Warhammer 40 000 3. Edition Army Builder

LAB-ammattikorkeakoulu
Tradenomi (AMK), Digitradenomi
2021
Iiro Kutvonen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Kutvonen, Iiro	Opinnäytetyö, AMK	2021
	Sivumäärä	
	18	

Työn nimi

Warhammer 40 000 3. Edition Army Builder

Tutkinto

Tradenomi (AMK)

Toimeksiantajan nimi, titteli ja organisaatio

-

Tiivistelmä

Projektin tavoite oli tuottaa demoversio ohjelmasta, jota voidaan hyödyntää Warhammer 40 000 pelin harrastuksessa. Ohjelman on tarkoitus antaa käyttäjälleen helposti lähestyttävä käyttöliittymä, jolla tuodaan harrastajille arvoa sekä käyttökokemuksen että ajansäästön kautta pelin pakollisen kirjanpidon kanssa.

Ohjelma kehitettiin Angular ohjelmistokehykseen yksisivuisena verkkoapplikaationa, joka kykenee kasaamaan armeijalistan ja laskemaan sen pisteet pelikäyttöä varten. Ohjelma sisältää yhden armeijalistan, joka tulee toimimaan sapluunana muiden armeijalistojen kehitystä varten jatkokehityksen aikana.

Opinnäytetyön lopputuloksena oli toimiva ohjelma ja ehdotuksia jatkokehitystä varten.

Asiasanat

harrastus, käyttöliittymä, käyttökokemus

Abstract

Author(s)	Type of Publication	Published
Kutvonen, Iiro	Thesis, UAS	2021
	Number of Pages	
	18	

Title of Publication

Warhammer 40 000 3. Edition Army Builder

Name of Degree

Bachelor of Business Administration (UAS)

Name, title and organization of the client

-

Abstract

The aim of the project is to produce a program demo to use in the Warhammer 40 000 gaming hobby. The program is meant to have an easily approachable and offer value to hobbyists through easy-to-use user experience and time savings when it comes to mandatory bookkeeping the game requires.

The program was developed with Angular framework as a single-page web application that can build an army list and calculate its point cost. The program will contain a single army list that will be used as a template for during later development.

The result of the thesis is a functional program and several suggestions for further development.

Keywords

hobby, user interface, user experience

Sisällys

1 Johdanto

Tässä opinnäytetyössä toteutetaan armeijalistan rakennusohjelma Games Workshop Groupin tuottamaan Warhammer 40 000 pelin 3. laitokseen. Warhammer 40 000 on sotapeli, jossa pelaajat simuloivat fiktiivistä taistelua miniatyyrien avulla. Aiheen valinta perustuu muutokseen miniatyyripelaamisen aputyökaluissa. Yhä useampi pelivalmistaja siirtää tuottamaansa sisältöä sähköiseen muotoon sekä tarjoaa asiakkailleen sähköisiä työkaluja. Näiden avulla pelaajien käyttökokemusta pyritään parantamaan sekä tukemaan harrastajia myös varsinaisen pelin ulkopuolella esimerkiksi peleihin valmistautumisessa.

Pelivalmistajien tarjoamat sovellukset keskittyvät pääsääntöisesti armeijalistan rakennusohjelmiin. Ne tuottavat myös esimerkiksi noppa-applikaatioita, ja sovelluksia, jotka tarjoavat ohjeistusta miniatyyrien maalaamiseen. Näin ollen sähköinen sisältö on tullut osaksi miniatyyripelaamista sekä peleissä itsessään että niihin liittyvissä tukitoiminnoissa. Tämä digitalisoituminen koskee pääsääntöisesti vain pelejä, jotka on julkaistu viimeisen viiden vuoden aikana. Tätä ennen miniatyyripeliyritykset eivät panostaneet samalla tavalla sähköiseen materiaaliin, vaan useimmiten ohjelmat ja applikaatiot olivat fanituotoksia.

Games Workshop Group on aiemmin tuottanut neljä armeijanlistan rakennusohjelmaa. Ensimmäinen ilmestyi 3. laitoksen aikana Cd-levyillä. Kyseinen ohjelma on kuitenkin poistunut myynnistä eikä se ole saatavilla muissa muodoissa. Seuraavat ohjelmat tuotettiin 8. ja 9. laitosten aikana. 8. laitoksen alku sijoittuu aikaan, jolloin sovellusten kehittäminen pelivalmistajien toimesta alkoi yleistyä. 8. ja 9. laitoksen aikana tuotetut ilmaiset sovellukset tarjoavat vain yksinkertaistetun version armeijalistan luonnista. 9. laitoksen aikana julkaistiin myös Warhammer 40 000 App, johon sisältyy armeijanlistan rakennusohjelma. Applikaatio kuitenkin vaatii käyttäjää ostamaan fyysisen kirjan armeijalistan avaamiseksi sekä kuukausimaksun käyttöä varten.

Idea opinnäytetyöhön löytyi, kun pelin harrastajien kesken huomattiin armeijalistan rakennusohjelman puuttuminen vanhemmista pelin laitoksista. Myös retropelaamisen yleistyminen viime vuosikymmenen aikana kannusti opinnäytetyön aiheen valintaan. Retropelaaminen liitetään pääsääntöisesti videopeleihin, mutta samaa ilmiötä on tavattavissa myös muiden pelityyppien kohdalta. Miniatyyripelaamisessa retropelaaminen on yleistä etenkin vuosia tai jopa vuosikymmeniä harrastaneiden joukossa. Mutta koska pelien vanhoja laitoksia ei ole yritysten toiminnan kannalta järkevää tukea, eivät ne nauti samanlaisesta tuesta kuin uudemmat laitokset.

Tässä nähtiin mahdollisuus tuottaa työkalu, jossa voidaan yhdistää vanhat pelin laitokset moderniin teknologiaan armeijalistan rakennusohjelman kautta. Tämä toteutetaan

käyttämällä Angularia, joka on suosittu yksisivuisten verkkosovellusten kehittämiseen tarkoitettu ohjelmistokehys. Angularin monipuolisuus mahdollistaa sekä sujuvan käyttöliittymän, että ohjelman toiminnallisuuden toteuttamisen.

