Oskari Piiroinen

Kommentointityökalun toteuttaminen verkkokoulutusalustalle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

2.4.2024

Tiivistelmä

Tekijä:	Oskari Piiroinen
Otsikko:	Kommentointityökalun toteuttaminen verkkokoulutusalustalle
Sivumäärä:	35 sivua + 4 liitettä
Aika:	2.4.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mediatekniikka
Ohjaajat:	Tutkijaopettaja Hannu Markkanen

Insinöörityön tavoitteena oli kehittää toimeksiantajan verkkokoulutusalustaan vuorovaikutukseen liittyviä ominaisuuksia tukemaan yhteistyötä verkkokoulutusten suunnittelussa. Toimeksiantajan verkkokoulutusalustalta puuttui mahdollisuus kommunikoida verkkokoulutuksen sisällöstä muiden käyttäjien kanssa. Verkkokoulutuksien suunnittelussa käytettiin muita viestintäkanavia, ja jouduttiin käyttämään kuvankaappauksia verkkokoulutuksen sisällöstä.

Työn tarkoituksena oli toteuttaa verkkokoulutusten kommentointityökalu toimeksiantajan verkkokoulutustyökaluun. Kommentointityökalu mahdollistaa verkkokoulutuksesta kommunikoimisen samassa paikkaa, jossa sisältökin on. Tämä vähentää useiden viestikanavien ja kuvankaappauksien käyttämistä.

Työn toteutus sisälsi frontend- ja backend-ohjelmointia. Työtä varten tehtiin muutoksia toimeksiantajan verkkokoulutusalustan olemassa olevaan ohjelmointirajapintaan. Esimerkiksi toteutusta varten suunniteltiin ja luotiin tietokantaan uusi taulu.

Työ eteni opinnäytetyön puitteissa testausvaiheeseen saakka. Työ julkaistaan testausvaiheen jälkeen. Toimeksiantajan sisällöntuottajilta kerättiin palautetta ennen testausvaihetta. Kaikki vastaajat kokivat työn hyödylliseksi ja aikovat käyttää sitä julkaisun jälkeen. Palautteen pohjalta koottiin jatkokehitysideoita, joita toteutetaan työn julkaisun jälkeisiin versioihin. Asiakkaiden palautetta työstä kuullaan julkaisun jälkeen.

Avainsanat: verkkokoulutusalusta, verkkokoulutus, verkkotyökalu, ohjelmointirajapinta, kommentointityökalu

---

Abstract

Author:	Oskari Piiroinen
Title:	Creating a commenting tool on an eLearning platform
Number of Pages:	35 pages + 4 appendices
Date:	2 April 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Media Technology
Supervisors:	Hannu Markkanen, Researching Lecturer

The goal of this final year project was to develop the communication-related features of a client's eLearning platform to support collaboration in online training planning processes. The client's eLearning platform was lacking the possibility to communicate with other users about the online training's content. Other communication channels and screenshots were used in the planning process of the online training's content.

The purpose of this project was to create a commenting tool for the online training on the client's eLearning platform. The commenting tool enables communication about the online training in one and the same place. This decreases the use of multiple communication channels and screenshots. Front-end and back-end programming was needed during the execution of this project and; also changes to the client's existing application programming interface were made. For example, a new table was planned and created in the client's database.

The final year project progressed well. Testing was not included in the project, but will be performed soon. The developed features will be released after the testing phase. Feedback was collected from the client's content producers before the testing phase. All the respondents found the work useful and plan to use it after the release. Based on the feedback, further development ideas were determined. The ideas will be implemented in the versions after the release of the project. Feedback will continue to be collected from the users of the client's eLearning platform after the release of the project.

Keywords: eLearning platform, online training, online tool, application programming interface, commenting tool

Sisällys

Lyhenteet ja käsitteet

AJAX: *Asynchronous JavaScript and XML*. Verkkokehitystekniikka, jossa verkkosovellus hakee sisältöä palvelimelta ja näyttää vastauksena saatua sisältöä ilman sivun uudelleenlatausta.

API: *Application programming interface*. Joukko protokollia, joka mahdollistaa eri ohjelmistokomponenttien kommunikaation ja datan siirron.

Builder: Toimeksiantajan verkkokoulutusalusta, jolla verkkokoulutuksia tehdään ja ylläpidetään.

Elementti: Toimeksiantajan verkkokoulutusalustan verkkokoulutuksen yksittäinen sivu.

Elementtityyppi:

Toimeksiantajan verkkokoulutusalustan yksittäisen sivun pohja. Pohjan mukaan tehdään tietyn tyyppisiä sivuja, jotka ovat kopioita pohjasta. Pohja koostuu HTML-, CSS- ja JavaScript-tiedostoista.

Fasilitaattori:

Toimeksiantajan verkkokoulutusalustan käyttäjä, jolla on käyttäjätunnus verkkoalustalle.

Feature: Toimeksiantajan verkkokoulutusalustan lisätoiminnallisuus, jonka toimintaa voidaan määritellä asiakasympäristökohtaisesti.

Komponentti:

Toimeksiantajan verkkokoulutusalustan yksittäisen sivun yksittäinen tieto esimerkiksi kysymysteksti.

LCMS: *Learning Content Management System*. Järjestelmä, jolla voidaan luoda, varastoida, uudelleen käyttää, hallita ja jakaa oppimismateriaalia keskitetystä säilytyspaikasta, yleensä tietokannasta.

LMS: *Learning Management System*. Ohjelmisto verkkokurssien hallinnointiin, seurantaan ja raportointiin.

Moduuli: Toimeksiantajan verkkokoulutusalustalla luotu yksittäinen verkkokoulutus.

Moduulityyppi:

Toimeksiantajan verkkokoulutusalustan asiakasympäristön konfiguroitava pohja verkkokoulutuksen rakentamiselle. Moduulityypille voidaan suunnitella esimerkiksi oma ulkoasu.

MVC: *Model-View-Controller*. Ohjelmistosuunnittelumalli. Malli (Model) on yhteydessä tietokantaan, josta haetaan tietoa näkymään (View) eli käyttöliittymään. Ohjain (Controller) eristää mallin sisältämän toimintalogiikan näkymän käyttöliittymäelementeistä ja käsittelee, miten sovellus vastaa käyttäjän vuorovaikutukseen näkymässä.

PhpMyAdmin:

Ohjelmistotyökalu, joka on tarkoitettu MySQL:n hallintaan verkossa.

SaaS: *Software as a service*. Ohjelmiston jakelumalli, jossa ohjelmisto jaetaan Internetin välityksellä käyttäjille maksua vastaan.

SCORM: *Sharable Content Object Reference Model*. Yleisin verkkokoulutusstandardi, joka mahdollistaa verkkokoulutusten jakamisen eri LMS-järjestelmien välillä.

Tenant: Toimeksiantajan verkkokoulutusalustan asiakkaan oma asiakasympäristö, jolla on omat käyttäjät, asetukset ja verkkokoulutukset.

Verkkokoulutusalusta:

Verkkopohjainen ohjelmisto, joka tarjoaa kokonaisvaltaisen koulutusratkaisun yrityksille.

XAMPP: *Cross-Platform, Apache, MYSQL, PHP and Perl*. Verkkopalvelin, joka mahdollistaa ohjelmiston kehittämisen paikallisella verkkopalvelimella ennen julkaisua.

Zend: Ohjelmistokehys, jota käytetään rakentamaan PHP-verkkosovelluksia.

1 Johdanto

Työn toimeksiantaja Apprix on ohjelmistoalan yritys, joka tarjoaa osallistavia verkkokoulutuksia yrityksille. Yrityksessä työskentelee alle 20 henkilöä. Yrityksen asiakaskunta koostuu erikokoisista yrityksistä, yhdistyksistä ja liitoista niin kotimaasta kuin ulkomailtakin. Asiakkaita on eri sektoreilta kuten energia-, rakennus- ja elintarvikealalta. Apprixin päätuote on Builder. Builder on verkkokoulutusalusta, jolla voidaan tehdä, jakaa ja seurata verkkokoulutuksia. Verkkokoulutuksia voidaan integroida asiakkaan intra-, ekstranet-, kulunvalvonta- ja HR-järjestelmiin. Verkkokoulutusten suorituksia tilastoidaan. Asiakas voi itse tehdä verkkokoulutukset Builderilla tai yhteistyössä Apprixin sisällöntuottajien kanssa. [1.] Builder on SaaS-palvelu. SaaS (Software as a service) tarkoittaa ohjelmiston jakamista Internetin välityksellä. Käyttäjä saa ohjelmiston käyttöönsä maksua vastaan, eikä hän tarvitse sen käyttöön yleensä kuin Internetselaimen. Ohjelmiston tarjoaja vastaa kaikesta ohjelmistoon liittyvästä kuten tietoturvasta. [2.]

Työn tavoitteena on kehittää toimeksiantajan verkkokoulutusalustaan vuorovaikutukseen liittyviä ominaisuuksia tukemaan yhteistyötä verkkokoulutusten suunnittelussa. Tarkoituksena on toteuttaa verkkokoulutusten kommentointityökalu toimeksiantajan verkkokoulutusalustalle. Toimeksiantaja saa työstä tarvitsemansa uudistuksen Builder-tuotteeseen. Aiemmin Builderin käyttäjät ovat joutuneet käyttämään muita viestintäkanavia kommunikointiin, koska sellainen on puuttunut Builderista. Kommentointityökalun tarkoituksena on mahdollistaa kommunikointi käyttäjien välillä. Kommentointityökalun ensimmäinen versio on tarkoitettu sisällöntuottajille koulutusten suunnittelun työkaluksi. Sisällöntuottajat voivat esimerkiksi saada palautetta suunnittelemastaan koulutuksesta suoraan Builderissa ilman muita viestintäkanavia.

Olen työskennellyt Apprixilla kaksi vuotta frontend-kehittäjänä. Työn aihe on valittu Apprixin tulevista projekteista ja sen valintaan on päästy vaikuttamaan. Aihe vaikutti mielenkiintoiselta ja opettavaiselta, koska se sisälsi frontend- ja backend-ohjelmointia.

2 Verkkotyökalut verkkokoulutuksen suunnitteluprosessin tukena

2.1 Verkkokoulutusalustat

Verkkokoulutusalusta on laaja termi. Sitä voidaan kuvailla verkkopohjaiseksi ohjelmistoksi, joka tarjoaa kokonaisvaltaisen koulutusratkaisun yrityksille. Sillä voi tehdä, jakaa ja seurata koulutuksia Internetin välityksellä. Koska verkkokoulutusalusta on laaja termi, se kattaa monenlaisia koulutusohjelmistoja, jotka voivat erota toiminnallisuuksiltaan. Verkkokoulutusalusta voi tarjota vain koulutuksen tekemisen tai koulutusten jakamisen ja seuraamisen. [3.]

Builder on verkkokoulutusalusta, joka voidaan määritellä LCMS-järjestelmäksi (Learning Content Management System). Se on hybridijärjestelmä, jossa yhdistyy LMS- (Learning Management System) ja CMS-järjestelmä (Content Management System). CMS-järjestelmä mahdollistaa kurssin luomisen lataamalla yleisimpiä tiedostotyyppejä kuten Word ja PowerPoint. Käyttäjän ei tarvitse itse muuntaa niitä web-muotoon kuten HTML:ssä. CMS-oppimisjärjestelmä mahdollistaa yhteistyön eri toimijoiden välillä eri toimintojen avulla. Nämä toiminnot on suunniteltu mahdollistamaan: tietoon pääsyn riippuen toimijan oikeuksista, tietojen keräämisen ja jakamisen, tietojen tallentamisen avustamisen, päällekkäisen sisällön tarkastamisen ja raportoimisen. [4, s. 644–645.]

LMS-järjestelmä yhdistää yleisimmät verkko-oppimisen toiminnot yhdeksi sovellukseksi. Se koostuu ohjelmistoista tai ohjelmista, jotka automatisoidusti mahdollistavat verkkokurssien hallinnoinnin, seurannan ja raportoinnin. Se tarjoaa keskitetyn ratkaisun verkkokurssien aikataulutukseen, oppijoiden rekisteröitymiseen ja oppijoiden oppimistulosten arvioimiseen. LMS-järjestelmätoimintoihin kuuluu keskitetty ja automatisoitu ylläpito, itsepalveluiden ja itseneuvovien palveluiden käyttö, oppimismateriaalin koonti ja jakaminen nopeasti, koulutusaloitteiden yhdistäminen skaalautuvalla verkkopohjaisella alustalla, tuki siirrettävyydelle ja standardeille, sisällön mukauttaminen ja tiedon uudelleenkäytön mahdollistaminen. [4, s. 645–646.]

