Miniprojektin loppuraportti

Ryhmä 4: Patrik Alanen, Milla Kelhu, Leevi Kokkinen, Kalle Lindholm, Riku Mattila, Jaakko Vuohtoniemi

Ensimmäisen sprintin aikana prosessiin liittyen teknisiä ja laadullisia ongelmia muodostivat kokemattomuus, kunnianhimoinen DoD ja vaatimukset sovellukselle, jotka kostautuivat integraatiossa. Scrumin oppimisen kannalta yksinkertaisemmatkin vaatimukset ja esimerkiksi komentorivikäyttöliittymä olisivat riittäneet. Retrospektiivin pitäminen sekä taskboardin käyttö ja ylläpito jäivät vähälle. Projektityöskentelyssä oli katkoksia tiedonkulussa, mikä johti kovakoodauksiin. Lisäksi joihinkin paikkoihin oli jäänyt turhia tiedostoja ja kaksoiskappaleita. Teknisiä ongelmia esiintyi ohjelmistoa Herokuun siirrettäessä. Hankaluuksia aiheuttivat edellä mainitut kovakoodaukset ja eri tasoilta tulevat importaukset luokissa. Valittu tech stack oli kunnianhimoinen. Sprintin storyt jäivät testaamatta, eivätkä täyttäneet DoD'ia. CI:tä ei otettu heti alusta kunnolla käyttöön.

Toisen sprintin prosessin retrospektiivinen tarkastelu jäi lyhyeksi. Projektityöskentelyä vaivasi ajanhukkaa aiheuttanut kommunikaatiokatkos. Tietokannan toteutukseen liittyi epäselvyys siitä, luodaanko kaksi taulua, joista toisessa on viittaus toiseen, vai luodaanko kolmas, viittaukset sisältävä taulu. Vaikka asia toteutettiin lopulta tietyllä tavalla, yhteen branchiin oli tehty koko sprintin ajan turhaa työtä toisella tavalla. Sprintin aikana huomattiin, että PostgreSQL aiheutti kohtuuttomasti ongelmia. Testaus ei jostain syystä toiminut PostgreSQL:n kanssa kaikkien koneilla. Sprintin lopuksi päätettiinkin vaihtaa sqliteen.

Kolmannen sprintin retrospektiivi jäi pitämättä. Projektityöskentelyn ongelmaksi osoittautui siihen asti lapsipuolen asemaan jätetty testaus. Lähes kaikki robot-testit tehtiin sprintin aikana. Niitä ei saatu kovin kattaviksi käytössä olleilla työtunneilla. Sivun toimivuus oli epävarmaa, ja sitä testattiin ja testejä kirjoitettiin vielä juuri ennen demoa. Juuriongelmana tähän voidaan pitää työn epätasaista jakautumista. Viime hetken paniikki olisi ollut vältettävissä.

Projektin sujumista auttoi se, että apua löytyi periaatteessa kellon ympäri. Ryhmän telegramissa oli lähes aina joku paikalla. Ryhmästä löytyi paljon erilaista osaamista ja erilaisia kiinnostuksen kohteita, minkä ansiosta työnjako kävi luontevasti.

Tiheämmin pidetyt daily scrumit olisivat nopeuttaneet ongelmien havaitsemista ja niihin reagoimista. Parannettavaa seuraavaan kertaan jäi etenkin hukan vähentämisessä. Olisi ollut helppo pysyä "mitä sain aikaan/mitä seuraavaksi/mitä esteitä"-formaatissa. Työtä olisi ollut hyvä tasoittaa sen sijaan, että pari tuntia ennen asiakastapaamista ongelmia setvitään hiki hatussa. Toisinaan ongelmana oli myös toisen työn odottaminen, että saa tehtyä jonkun tietyn osan tai palasen omasta taskistaan.

Osa työstä hajautui ja hukkui brancheihin. Työtä yhden taskin kanssa tehtiin koko sprintin ajan, loppumetrejä lukuun ottamatta, omassa branchissaan, ja se perustui yleensä master-haaran vanhaan versioon, joka oltiin saatettu päivittää täysin erilaiseksi. Pienemmät inkrementaaliset merget masteriin päivittäin ja vähempi määrä eri branchejä olisivat voineet

vähentää työn pirstaloitumista ja siitä seuraavia integraatio-ongelmia. Integraatiosta tuli siten melkoinen taakka, ja osa työstä oli pahimmillaan toteutettu kahdessa branchissa. Ensi kerralla voisi olla hyvä kokeilla pysyä lähempänä master- tai main-haaraa, esimerkiksi käyttämällä TBD:tä. Hukkaa ja integraatio-ongelmia oltaisiin voitu vähentää myös tehokkaammalla, tiheämmällä ja selkeämmällä kommunikaatiolla. Lisäksi ehdoton kiinnipito CI:n toimivuudesta olisi varmasti auttanut. CI:n tehtävät olisi siis hyvin olennaista määritellä heti projektin alussa ja pitää kiinni siitä että sen tehtävät toteutuvat. Taskboard olisi ehkä toiminut paremmin Excelissä tai Sheetsissä, jotka olisivat tarjonneet huomattavasti enemmän muokkausmahdollisuuksia. GitHub Projects oli joiltain osin turhan kankea.

Opimme hyvän ja selkeän suunnittelun merkityksen työskennellessämme ryhmänä. Erilaisten vaatimusten määrittely (esimerkiksi user storyn taskit) on tehtävä yksityiskohtaisesti, jotta kaikki ymmärtävät vaatimukset samalla tavalla. Toisin kuin yksilöprojekteissa, nyt oli tarpeen varmistaa erillisellä huolellisuudella, että kaikki olivat samalla aaltopituudella. Hyvän kommunikaation merkitys korostui. Samojen toimintatapojen noudattaminen esimerkiksi versionhallinnassa selkeyttää sovelluksen tilannetta muille ryhmäläisille. Tehdessämme sprinttien loppurutistuksia ennen asiakastapaamisia opimme myös teknisen velan hyötyjä ja haittoja. Integraatiovaikeuksia, testejä jää tekemättä, kovakoodausta vain toiminnan esittelyä varten – tällaista teknistä velkaa saattoi olla jopa ärsyttävää ja turhauttavaa maksaa jälkikäteen. GitHubin kanssa toimimisesta opittiin koko ryhmänä lisää, kuten myös PostgreSQL:n ja sqlite:n eroista ja ongelmista esimerkiksi testauksen ja Herokun suhteen.

Olisimme halunneet oppia TBD:n käytön. Se olisi voinut olla hyödyllisempää projektin kannalta, ja säästänyt integraatioihin käytettyä aikaa huomattavasti. CI oli rikki lähes koko projektin ajan. Sen kanssa työskentelystä olisi voinut saada lisäarvoa. Scrumia olisi ollut hyvä harjoitella vielä muutama viikko. Ensimmäisillä viikoilla aikaa kului paljon vain hyvien työtapojen ja käytänteiden hakemiseen. Ryhmän työskentelytavat kehittyivät viikkojen kuluessa, mutta kolme viikkoa on hyvin lyhyt aika. Olisi ollut mielenkiintoista nähdä, miltä ryhmän työskentely olisi näyttänyt erikseen määritetyn scrum masterin kanssa. Nyt scrum masterina ei ollut kukaan, ja samalla kaikki.

Turhalta tuntuivat branchit erillisille taskeille. Taisivat olla parhaimmillaankin turhia, ja pahimmillaan valtava päänsärky.