1.1 Opinnäytetyön tavoitteet

Opinnäytetyön tavoite on tuottaa toimiva demoversio ohjelmasta, jolla voidaan luoda armeijalista sekä laskea listan pistearvo Warhammer 40 000 3. laitoksessa. Demoversiossa kehitetään ohjelman perusrakenne, toimintaperiaatteet sekä tarvittavat laskukaavat, joita käytetään pistearvojen laskemiseen. Lisäksi opinnäytetyössä kehitetään ohjelmalle helposti omaksuttava ja selkeä käyttöliittymä. Käyttöliittymän suunnittelussa otetaan huomioon käyttökokemuksen tärkeys, sillä ohjelman tavoite ei toteudu, jos sen käyttäminen ei nopeuta tai paranna tavallista käsinkirjoitetun armeijalistan kasausta.

Käytettävien teknologioiden puolesta tavoitteena on syventyä Angulariin. Angularin käytössä halutaan erityisesti katsoa, kuinka pitkälle ohjelma voidaan kehittää pääsääntöisesti sen voimin. Normaalisti vastaavanlaiset ohjelmat käyttävät tietokantoja tukena, mutta ohjelman toteutuksen yhteydessä pyritään luomaan koko ohjelman toiminnallisuus Angularin komponenttien kautta.

1.2 Opinnäytetyön rajaukset

Ohjelma rajataan sisältämään yhden pelikirjan armeijalista. Näin on mahdollista keskittyä demoversion kannalta tarpeelliseen kehitykseen. Yhden pelikirjan pohjalta voidaan myös toteuttaa ohjelman perusrakenne, jota voidaan soveltaa muissa pelikirjoissa esiintyvien armeijalistojen toteutukseen.

Ohjelman lähdemateriaali sisältää sekä fiktiivisiä tarinoita, sääntöjä sekä kuvaukset yksiköistä ja niiden ominaisuuksista. Ohjelmaa varten kerättävä materiaali rajataan vain välttämättömiin tietoihin, joita ohjelma tarvitsee toimiakseen:

- yksiköiden nimet
- yksiköiden pistearvot
- yksiköiden minimi ja maksimi koot
- yksiköiden roolit
- yksiköiden lisäoptioiden nimet
- yksiköiden lisäoptioiden pistearvot

- varusteiden nimet
- varusteiden pistearvot

Ohjelman toteutus rajataan tietokonepohjaiseksi verkkosovelluksesi, joka toteutetaan Angularilla.

2 Teknologiat

2.1 Angular

Angular on avoimen lähdekoodin TypeScript-pohjainen front end alusta ja ohjelmistokehys, jonka avulla luodaan yksisivuisia applikaatioita verkko selaimille, mobiililaitteille ja tietokoneille. Angularin perustana ovat ngmoduulit, komponentit sekä erilaiset kirjastot, jotka sisältävät valmiita ohjelmia, luokkia ja aliohjelmia. Angular sisältää myös kehitystyökaluja, joiden avulla kehitettävää ohjelmaa voidaan testata ja päivittää sekä luoda siitä toimivan koontiversion.

Ngmoduulit luovat Angularin perusrakennuspalikat ohjelmistokehykselle. Angular applikaatio sisältää aina vähintään yhden ngmoduulin, joka on applikaation juurimoduuli. Juurimoduuli sisältää applikaation esilatausohjelman, joka mahdollistaa ohjelman käynnistyksen. Ngmoduuleja, joista yleisimmät ovat toimintomoduulit ja reititysmoduulit, voidaan lisätä tarpeen vaatiessa. Ngmoduulit sisältävät komponentteja, luokkia, palveluita, reittejä, piippuja tai muita tiedostoja, riippuen kyseisen moduulin tarkoituksesta. Esimerkiksi juurimoduuli sisältää aina vähintään yhden komponentin, koska moduuli toimii sekä applikaation rakenteen että ohjelman tuottaman näkymän perustana.

Angularin komponentit koostuvat TypeScript, Cascading Style Sheet ja Hypertext Markup Language tiedostoista. Komponentit toimivat käyttöliittymän palasina, joista muodostuu Angular applikaation visuaalinen ulkoasu. Angularin käytössä voidaan myös hyödyntää Angularin skematiikkoja, jotka luovat valmiita pohjia esimerkiksi komponentteja, luokkia tai palveluita varten. Skematiikat ovat nopea tapa aloittaa esimerkiksi uuden komponentin kehittäminen, sillä automatisoitu pohjan luominen hoitaa kaikki tarpeelliset riippuvuudet ja deklaraatiot, joita uuden komponentin toiminta vaatii.

Angularin päätarkoitus on yksisivuisten verkkosovellusten kehittäminen, jossa samalle sivulle ladataan näytettävät komponentit ilman varsinaista sivun uudelleen lataamista. Rakenteensa ansiosta verkkosovellukseen ladataan esiin vain tarvittavat komponentit reititystoimintonsa ansiosta. Ne toimivat omassa moduulissaan, jossa reitit sekä mahdolliset käyttäjäryhmien oikeudet määritellään. Reititys toimii osoiterivimuutoksilla, jotka toimivat kuten normaalissa verkkoselaimessa ilman tarvetta sivun uudelleen lataamiseen. Reititystä käytettäessä reittiin määritetty komponentti kutsutaan ja se tuodaan näytölle samalla kun edellisen näkymän komponentit siirretään pois näkymästä. Lisäksi komponentteja voidaan myös liittää toisiin komponentteihin, jolloin kun tietty komponentti kutsuttaan, tuo se mukanaan siihen liitetyt komponentit. (Angular 2021.)

2.1.1 Angular Material

Angular Material on Angularia varten kehitetty kirjasto, jonka avulla ohjelmiston graafiset elementit voidaan muotoilla helposti ja nopeasti käyttämällä valmiiksi määritettyjä elementtejä. Angular Material tarjoaa runsaan valikoiman erilaisia elementtejä, jotka vaihtelevat yksinkertaisemmista napeista ja valikoista työkaluriveihin ja kalenterimallisiin päivämäärän valinta työkaluihin. Angular Materialin elementit sisältävät usein animaatioita, jotka voidaan tarvittaessa myös ottaa pois käytöstä tarpeen vaatiessa. (Angular Material 2021.)