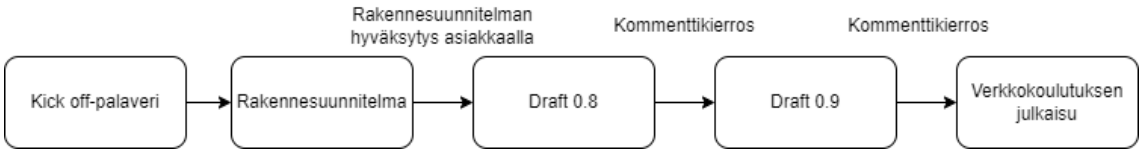
LCMS-järjestelmällä kehittäjät voivat luoda, varastoida, uudelleen käyttää, hallita ja jakaa oppimismateriaalia keskitetystä säilytyspaikasta, yleensä tietokannasta. LCMS-järjestelmän tarkoitus on mahdollistaa rajattoman verkkokurssien määrän käsittely ja hallinnointi. [4, s. 646–647.] LCMS-järjestelmä helpottaa sisällön uudelleen käyttöä, sisällön tuottamista, oppijoiden hallinnointia, oppijoiden kehittymisen seuraamista ja yhteistyötä. IDC:n (International Data Corporation) mukaan LMCS-järjestelmää käytetään personalisoidun verkkokoulutusmateriaalin luomiseen, varastoimiseen, kokoamiseen ja jakamiseen oppimisaihioiden muodossa. [5, s. 460.]

LCMS-järjestelmässä on säilytyspaikka oppimisaihioille ja työkalu niiden luomiseen. LCMS-järjestelmä on dynaaminen julkaisutyökalu verkkokoulutusmateriaalin jakamiseen. Järjestelmällä pystyy luomaan ja julkaisemaan verkkokursseja online- ja offline-muodossa. Verkkokoulutuksia voi jakaa eri formaateissa kuten HTML ja PDF. Verkkokoulutuksien suorituksia seurataan. Verkkokoulutuksia, opiskelijoiden arkistoja ja edistymistä sekä oppimisaihioita voidaan hallinnoida. Kouluttajat ja oppilaat voivat olla vuorovaikutuksessa keskenään esim. sähköpostin avulla. Järjestelmä tukee alan yleisimpiä standardeja kuten AICC:tä, SCORM:ää, IMS:ää, HTML:ää ja XML:ää toimiakseen myös muilla alustoilla. [5, s. 460.]

2.2 Verkkokoulutukset ja suunnitteluprosessi

Koulutus tarkoittaa muodollisessa koulujärjestelmässä organisoituja erilaisia kasvatuksen, opetuksen ja oppimisen toimintoja [6, s. 66]. Koulutuksen tavoitteena on kartuttaa kompetensseja ja osaamista, joilla on käyttöä työssä, ammatissa tai muussa sosiaalisessa elämässä [6, s. 67]. Oppimisella tarkoittaa laajimmassa merkityksessään prosessia, jossa tapahtuu muutosta oppijan taidoissa, tiedoissa, ajattelutoiminnassa tai toiminnassa [6, s. 59]. Verkkokoulutuksella tarkoitetaan oppimista tai koulutusta Internetin välistyksellä päätelaitteella kuten tietokone tai mobiililaite. Verkkokoulutus voi sisältää esim. tekstiä, kuvia, videoita, animaatioita ja pelillisiä toimintoja kuten tietovisoja ja testejä. Verkkokoulutuksia jaetaan yleensä esim. oppimisen hallintajärjestelmän (LMS) kautta. [7.]

Builderilla asiakkaat voivat itse suunnitella verkkokoulutuksia tai vaihtoehtoisesti tilata verkkokoulutuksen suunnittelun Apprixilta. Verkkokoulutuksen suunnittelu tilataan kahden pääsyyn takia: asiakkaalla ei ole aikaa suunnitella verkkokoulutusta tai asiakas haluaa laadukkaamman verkkokoulutuksen. Tilaus voi saada alkunsa siitä, että Apprixin asiakasvastaava näyttää asiakkaalle, millaisia koulutuksia muut asiakkaat ovat toteuttaneet ja kyseinen asiakas haluaa toteuttaa samankaltaisen koulutuksen. Asiakasvastaava sopii asiakkaan kanssa kuluista. Mahdollisista lisätöistä laskutetaan erikseen projektin aikana. Sisällöntuotantoprojektille päätetään heti alussa projektipäällikkö, jonka tehtävä on pitää projektin aikataulusta huolta eli koordinoida, että kaikki asiat tulevat tehtyä ja etenevät aikataulussa. [8.] Apprixin sisällöntuotantoprosessi voidaan jakaa eri vaiheisiin (kuva 1).



Kuva 1. Apprixin sisällöntuotantoprosessin vaiheet.

Kuvassa 1 nähdään Apprixin sisällöntuotantoprosessin eri vaiheet. Virallisesti sisällöntuotantoprojekti alkaa kick off -palaverilla, johon osallistuu asiakasvastaava, sisällöntuottajat ja erikoisasiantuntija. Asiakas on tähän mennessä toimittanut materiaalin, jonka pohjalta verkkokoulutus tehdään. Apprixin sisällöntuottajat ja erikoisasiantuntija käsittelevät asiakkaan toimittaman aineiston läpi ja poimivat sieltä avainkohtia, joita käytetään verkkokoulutuksen tehtävissä. Lisäksi asiakasvastaava tai sisällöntuottaja tekee aineiston pohjalta rakennesuunnitelman. Rakennesuunnitelma on ajatuskartta, jossa aineisto on jaoteltu pienempiin palasiin. [8.]

Sisällöntuotantoprojektissa asiakasta voi edustaa henkilöitä hyvin eri taustoista riippuen siitä, kenelle sisältöä ollaan tekemässä. Asiakkaan edustajat voivat olla esim. insinöörejä, HR-henkilöstöä, myyntihenkilöstöä ja yrityksen johtoa. Asiakkaan kanssa käydään läpi rakennesuunnitelma ja avainkohdat. Verkkokoulutuksen suunnittelussa halutaan panostaa ongelmakohtiin, joita työtekijät kohtaavat

työssään. Asiakas hyväksyy suunnitelman, jonka pohjalta tehdään Draft 0.8 -versio. Versio on 80 prosenttia lopullisesta versiosta. Käytännössä asiakkaalle luodaan verkkokoulutus toimeksiantajan verkkokoulutusalustalle. Verkkokoulutukseen lisätään suurin osa suunnitellusta sisällöstä. Asiakas näkee ensi kertaa verkkokoulutuksen konkreettisesti ja voi selailla sisältöä läpi. Asiakkaalle toimitetaan linkki verkkokoulutukseen ja projekti etenee kommenttikierrokselle. Kommenttikierroksella asiakas voi vapaasti kommentoida verkkokoulutuksen sisältöä. Sisältö toimitetaan asiakkaalle myös Word-tiedostona, johon kommentit kerätään. Kommenttikierros kestää noin 2 viikkoa. Joskus keskustelua sisällöstä käydään Microsoft Teams -palvelussa. [8.]

Asiakkaalta saatujen kommenttien pohjalta tehdään Draft 0.9 -versio, joka toimitetaan asiakkaalle kommentoitavaksi. Tässä vaiheessa tehdään yleensä hyvin pieniä muutoksia. Mahdollisten muutosten jälkeen asiakas hyväksyy lopullisen version. Verkkokoulutus on valmis julkaistavaksi ja asiakkaan käytettäväksi. [8.]

Asiakas voi halutessaan suunnitella verkkokoulutuksen itse. Uudelle asiakkaalle pidetään noin tunnin mittainen käyttöönottokoulutus, jossa opastetaan Builderin käyttöön. Lisäksi Apprix lisää esimerkkiverkkokoulutuksen asiakkaalle, mikä voi auttaa verkkokoulutusten suunnittelussa. Asiakas voi myös halutessaan tilata osallistavan verkkokoulutuksen suunnitteluun ohjaavan luennon Apprixilta. [8.]

Kommunikointi sisällöntuotantoprosessin aikana on tähän mennessä jakautunut useaan eri kanavaan, koska Builderissa ei ole ollut mahdollista kommunikoida muiden käyttäjien kanssa. Kommunikointi voi käydä hyvin raskaaksi ja vaikeaksi seurata, kun keskustelua käydään useassa eri kanavassa. Suunnitellun koulutuksen sisällöstä on jouduttu usein ottamaan kuvankaappauksia, jotta viesti ymmärrettäisiin.

2.3 Yhteistyön mahdollistaminen verkkotyökalujen avulla

Sähköinen yhteistyö (E-collaboration) on yleisesti ottaen yksilöiden välistä yhteistyötä yhteisen tehtävän parissa sähköistä teknologiaa käyttäen. Esimerkkejä sähköisistä yhteistyöteknologioista ovat esim. verkkopohjaiset chat-työkalut,

kokoustyökalut, sähköposti, yhteistyökirjoitustyökalut ja videoneuvottelutyökalut. [9, s. 1.] COVID-19 ja siitä seurannut dramaattinen työkulttuurin muuttuminen etätyöhön on asettanut yhteistyötyökalut kehitystyössä keskeisemmäksi kuin koskaan ennen. Esimerkiksi ohjelmistoa kehitetään harvemmin yhdessä paikassa usean hengen tiimissä. Yhteistyö on haastavaa aikavyöhykkeiden takia ja ilman palavereja kasvotusten. Yhteistyötyökalut mahdollistavat työskentelyn ja tuloksien saavuttamisen yhdessä. [10, s. 7.]

Bertrand Tavernier ja Christof Ebert [11, s. 7–11] kokosivat toimialakiteytyksiä ja -trendejä toimialakyselyn pohjalta, jossa haluttiin selvittää COVID-19 vaikutuksia työhön. Kyselytutkimus suoritettiin vuoden 2020 lopussa, ja siinä oltiin yhteydessä eri toimialojen ammattilaisiin eri puolilta maailmaa. Heiltä kysyttiin mielipidettä työn haasteista, opeista ja tulevaisuuden näkymistä. Kyselyyn osallistui 2 700 ihmistä. Vastaajien mukaan suurin vaikutus pandemialla on ollut verkon välityksellä tapahtuvaan yhteistyöhön. Tärkeämpi havainto oli, että innovaatio ja digitaalinen muutos yhdessä tehokkuuden lisäämisen tarpeen kanssa sijoittuvat korkealle vastauksissa. Kyselyyn vastanneista yli 90 % koki hyötyvänsä verkkopalavereista ja verkkokoulutuksista. [11, s. 7–11.] Seuraavaksi esitellään työkalut, jotka tulivat tutkimuksen mielipidekyselyissä ja haastatteluissa useimmin esille tai koetaan yleisesti olevan suosituimpia. [10, s. 7–12.]

Kommunikointityökaluilla pystytään pitämään yhteyttä käyttäjien välillä. Tällaisia teknologioita ovat chat, videoneuvottelu ja yhteisöpalvelu. Chat-työkalut mahdollistavat viestien vaihtamisen reaaliajassa tai asynkronisesti. Tällaisia palveluita ovat esim. Slack ja Google Chat, joista on tullut ubiikkeja alalla. Videoneuvottelutyökalut kuten Zoom ja WebEx täydentävät chat-työkalua tarjoamalla visuaalisen kohtaamisen kokemuksen. Videoneuvotteluja voidaan aikatauluttaa tai pitää valmistautumatta. Yhteisöpalvelut mahdollistavat kommunikaation, mutta niiden pääpaino on yhteisöjen rakennuksessa ja tiedon tarjoamisessa kuka on kuka organisaation sisällä. [10, s. 12.]

Apprixilla käytetään usein videoneuvotteluja, kun halutaan kommunikoida kasvotusten. Työntekijät tekevät töitä usein etänä ja asiakkaita mennään harvoin tapaamaan paikan päälle. Videoneuvottelut ovat helpoin tapa pitää palavereja ja

esityksiä. Apprixilla videoneuvotteluihin käytetään pääasiassa Google Meet -palvelua. Chat-palveluita Apprixilla käytetään päivittäin, ja ne ovat pääasiallisin kommunikointityökalu. Pääasiallinen chat-palvelu Apprixilla on Slack, mutta osa käyttää myös Microsoft Teams -palvelua. Microsoft Teams -palvelua käytetään usein siksi, että asiakkailla on se käytössä. Slack-palvelulla voidaan kommunikoida yhteisesti eri kanavilla tai kohdentaa viestejä eri henkilöille. Chat-palveluissa ei ole tarkoitus säilyttää tietoa esim. tiedostoja.

Artifaktien hallintatyökaluilla voidaan varastoida, jakaa ja muokata artifakteja [10, s. 12.]. Ohjelmistotuotannossa artifakti on ohjelmistokehityksen sivutuote, jota käytetään kuvaamaan sovelluksen toimintaa, arkkitehtuuria ja suunnitelmaa. Artifaktit ovat etenemissuunnitelmia, joita kehittäjät voivat seurata ohjelmistokehityksen ajan. Tyypillisiä artifakteja ovat kuvat, diagrammit ja prototyypit. [12.] Näihin työkaluihin kuuluu virtuaaliset valkotaulut, kaaviotyökalut ja jaettujen tiedostojen editointityökalut. Kaikissa näissä kohdennettu ryhmä voi tuottaa jaettua sisältöä kuten Google Docs -työkalussa. Kaikki työkalut tukevat asynkronista yhteistyötä, mutta pääpaino on reaaliaikaisessa yhteistyössä. Tiedonhallintatyökaluissa, kuten Confluence, voidaan hallita jatkuvasti resurssivarastoa, jossa voidaan jakaa kaikenlaista tietoa esim. projektista. [10, s. 12.]

