetään eikä kohtaa vaikeuksia perustoimintojen suorittamisessa.

Testauksessa käytetään sekä manuaalisia että tarvittaessa automatisoituja menetelmiä. Toiminnallisuuden ja käytettävyyden testaaminen suoritetaan periaatteessa manuaalisesti, koska se mahdollistaa ohjelman arvioimista paremmin käyttäjän näkökulmasta. Havaitut virheet ja testauksen tulokset dokumentoidaan, mikä auttaa seuraamaan ongelmia ja parantamaan projektin laatua.

8 Dokumentointi ja raportointi

Projektin aikana seuraavia dokumentaatiomenetelmiä tullaan käyttämään:

1. Projektin vaatimusdokumentti – Projektiin liittyvät tekniset vaatimukset, joita noudatetaan projektin tekemisessä

2. Viikoittaiset päivitykset - Tulemme lisäämään viikoittain päivityksiä projektin etenemiseen liittyen, esimerkiksi: Trello
3. Trello – Jatkuvasti päivittyvä projektinhallintataulu
4. Viikoittaiset tapaamiset – Tapaamme ryhmän kanssa viikoittain keskustelemaan tuotoksista ja miten hommia jatketaan
5. Projektiraportti – yhteenveto saavutuksista, haasteista ja tuloksista