2.2 Git ja Github

Git on versionhallinta järjestelmä, jonka avulla ohjelmaan tuotettua koodia voidaan hallita versioiden kautta lokaalisti. Git pystyy luomaan tallennuspisteitä, joiden kautta eri versioihin voidaan palata tai luoda niistä haarautuvia versioita. Tämän käytännön ansiosta voidaan kehitystyössä kokeilla erilaisia ratkaisuja haarassa ilman haittoja alkuperäiselle versiolle. Nämä omat haarat voidaan myös myöhemmin liittää takaisin alkuperäiseen, jolloin haarassa tehty kehitystyö saadaan lisättyä takaisin ohjelmaan. (Git 2021.)

Github on pilvipalvelu, joka tuo Gitin verkkoon. Verkkopalveluna Github tarjoaa Gitin käyttöön käyttöliittymän sekä verkkoselaimessa että ladattavana sovelluksena. GitHub tarjoaa verkossa toimivia repositorioita projekteille, joiden kautta useampi henkilö pystyy osallistumaan saman projektin kehitykseen. Lisäksi Github tarjoaa erilaisia työkaluja, jotka helpottavat haarojen seuraamista, tarkastusta sekä liittämistä takaisin alkuperäiseen versioon. Myös muita tiimityöskentelyä edistäviä ominaisuuksia on lisätty Githubiin parantamaan Gitin käyttökokemusta projekteissa. (GitHub Inc. 2021.)

2.3 WebStorm

WebStorm on JetBrainsin kehittämä ohjelmointiympäristö sovellus, joka käyttää JavaScriptiä sekä siitä johdettuja ohjelmointikieliä. WebStorm tukee suoraan useita erilaisia ohjelmistokehyksiä sekä verkko-, mobiili-, palvelin- ja työpöytäsovellusten osalta. Lisäksi WebStorm sisältää useita erilaisia työkaluja kehitystyöhön, kuten esimerkiksi automaattinen koodin tarkistustyökalu. WebStormiin voidaan myös liittää GitHub plugin. (JetBrains 2021a.)

2.4 Node Package Manager

Node Package Manager (npm) on paketinhallintajärjestelmä, joka koostuu komentorivistä, pakettirekisteristä ja verkkosivusta. Komentorivillä suoritetaan muun muassa pakettien lataukset ja se on pääsääntöinen kehittäjien käyttämä käyttöliittymä. Pakettirekisteri on tietokanta, josta löytyy sekä ilmaisia että maksullisia JavaScript ohjelmistoja. Verkkosivun avulla

voidaan selata rekisteristä löytyviä ohjelmistoja, mutta se toimii myös virallisena ohjeistuksena npm:n käyttöön. (npm, Inc. 2021.)

2.5 Node.js

Node.js on alustariippumaton avoimen lähdekoodin ajoympäristö JavaScript pohjaisen koodin suorittamiseen palvelimella. Node.js mahdollistaa verkkosivun lähettämisen käyttäjälle vasta, kun koodi on suoritettu palvelimella sen sijaan, että JavaScript koodi suoritettaisiin käyttäjän koneella. (OpenJS Foundation 2021.)

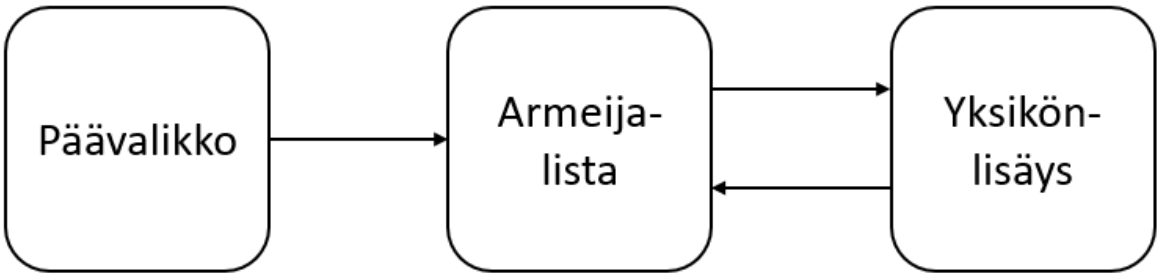
3 Suunnittelu

3.1 Oikeudelliset luvat

Ohjelman sisällön lähdemateriaali on tekijänoikeuslain mukaisesti suojattu, joten sen käyttöön haettiin Games Workshop Groupin lupa (Tekijänoikeuslaki 61/404). Lupa käyttöön hankittiin Games Workshop Groupin oikeudellisen osastolta sähköpostitse. Lupa materiaalin käyttöön myönnettiin seuraavin ehdoin: ohjelma ei saa olla kaupallinen, eikä sitä saa jakaa. Lisäksi Games Workshop Groupin verkkosivulta löytyy yrityksen immateriaalioikeuksiin liittyvät käytännöt. Näitä käytäntöjä kunnioittaen tehtiin päätös olla esittämättä tekijänoikeuslain alaista materiaalia opinnäytetyöraportissa, sillä raportti tullaan julkaisemaan opinnäytetyön ohjeen mukaisesti. (Games Workshop Group 2021.)

3.2 Rakenne

Ohjelman rakenteen suunnittelussa pääpaino oli selkeällä ja helposti toteutettavalla rakenteella. Tätä varten tutustuttiin kahteen armeijalistan rakennusohjelmaan, joista ensimmäinen on useaa peliä tukeva BattleScribe ja toinen Games Workshop Groupin julkaisema Combat Roster (BattleScribe 2021, Warhammer Community 2021). Näiden ohjelmien ulkoasusta ja toiminnasta haettiin tietoa armeijalistan rakennusohjelman rakenteesta. Ohjelmista tehtyjen havaintojen perusteella syntyi ajatus mahdollisimman yksinkertaisesta armeijalistan rakennusprosessista. Ajatus päätettiin toteuttaa kolmen näkymän kautta (Kuva 1), jotka seuraisivat suoraviivaista etenemispolkua armeijalistan rakentamisessa. Prosessin aloitus tapahtuisi päävalikosta, jossa armeijalista valitaan ja siirrytään armeijalistaukseen. Täältä avattaisiin napin painalluksella dialogi-ikkuna yksikön lisäystä varten. Seuraavaksi listasta valitaan haluttu yksikkö sekä päätetään yksikön lisäoptiot ja -varusteet. Lopulta tiedot tallennetaan armeijalistaan ja dialogi suljetaan. Tämä prosessi toistetaan, kunnes armeijalista on valmis.