Apprixilla on verkkolevy, jossa säilytetään erilaisia dokumentteja eri projekteista. Esimerkiksi sisällöntuottajat säilyttävät verkkolevyllä sisällöntuotantoon liittyviä materiaaleja. Jokaista kehitysprojektia varten verkkolevylle tehdään oma kansio, joka sisältää projektin määrittelyyn ja toteutukseen liittyviä dokumentteja. Dokumentteja voidaan muokata yhteistyössä ja yhtä aikaa, mikä on tärkeää projektityössä. Verkkolevyä käytetään aina, kun on tarkoitus dokumentoida jotain tärkeää ja pysyvää.

Tehtävienhallintatyökaluilla organisoidaan työtä ja voidaan seurata edistymistä. Tehtävienhallintateknologia käsittää projektin suunnittelu- ja hallintatyökalut, säilytyspaikat ja tilannekojelaudat. Projektin suunnittelu- ja hallintatyökaluja ovat esim. Asana ja Trello. Säilytyspaikkoja ovat esim. Github ja Gitlab, jotka tarjoavat keskitetyn paikan säilyttää kehityksen artifaktoja ja kontrolloidusti päivittää niitä.

Tilannekojelaudat tarjoavat tyypillisesti näkymiä tiimin päivittäisestä edistymisestä esittämällä avaintilastoja kehityksestä. [10, s. 12.]

Apprixilla käytetään tehtävienhallintaan Teamwork-palvelua. Teamwork-palveluun kirjataan kaikki projektiin liittyvät tehtävät, tunnit ja edistyminen. Projekteihin liittyvät keskustelut tulee pääosin käydä Teamwork-palvelun kommenteissa, koska se on pääasiallinen tehtävienhallintatyökalu. Koodin säilytykseen käytetään Github-palvelua.

3 Apprix Builder

Builder on verkkokoulutusalusta, jolla voidaan tehdä, jakaa ja seurata verkkokoulutuksia. Builderia jaetaan SaaS-jakelumallilla. Builderin asiakkaita on erilaiset yritykset, yhdistykset ja liitot. Builderin käyttäjiä ovat asiakkaan työntekijät, jotka tulevat eri työtehtävistä. Käyttäjät ovat asiakkaan nimeämiä henkilöitä, joille luodaan tunnukset Builderiin. Käyttäjien tehtävä on luoda, seurata ja ylläpitää verkkokoulutuksia. Builderin käyttäjiä kutsutaan fasilitaattoreiksi. Verkkokoulutuksia suorittavat henkilöt eli vastaajat eivät ole fasilitaattoreita. Heillä ei ole pääsyä verkkokoulutusalustalle, vaan ainoastaan verkkokoulutukseen.

Yksinkertaisuudessaan Builderilla voidaan luoda verkkokoulutus, joka julkaistaan verkossa. Julkaistun verkkokoulutuksen voi jakaa esim. työntekijöille, jotka suorittavat verkkokoulutuksen. Suorituksesta jää merkintä Builderin tilastoihin, joita fasilitaattorit voivat seurata. Builderin perustoiminnallisuuden lisäksi asiakkailla on mahdollisuus hankkia itselleen Builderin lisäominaisuuksia eli featureja. Featureilla voidaan parantaa käyttökokemusta ja mahdollistaa asioita kuten verkkokoulutuksen eri kirjautumistavat. Osa featureista on ilmaisia ja osa maksullisia. Featureja hallitsee Apprixin super-käyttäjät, joilla on oikeus muokata näitä asetuksia.

3.1 Builderin fasilitaattorityökalut

Fasilitaattoreille voidaan antaa erilaisia käyttöoikeuksia. Käyttöoikeuksia voidaan rajoittaa esimerkiksi fasilitaattorin pääsyä koulutuksen tilastoihin ja

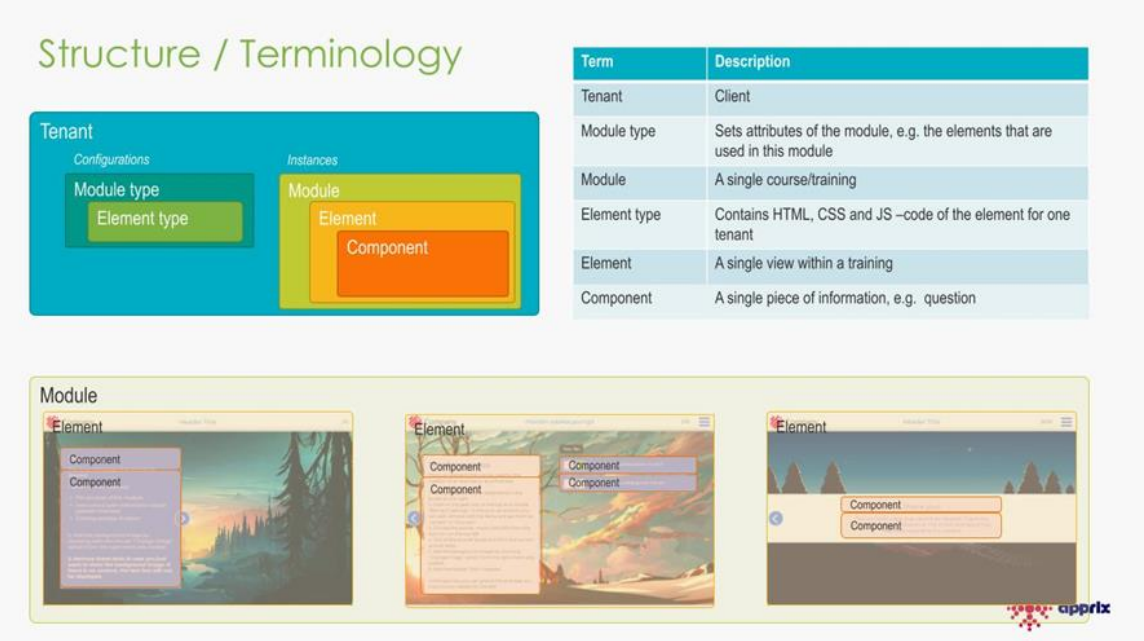
fasilitaattorin mahdollisuutta muokata koulutusta. Käyttöoikeuksia voidaan määritellä koulutuskohtaisesti. Fasilitaattorille voidaan antaa oikeus muokata muiden fasilitaattorien oikeuksia.

Koulutuksia voi muokata, lisätä ja poistaa vapaasti. Koulutuksesta voi muokata monia asioita riippuen asiakkaan lisäominaisuuksista eli featureista. Muokattavia asioita ovat esim. kirjautumistavat ja koulutuksen pisteet. Fasilitaattorit näkevät myös, kuka on tehnyt muutoksia koulutukseen ja milloin. Koulutusta voi myös esikatsella halutessaan. Koulutuksen voi jakaa julkaisemalla koulutuksen ja jakamalla koulutuksen linkin. Builder kerää koulutusten suoritukset, joita fasilitaattorit voivat tarkastella. Suoritukset voidaan myös lähettää asiakkaan haluamiin järjestelmiin.

Koulutuksen voi ladata myös SCORM-pakettina, jolloin sen voi jakaa muissa LMS-järjestelmissä. SCORM (Sharable Content Object Reference Model) on yleisin verkkokoulutusstandardi. SCORM-standardi määrittelee koulutuksen sisällön teknisen rakenteen ja tavan, jolla LMS-järjestelmä toimittaa tämän sisällön. SCORM-paketti on kokoelma tiedostoja, joita tarvitaan sisällön lataamiseen ja noudattaa SCORM-standardia. SCORM-paketti pystyy kommunikoimaan LMS-järjestelmän kanssa, ja käyttäjä pystyy näin lataamaan verkkokoulutuksen sisällön LMS-järjestelmässä. [13 s. 134–135.]

3.2 Yksittäisen verkkokoulutuksen rakenne

Builderiin on kehitetty terminologia kuvastamaan eri osia Builderin rakenteessa. Builderin eri osilla on tietynlainen hierarkia toisiinsa. Osilla on omat tehtävänsä lopullisessa verkkokoulutuksessa (kuva 2).



Kuva 2. Builderin terminologia ja hierarkia.

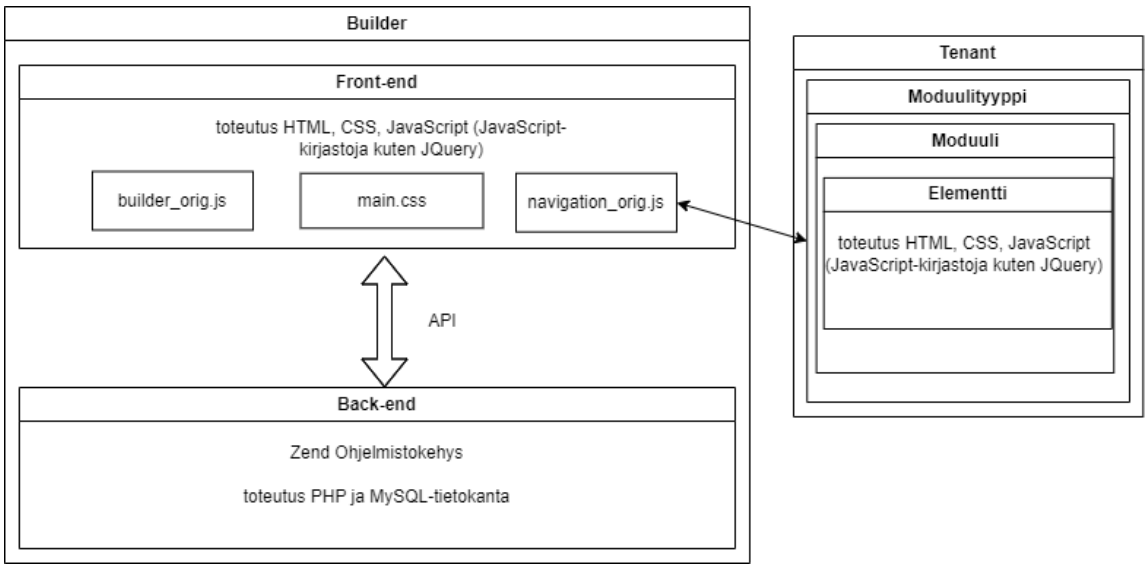
Kuvasta 2 nähdään Builderin rakenne ja hierarkia. Builderissa jokaisella asiakkaalla on oma asiakasympäristö eli tenant. Jokaisella tenantilla on omat käyttäjät, asetukset ja koulutukset. Tenantilla on moduulityypejä, jotka ovat pohjia koulutuksen rakentamiselle. Moduulityypille voidaan suunnitella esim. oma ulkoasu. Tenantilla voi olla useampia moduulityyppejä. Tämä tarkoittaa, että yrityksen eri osastoille voidaan luoda omat moduulityypit. Moduulityypit sisältävät elementtityyppejä, jotka ovat pohjia, joista luodaan elementtejä. Elementtityyppi koostuu HTML-, JavaScript- ja CSS-koodista, joka kopioituu niistä luotavaan elementtiin.

Moduulityypin koodi on erillään Builderin koodista, eikä Builderilla ole siihen pääsyä. Moduulityypit ovat ikään kuin erillisiä toteutuksia. Elementtityyppien määrä vaihtelee paljon eri moduulityyppien välillä. Alussa moduulityypille laitetaan peruspaketti elementtityyppejä, jotka ovat kaikille samat. Tarjolla on kymmeniä erilaisia elementtityyppejä, joita asiakas voi hankkia korvausta vastaan. Yksittäistä verkkokoulutusta kutsutaan moduuliksi. Moduulit koostuvat elementeistä, jotka ovat yksittäisiä sivuja koulutuksessa. Elementit koostuvat sisällöstä eli komponenteista. Komponentit ovat usein muokattavia kuten taustakuvia,

videoita tai tekstiä. Moduulityypit ja elementtityypit ovat konfiguraatioita, joiden pohjalta luodaan ilmentymiä (moduuli, elementti ja komponentti).

3.3 Builderin tekninen arkkitehtuuri

Builderilla on kaksi palvelinta: tuotanto- ja testipalvelin. Testipalvelimella jokaisella kehittäjällä on oma kehitysympäristönsä sekä yksi yleinen testiympäristö. Builderin testipalvelimen ympäristöt käyttävät testitietokantaa ja tuotantopalvelimen tuotantoympäristö käyttää tuotantotietokantaa. Builderin koodi muodostuu backendista ja frontendista (kuva 3).



Kuva 3. Builderin arkkitehtuuri.

Kuvassa 3 nähdään Builderin arkkitehtuurirakenne. Backend on toteutettu MySQL-tietokannalla ja PHP-ohjelmointikielellä. Frontendissa käytetään HTML-, CSS- ja JavaScript-ohjelmointikieliä. Frontendissa käytetään useita JavaScript-kirjastoja kuten jQuerya. Backendissa käytetään avoimen lähdekoodin Zend-ohjelmistokehystä. Zend-ohjelmistokehys noudattaa MVC-suunnittelumallia (Model-View-Controller). Builderin käyttöliittymää ohjaa builder_orig.js-JavaScript-tiedosto. Siellä on Builderin frontendin toimintalogiikka. Moduulin esikatselutila käyttää tenantin moduulityypin koodia sekä Builderin navigation_orig.js-tiedostoa. Navigation_orig.js-tiedoston tehtävänä on välittää tietoa Builderin ja

tenantin moduulityypin koodin välillä. Navigation-JavaScript-tiedosto ladataan moduulityypin koodissa eli jokaisen elementin latauksen yhteydessä sekä moduulin esikatselunäkymässä että elementin muokkausnäkymässä. Frontendissa tehdään AJAX-kutsuja ohjelmointirajapinnalle (Application Programming Interface) esim. tiedon tallentamista ja hakemista varten. Builderin tyylit löytyvät yhdestä main.css-tiedostosta.