Kuva 1. Näkymien suunniteltu rakenne

3.3 Käyttöliittymä

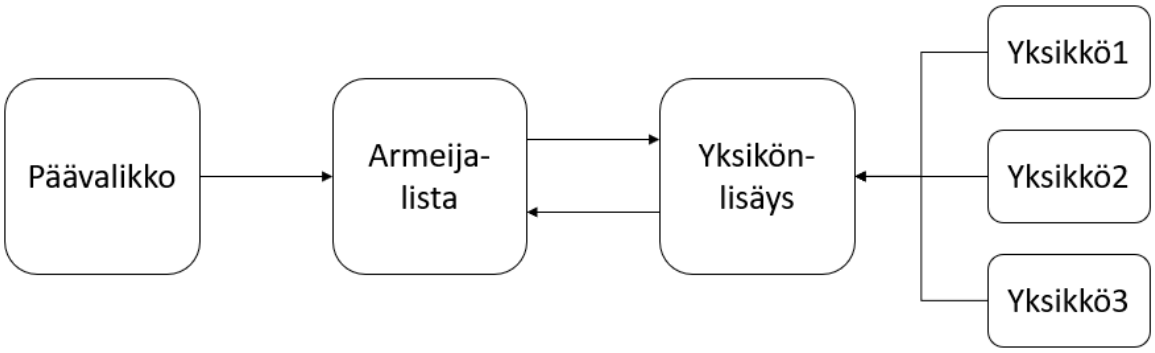
Käyttöliittymän suunnittelussa päätettiin, että ensimmäiseksi siitä tehtäisiin luonnosprototyyppi. Prototyypin tarkoituksena on saada käsitys visuaalisten elementtien sijainneista. Elementtien sijoittelun koetaan auttavan myös hahmottamaan kuinka toiminnallisten osuuksien esittäminen käyttäjälle tulisi tehdä. Käyttöliittymän lopullinen ulkomuoto toteutetaan Angular Materialin avulla mahdollisimman siistiksi ja selkeäksi. Käyttöliittymän suunnittelussa varauduttiin myös siihen, että sitä jouduttaisiin muokkaamaan useita kertoja projektin aikana toiminnallisuutta kehitettäessä.

3.4 Toiminnallisuus

Ohjelman toiminnallisuuden suunnittelu perustui ohjelman tarkoitukseen tuottaa armeijalista ja sen pistearvo. Listausta varten täytyy yksiköiden tiedot kerätä sellaiseen muotoon, jotta tiedot voidaan esittää armeijalistassa. Listausta varten pitää yksikön tiedoista ottaa talteen yksikön nimi, rooli, jäsenten määrä, sekä yksikölle valitut lisäoptiot ja -varusteet. Näiden tietojen pohjalta voidaan suorittaa yksiköiden pistearvojen laskeminen.

Pistearvojen laskeminen on matemaattisesti yksinkertaista. Yksikön jokaisella jäsenellä on peruspistearvo, joka kerrotaan yksikön jäsenten määrällä. Mikäli yksikköön lisätään lisäoptioita tai -varusteita, pitää jokaisen option ja varusteen kohdalla katsoa, nostaako se jokaisen yksikön jäsenen pistearvoa vaiko vain yhden. Jokaisen jäsenen pistearvoon vaikuttavat pistearvon nostot lisätään peruspistearvoon ennen määrällä kertomista, kun taas yhden jäsenen pistearvon nousu voidaan lisätä kertomisen jälkeen kokonaissummaan.

Suunnitteluhaasteen toiminnallisuuteen loi yksiköiden vaihtelevat pistearvot ja yksikön jäsenten määrät, sekä eri lisäoptio ja -varuste vaihtoehdot. Tämä ratkaistiin suunnitteluvaiheessa päättämällä, että jokaiselle yksikölle luodaan oma komponentti, johon tallennetaan kunkin yksikön tiedot (Kuva 2). Yksikkökomponentit antavat myös mahdollisuuden toteuttaa jokaiselle yksikölle räätälöidyn näkymän.



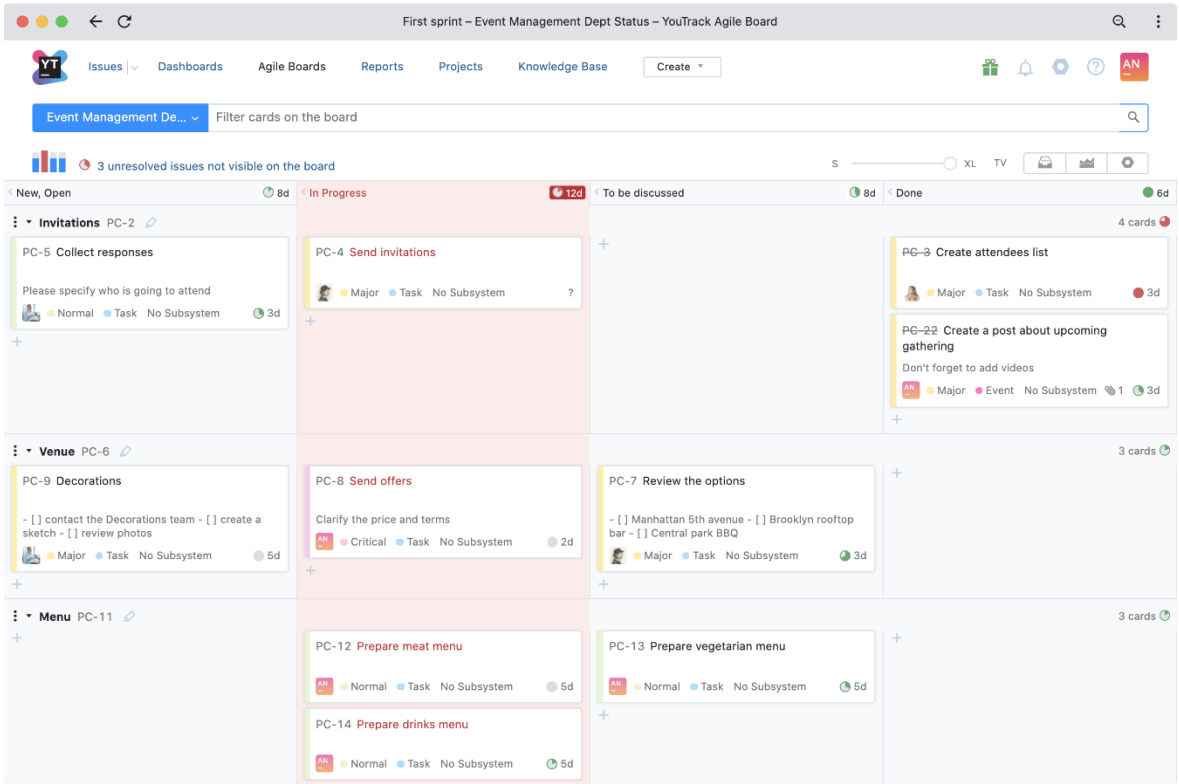
Kuva 2. Toiminnallisuuteen perustuva lisäys suunniteltuun rakenteeseen