Ohjelmointirajapinta

Ohjelmointirajapinta eli API on joukko toimintoja ja sääntöjä, jotka ovat sovelluksen sisällä. Ohjelmointirajapintaa voidaan kuvailla yksinkertaisena sopimuksena sitä tarjoavan sovelluksen ja muiden kohteiden, kuten kolmannen osapuolen ohjelmistojen tai laitteiden, välillä. [14.] Ohjelmointirajapinta mahdollistaa eri ohjelmistokomponenttien kommunikaation ja datan siirron. Ohjelmointirajapinta toimii jakamalla dataa sovellusten, systeemien ja laitteiden välillä. Tämä tapahtuu pyyntö-vastaussyklin kautta. Käyttäjä tekee pyynnön sovelluksessa esim. painamalla lähetä-nappia. Pyyntö lähetetään ohjelmointirajapinnalle, joka hakee dataa ja palauttaa sen käyttäjälle. [15.]

Ohjelmointirajapintaa voidaan verrata ravintolaan. Ravintolan asiakas on kuin käyttäjä, joka kertoo tarjoilijalle, mitä hän haluaa. Tarjoilija on kuin ohjelmointirajapinta, joka ottaa asiakkaan tilauksen vastaan ja välittää yksinkertaiset ohjeet keittiölle. Keittiön henkilökunta on kuin ohjelmointirajapinnan palvelin, koska se tekee tilauksen ohjeiden mukaan ja antaa tarjoilijalle. Tarjoilija vie tilauksen lopulta asiakkaalle. [15.]

Ohjelmointirajapinnalla on eri komponentteja kuten API-asiakas ja API-päätepiste. API-asiakas on vastuussa pyyntöjen kokoamisesta vastauksena käyttäjän toimiin ja lähettämisestä asianmukaiseen API-päätepisteeseen. Päätepisteet ovat URI-osoitteita, jotka antavat pääsyn tiettyihin tietokannan resursseihin. Esimerkiksi jos käyttäjä haluaa nähdä kaikki verkkokaupan tuotteet, API-asiakas tekee GET-pyynnön /products-päätepisteeseen. [15.]

Ohjelmointirajapintaa käytetään JavaScript-tiedostossa lähettämällä AJAX-kusuja. AJAX (Asynchronous JavaScript and XML) on verkkokehitystekniikka, jossa verkkosovellus hakee sisältöä palvelimelta tekemällä sinne HTTP-pyynnön ja näyttämällä vastauksena saatua sisältöä ilman sivun uudelleen latausta. Alun perin AJAX toteutettiin XMLHttpRequest-rajapinnalla, mutta nykyisin suositellaan Fetch-rajapinnan käyttöä. Fetch-rajapinta on sopivampi moderneihin verkkosovelluksiin, koska se on tehokkaampi, joustavampi ja integroituu paremmin perusverkkosovellusteknologioihin. [16.] Tässä työssä käytettiin XMLHttpRequest-rajapintaa, koska Builderissa käytetty JQuery-JavaScript-kirjasto käyttää sitä.

XMLHttpRequest-rajapintaa käytetään JavaScript-kielessä luomalla XMLHttpRequest-objekti [17]. Tämän jälkeen kutsutaan XMLHttpRequest-objektin metodia open, jossa määritellään resurssin URL-osoite, johon pyyntö toimitetaan. Lisäksi siinä määritellään HTTP-pyynnön metodi. [18.] Lopuksi kutsutaan XMLHttpRequest-objektin send-metodia, jolla HTTP-pyyntö välitetään. Kun transaktio on tapahtunut XMLHttpRequest-objekti sisältää vastauksen HTTP-vastauksen palvelimelta, jota voidaan käyttää verkkosivun päivittämiseen. HTTP-vastaus sisältää HTTP-statuksen, joka kertoo, onko HTTP-pyyntö suoritettu onnistuneesti. [17.] XMLHttp-objektin send-metodi voi sisältää tietoa, jota lähetetään palvelimelle HTTP-pyynnön yhteydessä [19].

HTTP-pyynnön metodi viittaa siihen, mikä on pyynnön tarkoitus ja mitä asiakas odottaa onnistuneena tuloksena. HTTP-pyynnön kohdetta kutsutaan resurssiksi. Useimmat resurssit tunnistetaan Uniform Resource Identifier (URI) -tunnuksella. [20.] Resurssi on jokin asia johon URI-tunnus osoittaa. Resurssi on abstrakti käsite. Esimerkiksi URL-osoite http://esimerkki.com/käyttäjät viittaa resurssiin nimeltä käyttäjät, joka palauttaa esityksen käyttäjistä. Resurssi ei koskaan muutu, vaan sen esitykset muuttuvat. [21.]

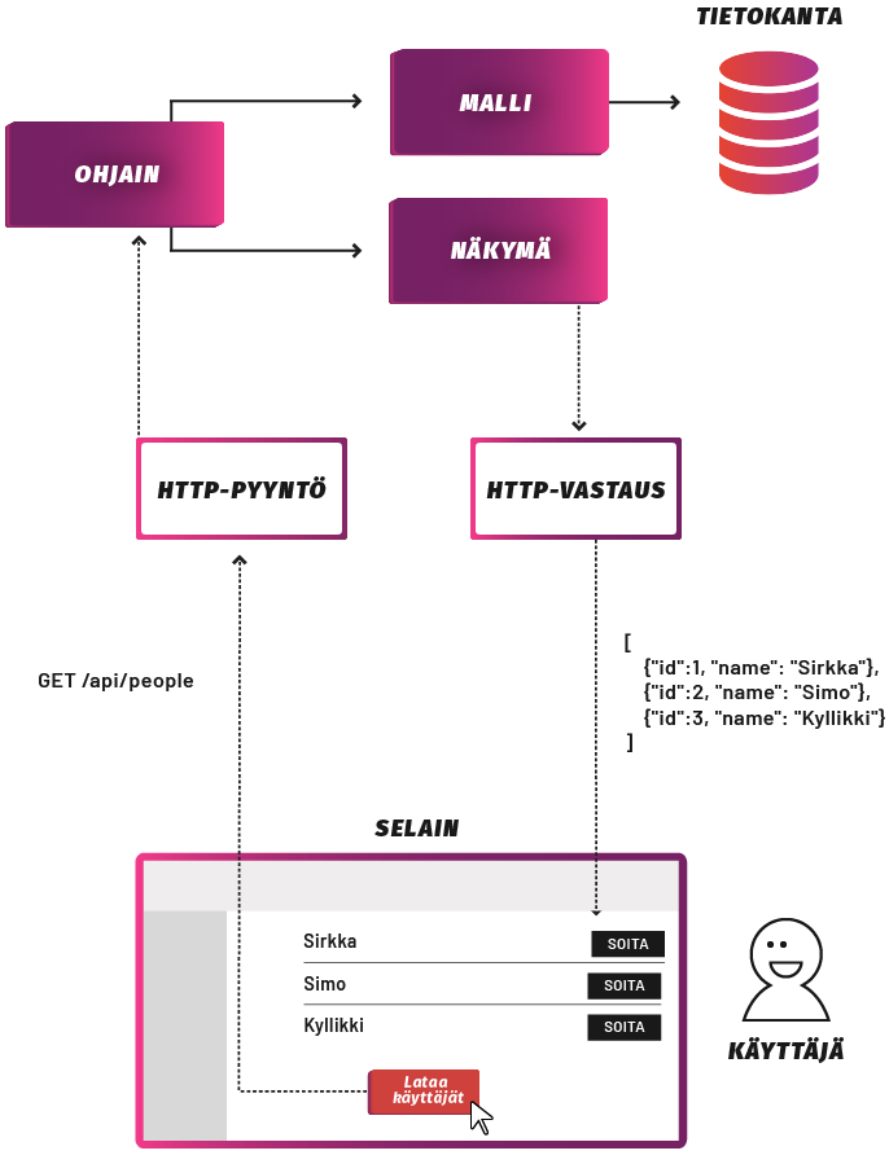
HTTP-pyynnön metodeja ovat esim. GET, POST, PUT ja DELETE. GET-metodi pyytää nykyisen valitun esityksen siirtoa kohderesurssille. POST-menetelmä pyytää, että kohderesurssi käsittelee pyynnön sisältämän esityksen resurssin oman semantiikkansa mukaisesti. POST-metodia käytetään esim. luomaan uusi

resurssi, jota palvelin ei ole vielä tunnistanut tai viestin lähettämiseen ilmoitustaululle, uutisryhmään, postituslistalle, blogiin tai vastaavaan artikkeliryhmään. PUT-menetelmä pyytää, että kohderesurssin tila luodaan tai korvataan pyyntöviestin sisällön sisällä olevan esityksen määrittelemällä tilassa. DELETE-menetelmä pyytää palvelinta poistamaan yhteyden kohderesurssin ja sen nykyisen toiminnan välillä. [20.]

HTTP-status on kolminumeroinen koodi, joka kuvaa HTTP-pyynnön tuloksen ja vastauksen semantiikan, mukaan lukien sen, onnistuiko HTTP-pyyntö ja mitä sisältöä siihen sisältyy. Numerolla 2 alkava statuskoodi kertoo, että pyyntö on onnistunut. Muilla numeroilla alkavat statuskoodit kertovat, että pyyntö on kesken tai epäonnistunut. [20.]

MVC-suunnittelumalli

MVC-suunnittelumalli (Model-View-Controller) on ohjelmistosuunnittelumalli. Se rakentuu kolmen pääkomponentin välisen yhteyden ympärille. Nämä komponenttityypit ovat malli (Model), näkymä (View) ja ohjain (Controller). Malli sisältää sovelluksen toimintalogiikan. Se voi sisältää esim. kuinka sovellus varastoi dataa tai käyttää kolmannen osapuolen palveluita. Koodia, jolla halutaan olla yhteydessä tietokantaan, säilytetään mallissa. Näkymä pitää sisällään käyttöliittymän elementtejä. Tämä voi sisältää HTML-, CSS- ja JavaScript-tiedostot. Näkymä pitää sisällään kaiken, mitä käyttäjä näkee tai mihin käyttäjä voi vuorovaikuttaa. Ohjain yhdistää mallin ja näkymän toisiinsa. Ohjain eristää mallin sisältämän toimintalogiikan näkymän käyttöliittymäelementeistä ja käsittelee, miten sovellus vastaa käyttäjän vuorovaikutukseen näkymässä. Ohjain ottaa ensin vastaan käyttäjän pyynnön ja toteuttaa tarvittavan mallin sekä näkymän, vastatakseen pyyntöön. [22, s. 1–2.] Kuvassa 4 on esimerkki, jossa käyttäjä painaa selaimen käyttöliittymästä Lataa käyttäjät -nappia.



Kuva 4. Esimerkki MVC-suunnittelumallista [23].

Kuvassa 4 Nappia painamalla lähtee GET-pyyntö /api/people-päätepisteeseen. Pyyntö ohjautuu ohjaimelle (Controller), joka välittää sen mallille. Malli tekee tietokantahaun ja palauttaa siitä saadun tuloksen takaisin ohjaimelle. Ohjain välittää tiedon eteenpäin näkymälle, joka päivittää selainkäyttöliittymään listan ihmisistä tietokannasta haetuilla tiedoilla.

Zend-ohjelmistokehys

Zend-ohjelmistokehystä käytetään rakentamaan PHP-verkkosovelluksia. Zend sisältää kaiken, jotta voidaan rakentaa valmis verkkosovellus. Zend on hyvin joustava, ja siinä voidaan käyttää niitä palasia ohjelmistokehyksestä, joita tarvitaan omassa sovelluksessa. Zend sisältää komponentteja, joiden avulla voidaan rakentaa moderneja sovelluksia. Komponentit voidaan jakaa kuuteen yläkategoriaan. Yläkategoriat ovat MVC-komponentit, todentamis- ja pääsyoikeuskomponentit, kansainvälistämiskomponentit, sovellusten välisen kommunikaation komponentit, verkkopalvelukomponentit ja ydinkomponentit. Jokainen komponentti sisältää lukuisia luokkia esim. Zend_Config. [24.]

MVC-komponentit tarjoavat sovellukselle MVC-systeemin. Todennus- ja pääsyoikeuskomponentit vastaavat nimensä mukaan sovelluksen käyttäjän todentamisesta ja pääsyoikeuksista. Yleensä käyttäjä todennetaan tunnuksilla kuten käyttäjätunnus ja salasana parilla, mutta se voi yhtä hyvin olla sormenjälki. Pääsyoikeuksilla pidetään huolta, että todennettu käyttäjä pääse tekemään vain hänelle tarkoitettuja toimintoja. Kansainvälistämiskomponentit mahdollistavat toimintoja, joilla paikallistaa sovellus käyttäjien mukaan. Sovelluksessa voidaan näin vaihtaa esim. kieltä, valuuttaa, aikaa ja päivämäärää käyttäjän olinpaikan mukaan. [24.]