Yksiköiden tietojen kerääminen ja pistearvojen laskeminen tapahtuu useiden komponenttien välillä, joten tietojen välitykseen tarvitaan palveluja. Demoversiossa palveluita on kaksi, tietojenkeräyspalvelu ja pisteiden keräyspalvelu. Palvelut ottavat vastaan ja välittävät tiedot muuttujina komponenttien välillä. Komponenteissa tiedot käyvät läpi prosessin, jossa ne muunnetaan sopivaan muotoon armeijalistan kannalta.

3.5 Kanban projektinhallinta menetelmä

Projektinhallinnassa otettiin käyttöön Kanban, joka on ketterä ohjelmistokehityksen projektinhallinnan menetelmä. Työnkulkua Kanbanissa hallinnoidaan Kanban taululla (Kuva 3), jonka voi kustomoida projektin tarpeiden mukaan. Opinnäytetyössä taulun käytön tavoite oli seurata projektin etenemistä sekä hallita projektiin liittyviä tehtäviä.

Opinnäytetyön Kanban taulu luotiin käyttämällä työpisteen viereistä seinää, johon liimattiin tehtävien nimillä varustettuja post-it lappuja. Lappujen sijainti seinällä määritti työn sen hetkisen tilan. Aloittamatta olevat työt pidettiin seinän oikeassa reunassa, kesken olevat työt pidettiin seinän vasemmassa reunassa ja valmiit työt poistettiin seinältä. (Jetbrains 2021b.)



Kuva 3. Esimerkki Kanban taulusta (JetBrains 2021c)

3.6 Tietoturvallisuus

Projektissa toteutettava ohjelma suunniteltiin toimimaan ilman, että se kerää tai tallentaa minkäänlaista tietoa käyttäjästä. Näin taataan käyttäjän yksityisyys. Armeijalistat tallentuvat käyttäjän selain näkymään vain siksi aikaa, kun käyttäjä on armeijalistanäkymässä. Näkymästä poistuttaessa, ohjelma hävittää kasatun armeijalistan pysyvästi. Näin vältetään kaikki mahdolliset tallennettavan tiedon harmaat alueet ja varmistetaan ohjelman toiminnan olevan yleisen tietosuoja-asetuksen mukainen (Tietosuoja-asetus 2016/679).

4 Toteutus

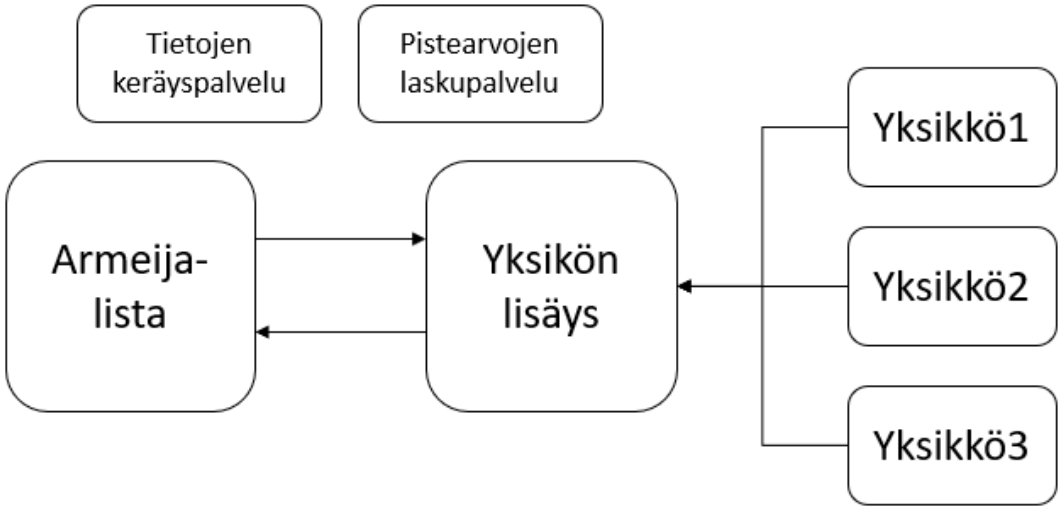
4.1 Rakenne

Rakenteen luomisen yhteydessä arvioitiin myös komponenttien tärkeys ohjelman toiminnan kannalta. Koska ohjelma tuottaa demoversiona vain yhden armeijalistan, ei päävalikkoa armeijalistan valitsemiseen tarvittu. Päävalikkokomponentin kehitys pysäytettiin ja se päätettiin siirtää jatkokehityksen piiriin.

Ohjelman rakenteen toteuttamisen ohessa havaittiin, ettei reititystä demoversioon tarvita. Ohjelman aloitusnäkymän siirryttyä armeijalistakomponenttiin kaikki ohjelman näkymät pystyttiin esittämään sen alaisuudessa seuraamalla suunnitelmassa kuvattua etenemispolkua kaiken muun paitsi päävalikon osalta.

Ohjelman rakenteen valmistuttua siirryttiin määrittelemään, mitä tietoja mikäkin komponentti pitää sisällään. Armeijalistakomponentti itsessään ei sisällä muuta, kuin yksikköroolleihin perustuvan listanäkymän, sekä napin yksikön lisäyskomponentin dialogi-ikkunan avaamiseksi. Yksikön lisäyskomponentissa esitetään listaus kaikista armeijalistan yksiköistä, yksikön tietojen perusrakenne (yksikön nimi, jäsenten määrä ja kokonaispistearvo) ja yksikkölistauksesta valitun yksikön komponentin näkymä. Muita elementtejä komponentissa ovat napit, joiden avulla yksikön kokoa voidaan säätää, sekä nappi yksikön armeijalistaan tallennusta varten. Yksikkökomponenttien sisältöön kuuluvat kaikki komponentin yksikön tiedot (lueteltu aiemmin opinnäytetyön rajauksessa), sekä yksikön lisäyskomponentissa olevasta perusrakenteesta poikkeavat elementit, kuten esimerkiksi lisäoptiot ja -varusteet.

Lopuksi rakenteeseen lisättiin palvelut, joita tullaan tarvitsemaan toiminnallisuutta kehitettäessä. Palveluiden sisältö jätetään oletus muotoonsa odottamaan toiminnallisuuden kehittämistä, sillä vasta silloin tiedetään, mitkä tiedot välitetään komponentilta toiselle. Palveluiden lisäämisen jälkeen ohjelman rakenne oli valmis (Kuva 4).



Kuva 4. Ohjelman rakenne toteutusvaiheessa

4.2 Käyttöliittymä

Ohjelman käyttöliittymän kehittämisessä seurattiin suunnitelmaa aloittamalla käyttöliittymän visuaalisesta prototyypistä. Prototyypin kehitys aloitettiin luomalla toimiva navigaatio ohjelman käyttöä varten, jotta komponentteihin tehdyt muutokset saataisiin helposti todennettua. Navigaation jälkeen tehtiin armeijalista- ja yksikön lisäyskomponentteihin visuaalisten elementtien sijoittelu. Armeijalistaan lisättiin suunniteltu listausnäkymä ja yksikön lisäykseen luotiin paikat yksikön perustiedoille, malli yksikkölistauksista sekä nappi tallennusta varten. Tässä vaiheessa prototyyppi oli valmis ja käyttöliittymän kehitys jätettiin taka-alalle.

Käyttöliittymän päivitys prototyypistä varsinaiseksi käyttöliittymäksi suoritettiin toiminnallisuuden kehityksen yhteydessä. Samalla käyttöliittymään otettiin käyttöön Angular Materialin graafiset elementit, joiden avulla käyttöliittymän ulkoasua kehitettiin suunnitelman mukaisesti selkeämmäksi.

Armeijalistanäkymän päivitys osoittautui hyvin yksinkertaiseksi, sillä ainoa muutos oli näkyvien elementtien päivitys käyttämään Angular Materialia. Yksikön lisäyksen päivitys suoritettiin vasta, kun pistearvojen laskutoimitukset toimivat ja kaavat liitettiin yksikkökomponentteihin. Tämä tarkoitti yksikkölistauksen muuttamista pelkästä mallista toimivaksi ominaisuudeksi. Samalla yksikkölistaus muutettiin prototyypin perinteisen mallisesta pudotusvalikosta visuaalisesti selkeämpään yksikön roolien mukaan jaoteltuun nappipohjaiseen valikkoon. Lisäksi yksikön lisäysnäkymä päivitettiin Angular Materialin elementteihin.

Yksikkönäkymien kohdalla päivittämisen tarve vaihteli suuresti. Yksiköstä riippuen yksikkönäkymään täytyi lisätä lisäoptiot, lisävarusteet, molemmat tai ei mitään. Näiden lisäysten määrä myös vaihteli yksiköiden välillä runsaasta määrästä lisäyksiä vain yhteen tai kahteen.

Prototyypissä käytettiin lisäoptio ja -varusteiden lisäämisen nappia, jonka luomasta pudotusvalikosta varusteet tai optiot voitiin valita. Tämä lähestymistapa hylättiin, koska sen käyttäminen oli kömpelöä. Sen sijaan yksikkönäkymissä kaikki vaihtoehdot tuotiin näkyville kerralla ja niiden valintaan otettiin käyttöön valintaruudut ja -napit. Tämä muutos oli visuaalisesti mielekkäämpi ja näytti paremmin yksikkö kohtaisesti määriteltävät lisäoptio ja -varuste vaihtoehdot. Näiden päivitysten jälkeen käyttöliittymä todettiin valmiiksi.

4.3 Toiminnallisuus

Toiminnallisuuden toteuttaminen päätettiin tehdä paloissa, joissa yksi toiminto tehtiin kerralla valmiiksi ennen siirtymistä seuraavaan toimintoon. Näin pystyttiin vahvistamaan jokaisen palan toiminta kaikissa kehitysvaiheessa. Tämä helpotti virheiden löytämistä ja korjaamista, sillä jokainen muutoksen aiheuttama ohjelman virhetoiminta oli aina juuri lisätyssä koodissa. Tämän etenemismallin kääntöpuoli oli tarve muokata valmiita toimintoja silloin kun uusi toiminto piti sovittaa toimimaan yhdessä aiemmin valmiin toiminnon kanssa.

Toiminnallisuuden kehitys aloitettiin tuottamalla yksikön lisäyskomponenttiin pistearvoja laskeva funktio yhdestä yksiköstä. Tätä varten pistearvojen keräyspalvelu liitettiin sekä yksikön lisäykseen että yksikkökomponenttiin. Palvelu välittää yksikön pistearvon laskemiseen tarvittavat tiedot yksikkökomponentista yksikön lisäyskomponenttiin. Yksikön lisäyskomponentissa tiedot tallennetaan muuttujiin, joita laskukaava funktio käyttää. Peruslaskutoimituksen onnistuessa laajennettiin laskukaava ottamaan huomioon yksikön koon muutokset sekä yksikön lisäoptiot ja -varusteet. Lisäysten toiminnan kannalta ohjelmaan kehitetään pienempiä apufunktioita, joiden tehtävänä on pitää yksikön pistearvo ajan tasalla jokaisen pistearvoon vaikuttavan muutoksen yhteydessä.

Lisäoptioiden ja -varusteiden kohdalla kehityksessä ongelmaksi muodostui pistearvojen päivittyminen yksikön lisäysnäkymässä. Ongelman korjaamiseksi yksikön lisäys- ja yksikkökomponenttien välille jouduttiin luomaan yhteys, jotta valittujen lisäoptioiden ja -varusteiden pistemuutokset näkyisivät ohjelmassa reaaliajassa. Normaalisti tämä pitäisi saada suoritettua palvelun kautta, mutta tässä tapauksessa kyseinen ratkaisu toimi. Ratkaisun toiminnan johdosta paremman ratkaisun löytäminen päätettiin jättää jatkokehityksen puolelle, jolloin toimivan ratkaisun hiominen paremmaksi ei vaikuttaisi demoversion aikatauluun. Koska laskutoimitukset toimivat, laajennettiin logiikka kaikkiin yksikkökomponentteihin.