Sovellusten välisen kommunikaation komponentit mahdollistavat datan lukemisen muilta verkkosivuilta. Muilta verkkosivuilta ja palveluista voidaan kerätä dataa ja näyttää omalla verkkosivulla. Verkkopalvelukomponentit tarjoavat laajan joukon toimintoja, jotka mahdollistavat muiden palveluiden käytön sovelluksessa. Sovellus voi käyttää esim. Googlen, Yahoon ja Amazonin rajapintoja. Google tarjoaa oman rajapintansa kautta pääsyn esim. Google-kalenteriin ja Youtubeen. Ydinkomponentit sisältävät komponentteja, jotka eivät kuulu muihin kategorioihin. Muihin kategorioihin kuuluu komponentteja, jotka liittyvät esim. välimuistiin, hakutoimintoihin, PDF:n luomiseen ja sähköpostin lähettämiseen. [24.]

4 Tekninen toteutus

4.1 Määrittely

Työn määrittely aloitettiin tuotekortin suunnitelulla. Tuotekortti on yrityksen sisäinen dokumentti, jossa esitellään esim. työn nimi, työryhmä, tarkoitus ja julkaisussa huomioitavat asiat. Tuotekortin jälkeen tehtiin selvitystyötä eri JavaScript-kirjastoista ja kommentointityökaluista. Tarkoituksena oli selvittää, voisiko työssä hyödyntää olemassa olevia ratkaisuja ja miten muut kommentointityökalut toimivat. Selvityksessä kävi ilmi, että ilmaiset JavaScript-kirjastototeutukset olivat rajoittuneita, eikä niitä saisi räätälöityä omiin tarpeisiin. Maksullisia palveluita löytyi useampia, mutta ne olivat kalliita, eikä niitä olisi saanut liitettyä osaksi Builderia. Lopputuloksena päädyttiin kehittämään oma kommentointityökalu.

Ensimmäiset viralliset määrittelyt tehtiin aloituspalaverissa, jossa päätettiin työn nimi ja ideoitiin kommentointityökalun ominaisuuksia. Palaverissa päätettiin myös työryhmä ja työtehtävät. Projektipäälliköksi valittiin työn toteuttaja. Projektipäällikkö vastaa työn edistymisestä ja osallistuu työn kaikkiin vaiheisiin aina suunnittelusta testaukseen. Ensimmäisessä palaverissa ideointiin, mitä kommentointityökalu voisi olla. Ideoita tuli paljon, eikä lopullisia päätöksiä halutuista ominaisuuksista päästy tekemään. Palaverista tehtiin muistiinpanot, joiden pohjalta halutut ominaisuudet päätettiin. Myöhemmin pidettiin uusi palaveri, jossa päätettiin lopulliset ominaisuudet. Tämän jälkeen tehtiin toiminnallinen määrittely, jossa selostettiin, kuinka kommentointityökalu toimii ja priorisoitiin halutut ominaisuudet. Dokumenttiin lisättiin myös jatkokehitysideat.

Työn alussa tehtiin projekti Teamwork-projektinhallintatyökaluun. Teamwork-ohjelmaa käytetään työn organisointiin. Työ jaoteltiin Teamwork-projektiin pienempiin osiin. Näin pystyttiin helpommin seuraamaan, kuinka työ etenee. Työtä pystyttiin myös aikatauluttamaan ja seuraamaan työkalun avulla.

Kommentointityökalusta päätettiin tehdä Builderin feature, jolloin featuren käyttö voidaan määritellä tenant-kohtaisesti. Toimeksiantajan kanssa sovittiin, että kommentointityökalu tulee oletuksena käyttöön kaikille tenanteille.

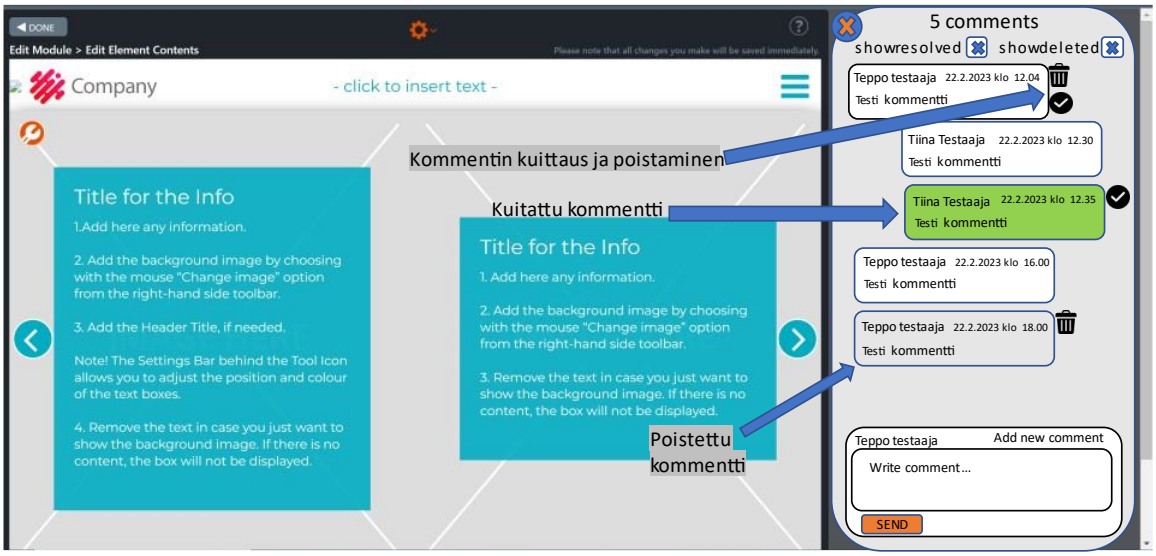
Kommentointityökalua voidaan käyttää kahdessa eri näkymässä riippuen fasilitaattorin käyttöoikeuksista. Kommentointityökalua käytetään elementin muokkausnäkymässä sekä moduulin esikatselutilassa. Kommentointityökalu toimii samalla tavalla molemmissa näkymissä.

Fasilitaattorit, joilla on moduulin muokkausoikeus, voivat käyttää kommentointityökalua moduulin muokkausnäkymässä. Muut fasilitaattorit sekä anonyymit käyttäjät voivat käyttää kommentointityökalua vain moduulin esikatselutilassa. Anonyymit käyttäjät ovat henkilöitä, jotka tulevat moduuliin esikatselutilaan heille jaetun linkin kautta, eivätkä ole kirjautuneena Builderiin. Tässä tapauksessa käyttäjää ei voida tunnistaa. Kuvassa 5 kommentointityökalu anonyymin käyttäjän näkymässä.

Kuva 5. Rautalankamalli, jossa näkyy moduulin esikatselunäkymä anonyymin käyttäjän näkökulmasta.

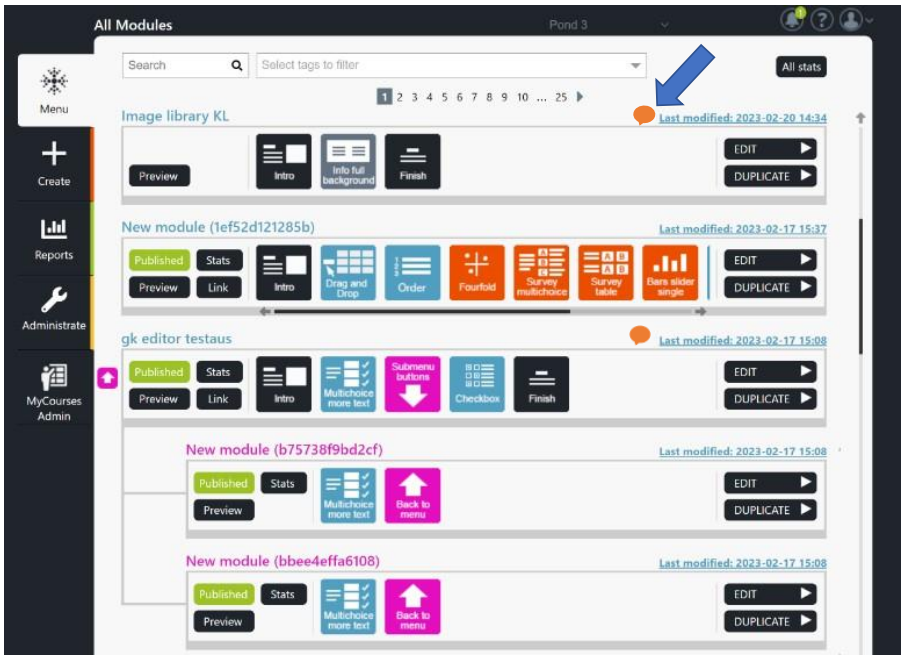
Kuvassa 5 näkyy moduulin yksittäinen elementti. Kommentointityökaluikkuna avautuu elementin päälle oikeaan reunaan. Anonyymi käyttäjä voi syöttää nimimerkin kommenttiinsa. Liitteessä 1 on nähtävissä kommentointityökalun lopullinen käyttöliittymä moduulin esikatselunäkymässä. Liitteestä nähdään, että kommentointityökalun rakenne on pysynyt hyvin samanlaisena. Anonyymin

nimimerkki lisätään kommenttiin samasta paikkaa ja kommentointityökalu avautuu samaan kohtaan kuin rautalankamallissa. Fasilitaattorit, joilla on moduulin muokkausoikeus, voivat poistaa kommentin, palauttaa poistetun kommentin, kuitata kommentin, poistaa kuittauksen kommentista ja näkee poistetut kommentit (kuva 6). Kaikki käyttäjät voivat kommentoida ja nähdä kuitatut sekä kuittaamattomat kommentit.



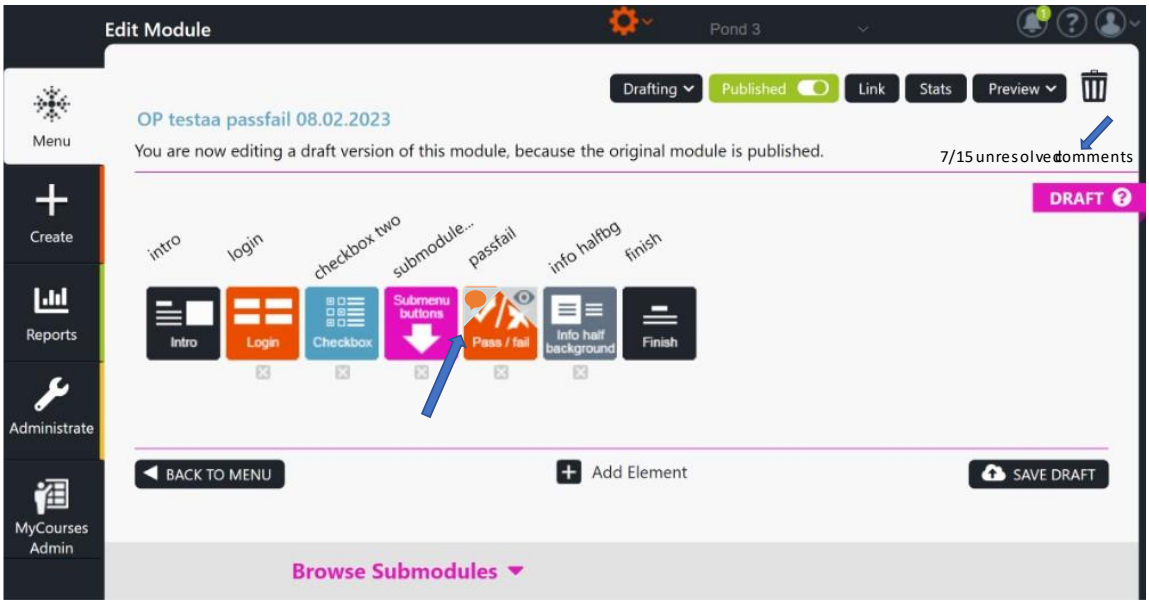
Kuva 6. Rautalankamalli Builderin elementin muokkausikkunasta, jonka viereen on avattu kommentointityökaluikkuna.

Kuvassa 6 nähdään kommentointityökalun kommentin eri tilat ja toiminnot, joita kommentteihin voi kohdistaa. Kommentit on sidottu näytettävään elementtiin. Kommentointityökalu näyttää yläreunassa, kuinka monta kuittaamatonta kommenttia kyseisessä elementissä on. Poistetut ja kuitatut kommentit näytetään työkalussa toisistaan eroavasti. Kuitattu kommentti näytetään vihreällä ja poistettu harmaalla pohjalla. Kommentti voi sisältää vain tekstiä. Näytettäviä kommentteja voi suodattaa. Fasilitaattorit, joilla on moduulin muokkausoikeus, voivat nähdä myös poistetut kommentit. Lisäksi he voivat kuitata ja poistaa kommentin sen kohdalla näkyvistä ikoneista. Listatun kommentin kohdalla näkyy kommentin poistamisnappi (roskakori-ikoni) ja kuittausnappi (kuittausikoni). Builderin moduulilistausnäkymässä on listattu kaikki tenantin moduulit (kuva 7).