Seuraavaksi kehityksessä keskityttiin armeijalistan muodostamiseen. Tärkeimmät asiat armeijalistan kannalta ovat tallennettujen yksiköiden tietojen paikkansa pitävyys, tietojen tallennetun muodon sopivuus niiden esittämiseen, ja tietojen tallentaminen armeijalistaan.

Tietojen paikkansa pitävyys on pitkälti kiinni yksikköön käyttäjän tekemistä muutoksista, jotka vaikuttavat yksikön kokoon, pistearvoon sekä lisäoptioihin ja -varusteisiin. Tässä voidaan hyödyntää jo kehitettyä pistearvojen laskemiseen tarkoitettua funktiota. Koska jokainen käyttäjän tekemä muutos aiheuttaa pisteiden uudelleen laskemisen, voidaan laskuihin liittää myös jatkuva tietojen päivitys. Tämä suoritetaan tallentamalla jokainen muutos muuttujiin, jotka päivitetään muutosten yhteydessä. Näin tiedot pysyvät ajan tasalla, kunnes ne tallennetaan armeijalistaan.

Tietojen tallennus sopivaan muotoon voidaan suorittaa keräämällä kaikki tarvittava tieto (yksikön nimi, rooli, koko, kokonaispistearvo sekä lisäoptiot ja -varusteet) taulukkomuuttujaan. Koska tietojen tarvitsee olla taulukko muodossa vain armeijalistaan tallentamisen aikana, voidaan tietojen taulukkoon tallennus ja yksikön tietojen tallentaminen armeijalistaan suorittaa samassa funktiossa. Tätä varten kehitetään funktio, joka ensin hakee tiedot ja tallettaa ne taulukkomuuttujaan. Seuraavaksi tiedot lähetetään esitettäväksi armeijalistakomponentin näkymässä, ja lopuksi tyhjentää taulukkoon tallennetut tiedot. Tämän jälkeen listassa on yksikkö, joka on sijoitettu listaan roolinsa perusteella ja yksikön lisäysprosessi on valmis toistettavaksi.

Armeijalistaan tallennuksen myötä opinnäytetyön toiminnalliset tavoitteet saavutettiin. Ohjelma pystyy laskemaan yksikön pistearvon ja tallentamaan yksikön tiedot armeijalistaan virheittä.

5 Yhteenveto

Opinnäytetyön lopputulos on toimiva armeijalistan rakennusohjelman demoversio, joka täyttää sille asetetut toiminnalliset vaatimukset. Ohjelman avulla voidaan tuottaa ja laskea armeijalista Warhammer 40 000 3. laitokseen. Kokonaisuutena ohjelma todetaan onnistuneeksi.

Tavoitteeksi asetettu käyttöliittymän selkeys ja yksinkertainen käyttökokemus toteutui vaihtelevalla menestyksellä. Ohjelman käyttöliittymän koetaan kommunikoivan käyttäjälle ohjelman toiminnot hyvin, mutta sillä varauksella, että käyttäjä on tutustunut Warhammer 40 000 3. laitoksen armeijalistan rakentamiseen. Käyttöliittymän visuaalinen toteutus on selkeä ja Angular Materialin käyttö antaa sille modernin ilmeen. Kokonaisuutena käyttöliittymässä on myös puutteita, jotka tulevat ilmi paremmin käyttökokemuksen perusteella. Käyttöliittymä on kuitenkin riittävä toteuttamaan armeijalistan luonnin alusta loppuun.

Käyttäjäkokemuksen kannalta suurin negatiivinen osatekijä on armeijalistasta puuttuva mahdollisuus poistaa yksikkö listasta. Ilman poistotoimintoa käyttäjä ei kykene tekemään korjausta virheelliseen tietoon ilman koko listan poistamista ja alusta aloittamista. Toinen ratkaisu virheiden korjaamiseen on lisätä ohjelmaan listassa jo olevan yksikön muokkaus toiminto. Muokkaustoiminnon avulla virheiden korjaaminen olisi nopeampaa, mutta toiminnon kehitys on vaikeammin toteutettavissa kuin poistotoiminto. Siinä missä poistotoiminto yksinkertaisesti poistaa ruudulla näkyvän tekstin, muokkaustoiminnon kanssa pitää ottaa huomioon ohjelman rakenne. Haasteet muokkaustoiminnolle olisivat tietojen palautus muokattavassa muodossa sekä yksikön muokkaukseen käytettävän yksikön lisäysnäkymän toiminta. Molemmat toiminnot päätettiin lisätä jatkokehityksen piiriin.

Käyttäjäkokemuksessa kohdatut epäkohdat olisi voitu paljastaa, jos projektiin olisi sisällytetty testaus. Opinnäytetyön aiheen suunnittelussa testaus oli yksi mahdollisista opinnäytetyön vaiheista, mutta kaksi seikkaa vaikeuttivat sen toteuttamista. Ensiksi oikeudelliset luvat ohjelman kehittämiseen eivät sallineet ohjelman jakamista, minkä vuoksi ohjelmaa ei saa lähettää muille tai laittaa palvelimelle. Luvista johtuen kaikki testaaminen olisi pitänyt suorittaa kehittäjän tietokoneella. Tähän liittyen toinen ongelma testauksessa olisi ollut COVID-19 pandemia. COVID-19 varotoimet tarkoittivat Imatran Miniatyyrikillan toiminnan keskeyttämistä pandemian ajaksi. Toiminnan keskeytyksestä johtuen ohjelmaa ei olisi voitu viedä killan kokoontumisiin testattavaksi, sillä kokoontumisia ei ollut. Näistä seikoista johtuen, testaaminen päätettiin jättää opinnäytetyön ulkopuolelle. Kuitenkin ohjelmasta löytyneet käyttäjäkokemuksen epäkohdat ovat selkeä osoitus testauksen tarpeellisuudesta ohjelmistokehityksessä.