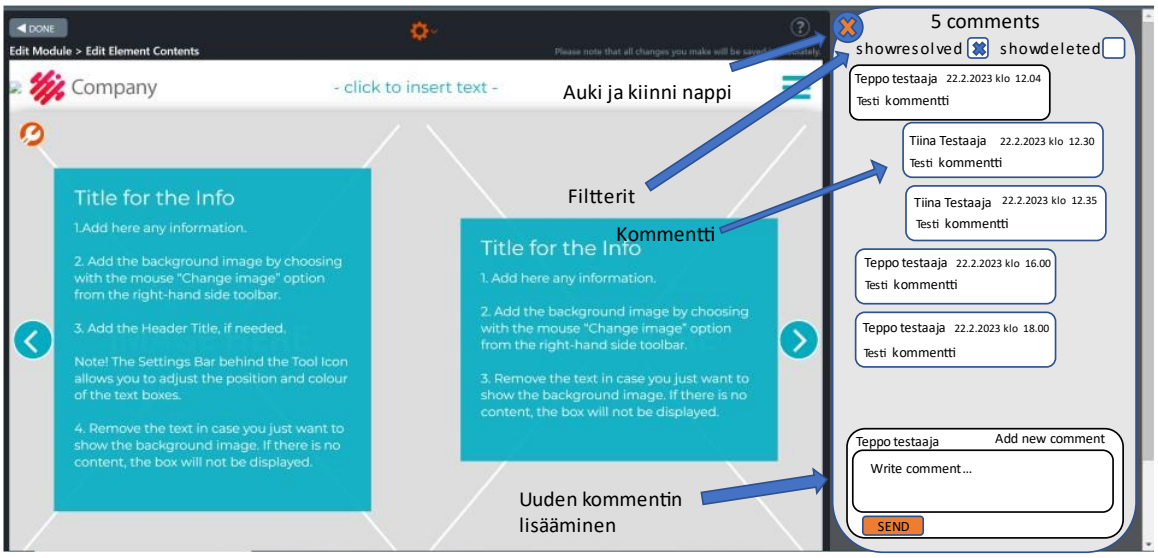
Kuva 7. Rautalankamalli Builderin moodulilistausnäkymästä.

Kuvassa 7 nähdään Builderin moduulilistausnäkymä, jossa näytetään moduulin kohdalla puhekuplaikonilla, jos moduulissa on vähintään yksi kuittaamaton kommentti. Liitteessä 2 on nähtävissä Moduulilistausnäkymän lopullinen käyttöliittymä. Liitteestä voidaan nähdä, että vain puhekuplaikonin väri on muuttunut rautalankamallista. Fasilitaattori pääsee muokkaamaan moduulia klikkaamalla moduulin kohdalla näkyvää EDIT-nappia. Moduulin muokkausnäkymässä käyttäjä voi esim. muokata moduulin rakennetta, asetuksia ja julkaisutilaa (kuva 8).



Kuva 8. Rautalankamalli moduulin muokkausnäkymästä.

Kuvassa 8 on nähtävissä moduulin rakenne, joka koostuu rivillä näkyvistä elementeistä. Puhekuplaikoni elementin päällä merkitsee, että elementissä on vähintään yksi kuittaamaton kommentti. Näkymän oikeaan yläkulmaan on merkitty kuittaamattomien kommenttien lukumäärä ja kaikkien kommenttienlukumäärä. Fasilitaattori pääsee muokkaamaan yksittäistä elementtiä klikkaamalla rivillä näkyvää elementtikuvaketta. Liitteessä 3 on nähtävissä moduulin muokkausnäkymän lopullinen käyttöliittymä. Liitteessä elementin kohdalle lisättävä puhekuplaikonin väri on muuttunut, ja puhekuplaikoniin on lisätty kuittaamattomien kommenttien lukumäärä. Moduulista näytetään vain kuittaamattomien kommenttien kokonaislukumäärä oikean reunan puhekuplaikonissa. Elementin muokkausikkunassa fasilitaattori voi muokata elementin asetuksia ja komponentteja kuten kuvia ja tekstejä (kuva 9).



Kuva 9. Rautalankamalli, jossa näkyy Builderin elementin muokkausikkuna ja viereen on avattu kommentointityökaluikkuna.

Kuvasta 9 nähdään kommentointityökalun eri osia. Kommentointityökalun voi avata oikeaan yläkulmaan ilmestyvällä napilla, joka tulee näkyville muokkaamalla elementtiä tai esikatselemalla koulutusta. Napin alapuolella on kommenttien suodatusvaihtoehdot. Suodattimien alapuolella on annetut kommentit listattuna uusimmasta vanhimpaan. Uuden kommentin voi lisätä kommentointityökalun alareunasta, jossa on yksinkertainen tekstikenttä ja lähetysnappi. Kommentointityökalu tunnistaa kirjautuneen fasilitaattorin. Kommentointityökalu luodaan sen perusteella, mitkä käyttöoikeudet fasilitaattorilla on. Kommentista näytetään kommentoijan nimi, kommentointiaika ja kommentin sisältö. Käyttäjä voi jättää uuden kommentin tai vastata toiseen kommenttiin. Liitteessä 4 on nähtävissä kommentointityökalun lopullinen käyttöliittymä. Liitteestä nähdään useita muutoksia: Filtterit on muutettu kytkimiksi. Kuittaamattomat kommentit näytetään puhekuplaikonissa kommentointityökalun vasemmassa yläkulmassa. Kommenttiin voi vastata täyttämällä vastauslomakeen, joka avautuu kommentin vastausikonia painamalla. Kommentin poistamis- ja kuittaamisnapit avautuvat näkyville painamalla kommentin oikean reunan kolmea pistettä. Kommentin vasempaan yläkulmaan on lisätty kommentin lisääjän nimikirjaimet. Kuitatut kommentit

näkyvät vihreällä kuittausikonilla. Kommentin lähettämislomakkeelle on lisätty peruuttamisnappi, jolla lomake menee piiloon.

4.2 Suunnittelu

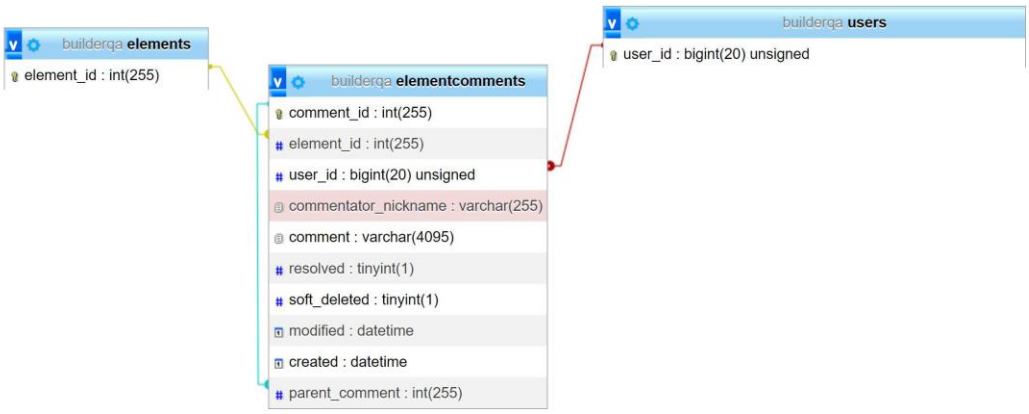
Käyttöliittymäsuunnittelu

Työn suunnittelu aloitettiin käyttöliittymän suunnittelusta, joka tehtiin yhteistyössä toimeksiantajan käyttöliittymäsuunnittelijan kanssa. Ajatuksena oli tehdä ensin itsenäisesti rautalankamallit, jonka jälkeen käyttöliittymäsuunnittelija tekee lopullisen suunnitelman. Suunnittelu oli hyvä aloittaa käyttöliittymästä, jotta ei jouduttaisi odottelemaan lopullista käyttöliittymäsuunnitelmaa. Suunnittelun tueksi otettiin kuvankaappauksia muista palveluista, joissa on kommentointityökalu. Tämän jälkeen suunniteltiin rautalankamallit PowerPointilla. Jokaisesta näkymästä tehtiin oma kuva, perustuen havaintoihin muiden palveluiden kommentointityökaluista. Käyttöliittymäsuunnittelija teki lopullisen suunnitelman, perustuen rautalankamalleihin ja kuvankaappauksiin muista palveluista.

Tekninen suunnittelu

Tekninen suunnittelu aloitettiin Builderin backend-koodiin tutustumisella. Backendiin tutustuttiin itsenäisesti ja kollegan opastuksella. Tämän pohjalta tehtiin alustavaa suunnitelma sitä, mihin tiedostoihin tehtäisiin muutoksia ja miten rakennettaisiin tietokantaan lisättävä taulu. Suunnittelun lähtökohtana oli tehdä mahdollisimman erillinen toteutus, jotta sitä voisi hyödyntää jatkossa muualla sekä hyödyntää mahdollisimman paljon Builderin olemassa olevaa toteutusta.

Itsenäisen suunnitellun jälkeen pidettiin palaveri Builderin pääsuunnittelijan kanssa. Pääsuunnittelijan kanssa tehtiin lopulliset suunnitelmat teknisestä toteutuksesta. Palaverissa tietokantasuunnitelmaan tehtiin esim. nimimuutoksia ja tyyppimuutoksia. Lisäksi tarkennettiin, mihin tiedostoihin tehdään muutoksia ja minne tehdään uusia tiedostoja. Muutokset kirjattiin tekniseen suunnitelmaan. Tietokantaan tehtiin suunnitelman pohjalta uusi taulu elementcomments kommenttien säilömiseen (kuva 10).



Kuva 10. Kommentointityökalua varten tehty tietokantarakenne.

Kuvassa 10 nähdään elementcomments-taulun rakenne ja viittaukset muihin tauluihin. Elementcomments-taulun element_id viittaa elements-taulun element_id-kenttään, koska kommentti liittyy aina johonkin elementtiin. Fasilitaattoreiden tiedot löytyvät users-taulusta. Elementcomments-taulun user_id viittaa users-taulun user_id-kenttään, koska näin fasilitaattoreiden kommentit voidaan yhdistää toisiinsa. Elementcomments-taulun parent_comment viittaa saman taulun comment_id-kenttään, koska näin voitiin luoda rajoite.

Kun poistetaan kommentti tietokannasta, kommentin vastausten viittaus tähän kommenttiin poistetaan. Jos fasilitaattori tai elementti poistetaan, jolla on kommentteja, ne poistetaan tietokannasta. Kommentteja ei oikeasti poisteta, kun fasilitaattori poistaa kommentointityökalulla kommentin, vaan kommentti säilyy elementcomments-taulussa ja sen soft_deleted-kentän arvoksi muuttuu yksi. Elementcomments-taulun comment_id-, element_id- ja user_id-kentät indeksoitiin, mikä nopeuttaa tietokantahakuja. Tietokantaa määriteltiin, että tyhjiä kenttiä voi olla user_id-, commentator_nickaname- ja parent_comment-kentät.

4.3 Kooditoteutus

Työn kooditoteutus aloitettiin luomalla Builderin koodista oma git branch ja laittamalla se henkilökohtaiseen Builderin kehitysympäristöön (b_dev_8). Ohjelmointi aloitettiin backendista, koska se toimii tiedonlähteenä frontendille. Tietokanta oli yksi oleellisimmista asioista backendissa. Builderin testitietokanta oli jo olemassa, joten sinne piti vain luoda tarvittava taulu. Ensin tehtiin lokaali tietokanta Builderin testitietokannasta. Tähän käytettiin XAMPP:a.

XAMPP on verkkopalvelin, joka mahdollistaa ohjelmiston kehittämisen paikallisella verkkopalvelimella ennen julkaisua. Se koostuu verkkopalvelimesta (Apache), tietokannan hallintajärjestelmästä MariaDB ja tulkista eri ohjelmointikieliä varten kuten PHP ja Perl. Se sisältää myös erilaisia työkaluja kuten PhpMyAdmin, jolla voi käyttää MariaDB:tä. [25.] PhpMyAdmin on ilmainen PHP:llä tehty ohjelmistotyökalu, joka tarkoitettu MySQL:n hallintaan verkossa. Se mahdollistaa monia toimintoja MySQL- ja MariaDB-tietokannoissa. Yleisimpiä toimintoja voi tehdä suoraan käyttöliittymästä, mutta käyttäjä voi myös suorittaa SQL-lauseita suoraan. [26.]

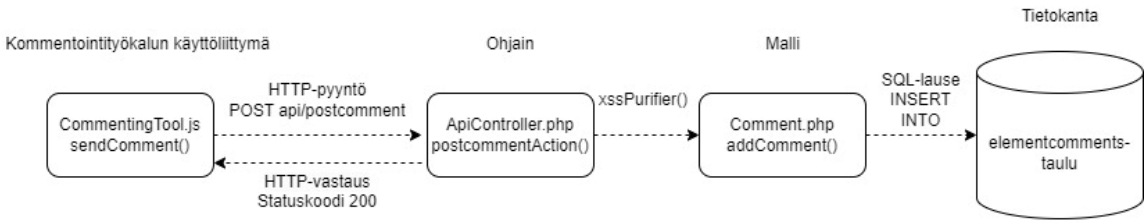
XAMPP:n avulla pystyttiin luomaan helpommin taulu tietokantaan keksimättä itse jokaista tietokantalausetta. Samalla tietokantarakennetta pystyttiin testaamaan vaikuttamatta palvelimella olevaan tietokantaan. Kun oli saatu aikaan halutunlainen tietokantataulun, pystyttiin se viemään ulos SQL-muodossa. Työkalu antaa siis valmiit SQL-lauseet, joilla rakentaa kyseinen taulu. SQL-lauseet kopioitiin Builderin b_test-tietokantaan, jolloin saatiin identtinen taulu testipalvelimelle.

Kommentointityökalua varten tehtiin muutamia uusia tiedostoja Builderiin, koska toteutus haluttiin pitää irrallisena. Työkalulle tehtiin oma tyylitiedosto commentingTool.css, oma JavaScript-tiedosto commentingTool.js ja oma PHP-malli Comment.php. Lisäksi tehtiin muutoksia olemassa oleviin tiedostoihin. Kommentointityökalu luodaan ja alustetaan Navigation.js-tiedostossa omassa funktiossaan. Funktio luo kommentointityökalun, jos kyseinen feature on päällä. Se lisää commentinTool.css HTML:n. Lisäksi se hakee moduulin, elementin ja

fasilitaattorin tiedot AJAX-kutsuilla ja kutsuu commentingTool.js-tiedoston createCommentingTool-funktiota haetuilla tiedoilla. CreateCommentingTool-funktio luo koko kommentointityökalun käyttöliittymän. Kommentointi, poistaminen ja kuittaaminen tapahtuu AJAX-kutsuilla, jotka backend käsittelee.

Builderin käyttöliittymämuutoksia varten tehtiin muutoksia olemassa oleviin AJAX-kutsuihin. HTTP-vastauksiin lisättiin kaikkien kommenttien lukumäärä ja kuittaamattomat kommentit. Builderin frontend-koodissa builder_orig.js- ja main.css-tiedostoissa pystyttiin lisäämään Builderin käyttöliittymään puhekuplaikonit moduuleille ja elementeille.

Kaikki Builderin AJAX-kutsut etenevät samalla tavalla. Esimerkkinä kommentointityökalun kommentin lähettäminen. Kuvassa 11 havainnollistetaan kommentointityökalun kommentin lisäämisen AJAX-kutsun toimintaketju.



Kuva 11. Kommentointityökalun AJAX-kutsun toimintaketju, kun lisätään kommentti.

Kommentin lähettäminen alkaa commentingTool.js-tiedostosta. Tiedostossa on käytetty jQuerya. Käyttäjän lähettäessä kommentin tarkastetaan, onko kommentin sisältö tyhjä, onko lähettäjällä nimimerkki ja onko se vastaus toiseen kommenttiin. Tämän jälkeen kutsutaan sendComment-funktiota (esimerkkikoodi 1).

```
async function sendComment(
    form,
    builderData,
    userData,
    elementData,
    commentData,
    moduleData
  ) {
    $('.comment-list-container .comments', parentDocument).hide();
    $('.comment-list-container .commenting-tool-loader', parentDocu-
ment).show();
    try {
      await $.ajax{
        url: builderData.builderUrl + 'api/postcomment',
        type: 'post',
        data: {
          elementid: elementData.id,
          userid: userData.user_id ? userData.user_id : '',
          comment: commentData.comment,
          commentatornickname: commentData.commentator_nickname,
          parentcomment: commentData.parent_comment,
        },
        success: function (content) {
          console.log('Posted comment succesfully');
          if ($(form).find('.commentator-nickname').length > 0) {
            $(form).find('.commentator-nickname').val('');
          }
          $(form).find('.comment').val('');
          updateCommentList(builderData, moduleData, elementData);
        },
      });
    } catch (error) {
      console.log('Cannot post comment');
      $('.error-text', parentDocument).text('Error: Cannot post com-
ment');
      $('.error-overlay', parentDocument).show();
      updateCommentList(builderData, moduleData, elementData);
      $(
        '.comment-list-container .commenting-tool-loader',
        parentDocument
      ).hide();
      $('.comment-list-container .comments', parentDocument).show();
    }
  }
```

Esimerkkikoodi 1. commentingTool.js-tiedoston sendComment-funktio.

Esimerkkikoodissa 1 sendComment-funktio saa parametreinä esim. lähetyslomakkeen, elementin, johon kommentti kohdistuu, ja kommentin sisällön. Funktio on asynkroninen, jotta AJAX-kutsun valmistumista odotellaan. AJAX-kutsu on käärittу try catch -koodiblokkiin. Jos AJAX-kutsussa tapahtuisi virhe, koodi jatkaisi suoritusta catch-koodiblokkiin, jossa virhetilanne voidaan käsitellä. AJAX-kutsu tekee POST pyynnön Builderin ohjelmointirajapinnan URL-osoitteeseen, jonne se antaa parametreinä elementin id:n, käyttäjän id:n (jos fasilitaattori),

kommentin tekstisisällön, kommentoijan nimimerkin (jos anonyymi käyttäjä) ja kommentin id:n (jos vastaus toiseen kommenttiin). AJAX-kutsu etenee Builderin ohjelmointirajapinnalta ohjaimelle ApiController.php-tiedostoon. Kommentin lähettämistä varten sinne tehtiin postcommentAction-metodi (esimerkkikoodi 2).

```
public function postcommentAction() {
    $comment = new Comment();
    $elementId = $this->xssPurifier($this->getRequest()->getPost('ele-
mentid'));
    $userId = $this->xssPurifier($this->getRequest()->get-
Post('userid'));
    $commentatorNickname = $this->xssPurifier($this->getRequest()-
>getPost('commentatornickname'));
    $commentContent = $this->xssPurifier($this->getRequest()->get-
Post('comment'));
    $parentComment = $this->xssPurifier($this->getRequest()->get-
Post('parentcomment'));

    $comment->addComment($elementId, $userId, $commentatorNickname,
$commentContent, $parentComment);

    $this->_helper->layout->disableLayout();
    $this->_helper->viewRenderer->setNoRender(true);
    $this->render('index');
}
```

Esimerkkikoodi 2. ApiController-ohjaimen postcommentAction-funktio.

Metodin alussa luodaan uusi ilmentymä kommenttiluokasta. Tätä varten kommentille luotiin oma malli Comment.php, jossa määritellään Comment-luokan ominaisuudet. Malli on yhteydessä tietokantaan, ja sinne määritellään lopulliset tietonkantalauseet. PostcommentAction-metodi tarkastaa parametreinä saadut arvot xssPurifier-funktiolla. XssPurifier estää haitallisen koodin pääsemistä järjestelmään. Tämän jälkeen parametreinä saaduilla arvoilla kutsutaan kommentin metodia addComment (esimerkkikoodi 3).

```
public function addComment($elementId, $userId, $commentatorNickname,
$commentContent, $parentComment)
    {
        $h = new Helper();
        $now = date('Y-m-d H:i:s');

        if(!empty($parentComment)){
            $q1 = $h->dbQuery(
                'SELECT resolved, soft_deleted from elementcomments
WHERE comment_id=?',
                [$parentComment],
                1
            );
            if($q1[0]['resolved'] == 1 || $q1[0]['soft_deleted'] ==
1){
                return false;
            }
        }

        $q = $this->_db->query(
            'INSERT INTO elementcomments (element_id, user_id, commen-
tator_nickname, comment, resolved, soft_deleted, modified, created,
parent_comment) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)',
            [$elementId, (!empty($userId) ? $userId : NULL),
(!empty($commentatorNickname) ? $commentatorNickname : NULL) , $com-
mentContent, 0, 0, $now, $now, (!empty($parentComment) ? $parentCom-
ment : NULL)]
        );
    }
```

Esimerkkikoodi 3. Comment-mallin addComment-metodi.

Esimerkkikoodissa 3 AddComment-metodi saa parametreinä aiemmin esille tuodut arvot. Jos kommentilla vastataan toiseen kommenttiin, tarkastetaan, onko vastattava kommentti kuitattu tai poistettu. Jos näin on, ei kommenttia lähetetä tietokantaan. Muussa tapauksessa suoritetaan SQL-lause INSERT INTO, jolla tehdään uusi rivi elementcomments-tauluun sille syötetyillä arvoilla.

4.4 Toiminnallinen testaus, käyttöönotto ja jatkokehitys

Työtä ei saatu julkaisuvalmiiksi opinnäytetyön puitteissa. Työ noudattaa Apprixin julkaisuprosessia, joka sisältää toiminnallisen testaamisen ja koodin tarkastamisen. Ennen toiminnallista testaamista työn ulkoasuun tehdään vielä lopulliset muutokset. Näiden muutosten jälkeen työlle tehdään testaussuunnitelma. Testaussuunnitelmassa pilkotaan kaikki kommentointityökalun toiminnot pieniksi tehtäviksi. Testaaja suorittaa tehtävän ohjeiden mukaan ja kommentoi, toimiiko kommentointityökalu kuten sen on määritelty toimivan. Testausta tulee

tekemään useampi työntekijä ja testaus suoritetaan kaikilla Builderin tukemilla selaimilla.

Testauksen jälkeen käydään läpi siinä tulleet havainnot toimeksiantajan kanssa. Tarkoituksena on päättää mitä korjauksia tehdään ensimmäiseen julkaistavaan versioon ja mitä korjataan tuleviin versioihin. Lopuksi tehdään tarvittavat korjaukset. Lopullisten korjausten jälkeen työ etenee koodin arviointiin, jota tekevät siihen erikseen nimetyt henkilöt. Koodin arvioinnissa otetaan huomioon, että koodi on yhdenmukaista muun Builderin koodin kanssa. Kun koodi on hyväksytty, laitetaan se yleiseen Builder-testiympäristöön (b_test), jossa voidaan tehdä vielä viime hetken testausta ennen julkaisua. Tämän jälkeen fasilitaattoreille viestitään tulevasta uudistuksesta esim. uutiskirjeellä ja työ julkaistaan tuotantoversioon.

Työn aikana on tullut useita jatkokehitysideoita. Julkaisun jälkeen saadaan kommentteja muilta fasilitaattoreilta siitä, mitä he toivoisivat kommentointityökalulta. Jatkokehitysideoita on tarkoitus toteuttaa kommentointityökalun tuleviin versioihin. Seuraavan version aikataulua ei ole vielä sovittu.

5 Tulokset ja löydökset

Kommentointityökalu tulee olemaan osa sisällöntuottajien käyttämiä työkaluja koulutusten suunnittelussa. Toimeksiantajan sisällöntuottajilta haluttiin mielipiteitä kommentointityökalun nykyisestä toteutuksesta kirjalliseen työhön sekä kommentointityökalun kehitystä varten. Tätä varten sisällöntuottajien kanssa pidettiin palaveri, jossa esiteltiin kommentointityökalu. Lisäksi tehtiin kirjalliset ohjeet kommentointityökalun toiminnasta. Kommentointityökalun testausta varten tehtiin moduuli henkilökohtaiseen Builder-testiympäristöön (b_dev_8-ympäristö). Palautetta varten tehtiin tekstitiedosto, johon sai antaa palautetta vapaamuotoisesti ja anonyymisti. Palautelomakkeella oli muutamia apukysymyksiä, jotka liittyivät työkalun hyödyllisyyteen, sen hyödyntämiseen ja kehitysideoihin.

Kaikki testaajat kokivat työkalun erittäin hyödyllisenä, koska koulutuksen suunnittelussa asiakkaan kanssa käydään aina kommentointivaihe. Tällä hetkellä

asiakkaan kanssa kommunikointiin ei ole Builderissa mitään mahdollisuutta. Kaikki olivat halukkaita käyttämään työkalua työssään. Se ei välttämättä korvaa kaikkien muiden viestintä kanavien käyttöä, mutta voi vähentää sitä. Työkalun avulla kommentteja pystyy kohdentamaan tarkasti, ja ne löytyvät samasta paikasta kuin sisältökin. Turhien kuvankaappauksien lähettely tulee todennäköisesti vähenemään. Epäselvien kommenttien määrä tulee luultavasti vähenemään, kun ne on sidottu tiettyyn elementtiin. Asiakkaan kanssa kommentit on helppo käydä läpi. Yhteenvetona työkalu helpottaa, nopeuttaa ja suuntaa kommunikointia. Lähes kaikki testaajat jäivät kaipaamaan sähköposti-ilmoituksia esim. uusista kommenteista tai vastauksista. Lisäksi tuli muutamia käyttökokemusta parantavia kehitysideoita. Tällaisia oli esim. tekstieditorin lisääminen sekä mahdollisuus lisätä kuvia ja linkkejä.

Toimeksiantaja esitti kritiikkiä Builderin käyttöliittymästä, joka ei päivity automaattisesti. Jos annetaan palautetta esikatselunäkymässä, Builder ei hae kommenttien muutoksia automaattisesti, vaan sivun uudelleen latauksen yhteydessä. Jatkossa Builderin käyttöliittymässä halutaan näyttää kaikkien kommenttien lukumäärä. Muutos tulee olemaan helppo, sillä määrä haetaan jo nyt tietokannasta, mutta niitä ei näytetä käyttöliittymässä. Toimeksiantaja oli kokonaisuudessaan tyytyväinen lopputulokseen ja toivoi työkalua tuotantoon mahdollisimman nopeasti.

Työn ohjelmointi eteni ilman suuria ongelmia. Työssä opittiin backend-ohjelmointia. Työn aikana kasvoi syvä ymmärrys Builderin arkkitehtuurista ja toiminnasta. Lisäksi saatiin arvokasta kokemusta projektin läpiviemisessä alusta loppuun sekä projektin organisoimisessa. Työn aikana huomattiin, että mitä huolellisemmin ja tarkemmin alussa määrittelee projektin sisällön, sitä helpompaa itse toteuttaminen on.

6 Yhteenveto

Työn tavoitteena oli kehittää toimeksiantajan verkkokoulutusalustaan vuorovaikutukseen liittyviä ominaisuuksia tukemaan yhteistyötä verkkokoulutusten suunnittelussa. Toimeksiantajan verkkokoulutusalustassa ei ollut aikaisemmin

mahdollista kommunikoida muiden käyttäjien kanssa. Sisällöntuottajat kommunikoivat sisällöntuotantoprojekteissa useissa eri kanavissa ja joutuivat turvautumaan esim. kuvankaappauksiin. Työssä verkkokoulutusalustalle kehitettiin kommentointityökalu, jossa verkkokoulutusten sisältöä voidaan kommentoida.

Työ sisälsi frontend- ja backend-ohjelmointia. Kommentointityökalu toteutettiin ohjelmointirajapinnalla. Tätä varten tehtiin esimerkiksi tietokantamuutoksia. Työ oli erittäin opettavainen, koska Builderin backend-ohjelmointi ei ollut aiemmin tuttua. Työstä saatiin lisää itsevarmuutta tehdä backend-ohjelmointia ja samalla mielenkiinto siihen heräsi.

Kaikki työn määrittelyvaiheessa asetetut ominaisuudet saatiin toteutettua kommentointityökaluun. Työ saatiin opinnäytetyön puitteissa testausvaiheeseen. Ennen testausta kommentointityökalusta kerättiin palautetta sisällöntuottajilta. Kaikki sisällöntuottajat kokivat kommentointityökalun hyödylliseksi ja helpottavan kommunikointia asiakkaiden kanssa. Kaikki sisällöntuottajat kertoivat käyttävänsä kommentointityökalua, kun se julkaistaan. Palautteen pohjalta koottiin jatkokehitysideoita, joita toteutetaan julkaisun jälkeisiin versioihin. Asiakkaiden mielipiteitä kommentointityökalusta kuullaan vasta julkaisun jälkeen.

Työn alussa haasteita tuotti määrittely ja koordinointi. Määrittelyvaiheessa ideoita oli paljon, ja toiminnallista määrittelyä oli tekemässä useita henkilöitä. Haasteena oli määritellä työlle realistiset mittasuhteet ja tavoitteet. Vaikeaa työn etenemisen kannalta oli tehdä tarvittavat päätökset, ja työn etenemistä varten päätöksiä tuli välillä vaatia. Palaverien valmistelu ja palaverimuistiinpanojen tekeminen ei ollut aiemmin tuttua, mikä oli haastavaa.

Lähteet

1 Jokinen, Jesse. 2023. Managing Partner, Apprix Oy, Helsinki. Keskustelu 25.5.2023.

2 What is SaaS? Verkkoaineisto. Microsoft. <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas>. Luettu 12.5.2023.

3 Boucher, Claire. 2022. What Is An Online Training Platform? Verkkoaineisto. HowToo. <https://www.howtoo.co/posts/what-is-an-online-training-platform>. 26.5.2022. Luettu 13.10.2023.

4 Ninoriya, Suman; Chawan, P.M & Meshram B.B. 2011. CMS, LMS and LCMS For eLearning. International Journal of Computer Science Issues, 8(2), s. 644–647.

5 Ninoriya, Suman; Chawan, P.M & Meshram B.B. 2011. Development of LCMS for Collaborative E-learning. International Journal of Computer Science Issues, 2(2), s. 459–462.

6 Siljander, Pauli. 2014. Systemaattinen johdatus kasvatustieteeseen: peruskäsitteet ja pääsuuntaukset. Tampere: Vastapaino.

7 Hurrell, Susan. What is online training? Verkkoaineisto. Neovation. <https://www.neovation.com/learn/47-what-is-online-training>. Luettu 13.10.2023.

8 Korhonen, Konsta. 2023. Erikoisasiantuntija, Apprix Oy, Helsinki. Haastattelu 7.7.2023.

9 Kock, Ned; Davison, Robert; Wazlawick, Raul & Ocker, Rosalie. 2001. E-collaboration: A look at past research and future challenges. Journal of systems and information technology, 5(1), s. 1–8.

10 Jackson, Victoria; van der Hoek, Andre; Prikladnicki, Rafael & Ebert, Christof. 2022. Collaboration Tools for Developers. IEEE software, 39(2), s. 7–15.

11 Ebert, Christof & Tavernier, Bertrand. 2021. Technology Trends: Strategies for the New Normal. IEEE software, 38(2), s. 7–14.

12 Gillis, Alexander. 2022. artifact (software development). Verkko-aineisto. TechTarget. <https://www.techtarget.com/searchsoftwarequality/definition/artifact-software-development/>. 1.2.2022. Luettu 17.11.2023.

13 Petrovica, Sintija; Anohina-Naumeca, Alla & Kikans Andris. 2020. Definition and Validation of the Subset of SCORM Requirements for the Enhanced Reusability of Learning Content in Learning Management Systems. Applied Computer Systems (Online), 25(2), s. 134–144.

14 API. 2023. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Glossary/API>. Päivitetty 8.6.2023. Luettu 24.11.2023.

15 What is an API? Verkkoaineisto. Postman. <https://www.postman.com/what-is-an-api/>. Luettu 27.10.2023.

16 Ajax. 2023. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Glossary/AJAX>. Päivitetty 22.11.2023. Luettu 24.11.2023.

17 Using XMLHttpRequest. 2023. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest_API/Using_XMLHttpRequest>. Päivitetty 9.11.2023. Luettu 24.11.2023.

18 XMLHttpRequest: open() method. 2023. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/open>. Päivitetty 8.11.2023. Luettu 24.11.2023.

19 XMLHttpRequest: send() method. 2023. Verkkoaineisto. MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/send> Päivitetty 8.11.2023. Luettu 24.11.2023.

20 Fielding, Roy; Nottingham, Mark & Reschke, Julian. RFC 9110 HTTP Semantics. 2022. Verkkoaineisto. Internet Engineering Task Force. <https://www.rfc-editor.org/rfc/rfc9110.html>. 1.6.2022. Luettu 8.12.2023.

21 Thibault. 2016. What are HTTP Resources? Verkkoaineisto. Devblast. <https://devblast.com/b/what-are-http-resources>. 23.10.2016. Luettu 22.12.2023.

22 Pitt, Chris. 2012. Pro PHP MVC. E-kirja. Apress.

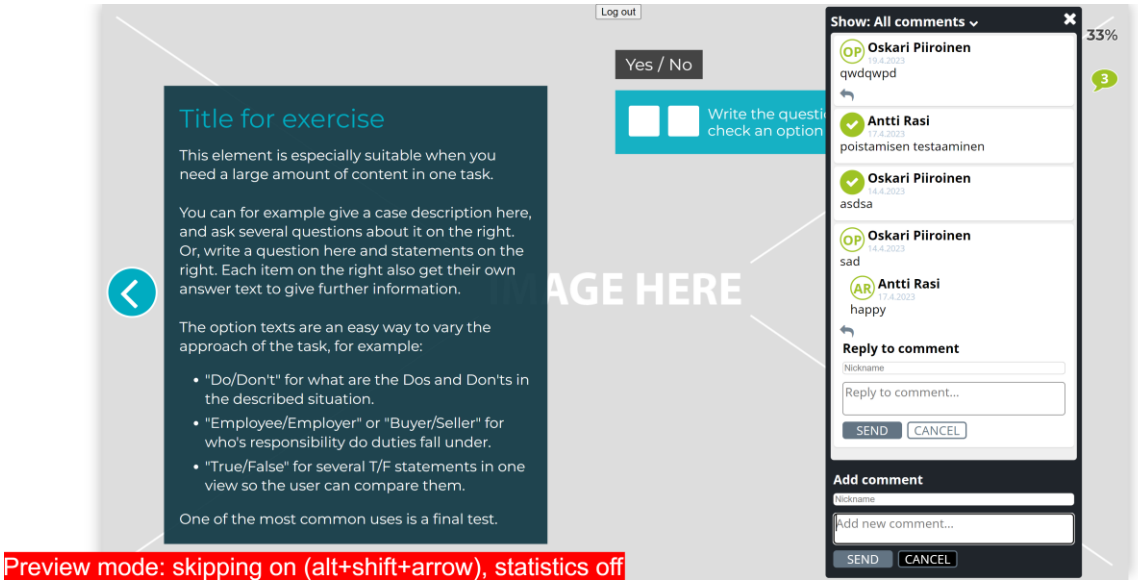
23 MVC for dummies: malli, näkymä ja ohjain -arkkitehtuuri web-sovelluksissa. 2020. Verkkoaineisto. Hurja. <https://www.hurja.fi/blogi/mvc-for-dummies-malli-nakyma-ja-ohjain-arkkitehtuuri-web-sovelluksissa/>. Päivitetty 2.2.2023. Luettu 3.11.2023.

24 Allen Rob; Brown, Steven & Lo Nick. 2009. Zend Framework in Action. E-kirja. Manning Publications.

25 XAMPP TUTORIAL. Verkkoaineisto. JavaTpoint. <https://www.javat-point.com/xampp>. Luettu 20.10.2023.

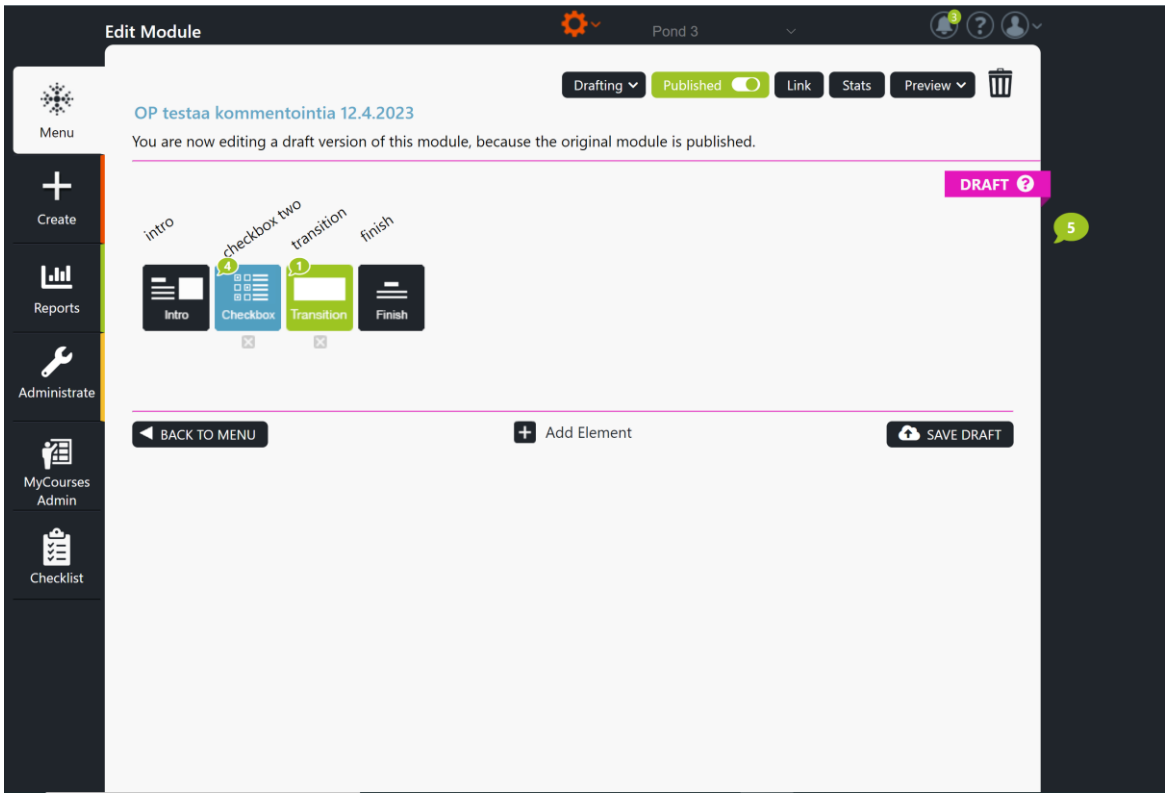
26 Bringing MySQL to the web. Verkkoaineisto. phpMyAdmin. <https://www.phpmyadmin.net/>. Luettu 20.10.2023.

Kommentointityökalu moduulin esikatselunäkymässä



Builderin moduulilistausnäkymän käyttöliittymä

Builderin moduulin muokkausnäkymän käyttöliittymä



Kommentointityökalu Builderin elementin muokkausikkunassa