Opinnäytetyön tavoitteisiin kirjattu ohjelman luonti Angularilla ilman tietokantaa onnistui. Tietokanta pystyttiin korvaamaan luomalla tarvittavat tiedot Angularin komponenteissa. Komponenttien käyttö tähän tarkoitukseen paljastui yllättävän suoraviivaiseksi ja varsinaisen koodin kirjottamisen kannalta helpoksi. Tietokanta on järkevä vaihtoehto esimerkiksi silloin, kun ohjelma käyttää samoja tietoja useammassa komponentissa. Ohjelman kannalta tietokanta voitaisiin valjastaa esimerkiksi yksikön tietojen, varusteiden ja muiden lisäoptioiden kohdalla. Kuitenkin tällainen menettely vaatisi ohjelmaan lisättäväksi useita tietokantoja, sillä useimmat armeijalistat eroavat esimerkiksi varusteiden ja lisäoptioiden puolesta toisistaan. Lisäksi armeijalistat, jotka käyttävät osittain samoja varusteita voivat erota toisistaan varusteiden pistearvoissa. Tästä syystä ohjelmassa käytetty rakenne yksikön tietojen, varusteiden ja lisäoptioiden kohdalla on armeijalistan rakennusohjelmaan sopivampi.

Ohjelman kehittäminen käyttäen pääosin vain Angularin ohjelmistokehystä koettiin erittäin joustavaksi. Tästä kehitysmallista havaittiin, että rajoitetulla määrällä työkaluja pystytään pääsemään toimiviin tuloksiin. Tämän lisäksi rajoitukset luovat myös tilanteita, joihin tarvitaan luovia ratkaisuja. Luovista ratkaisuista hyvä esimerkki on yksikkökomponentti. Tietokannan puuttuessa tarvitaan yksikön tiedoille jokin säilytystapa ja samalla ratkaistaan yksikköjen vaihtuva lisäoptioiden ja varusteiden listaus. Tämän työn perusteella voidaan todeta, että työlle valittu kehitysmalli toimi.

6 Jatkokehitys

Ohjelman suunnittelun ja kehittämisen yhteydessä löytyi useita jatkokehitysmahdollisuuksia. Ensimmäinen selkeä jatkokehityksen suunta on parantaa käyttökokemusta. Vaikka ohjelman suorittamat laskutoimitukset ja armeijalistan luominen toimivat sujuvasti, yhdenkin virheen tekeminen listaa luodessa tarkoittaa koko listan rakentamisen aloittamista alusta. Tästä syystä jatkokehityksen ensimmäinen prioriteetti on tämän epäkohdan korjaaminen, etenkin listasta poistotoiminnon kehittäminen.

Myös ohjelman sisällön laajentaminen on selvä jatkokehityksen kohde. Warhammer 40 000 3. laitos sisältää yhteensä kaksitoista erilaista armeijalistaa, useita näille suunnattuja variantti listoja sekä muutaman yhdistelmä listan. Jatkokehityksessä suurin osa listoista on tarkoitus lisätä ohjelmaan demoversion luoman rakenteen pohjalta. Sisällön laajentuessa ohjelman tulisi tukea myös laajempaa navigaatio rakennetta, johon sisältyy toteutusvaiheessa leikattu päävalikko.

Lisäksi ohjelman koodipohjan siistiminen ja parempien ratkaisujen löytäminen voidaan lisätä yhdeksi jatkokehityksen haaroista. Ohjelma toimii tarkoituksensa mukaisesti, mutta joihinkin ohjelman kehityksessä kohdattuihin ongelmiin on olemassa parempia ratkaisuja. Nämä ratkaisut tulisi löytää ja toteuttaa koodissa sen luettavuuden parantamiseksi. Parempi koodin luettavuus on tärkeää erityisesti silloin, jos kehitysprosessissa on useampi kuin yksi henkilö.

Jatkokehitykseen kuuluu myös selvitys ohjelman mahdollisesta lisensoinnista. Lisenssin avulla ohjelma voitaisiin täysversion kehittämisen myötä laittaa julkiseen jakeluun. Lisensoinnin kautta voitaisiin mahdollisesti hakea lähdemateriaalin laajempaa käyttöä. Tämä sisältäisi yksiköiden pelillisten attribuuttien kopioimista osaksi ohjelman käyttämiä tietoja. Tavoitteena lisäykselle ohjelman käytöllä voitaisiin käytännössä korvata pelissä käytetyt 90- ja 2000- luvuilla loppuunmyydyt kirjat.

Ohjelman rakenteen puolesta, ohjelma voitaisiin kääntää toimimaan myös muiden Warhammer 40 000 laitosten kanssa. Muutokset laitosten välillä tarkoittaa pääsääntöisesti yksikkötietojen muuttamista, etenkin pistearvojen kohdalla. Jotkin laitokset vaatisivat myös rakenteellista kehitystä, kuten esimerkiksi 7. laitoksessa lisätyt yksikköformaatiot. Ohjelmaa voitaisiin myös käyttää pohjana muiden vastaavanlaisten pelien armeijalistan rakennusohjelmia varten. Näissä tapauksissa ohjelman nykyinen yksiköiden roolipohjainen kategorisointi tulisi muuttaa sopimaan uuden pelin armeijalistan rakennusmalliin. Yksiköiden pistearvojen laskufunktio sen sijaan selviäisi hyvin pienellä määrällä muutoksia, sillä pelien välillä on yleensä hyvin vähän eroa pistearvojen laskutavoissa.

Lähteet

BattleScribe, 2021. A fast and powerful army list creator for tabletop wargamers. Viitattu 26.5.2021. Saatavilla https://battlescribe.net/?tab=news

Games Workshop Group, 2021. Intellectual property policy. Viitattu 27.5.2021. Saatavissa https://www.games-workshop.com/en-EU/Intellectual-Property-Policy

Git, 2021. Git. Viitattu 27.5.2021. Saatavissa https://git-scm.com/

GitHub, Inc., 2021. Github. Viitattu 27.5.2021. Saatavissa https://github.com/

Angular, 2021. Introduction to Angular Concepts. Oppaat. Viitattu 25.5.2021. Saatavissa https://angular.io/guide/architecture

Angular Material, 2021b. Angular Material. Viitattu 26.5.2021. Saatavissa https://material.angular.io/

JetBrains, 2021a. WebStorm, the smartest JavaScript IDE. Viitattu 26.5.2021 Saatavissa https://www.jetbrains.com/webstorm/

JebtBrains, 2021b. Kanban tools for agile teams. Viitattu 30.5.2021. Saatavissa https://www.jetbrains.com/youtrack/agile/kanban/

JetBrains, 2021c. Kanban board.Viitattu 30.5.2021. Saatavissa https://www.jetbrains.com/youtrack/agile/kanban/img/screenshots/kanban_board.png

npm, Inc., 2021. npmjs. Viitattu 27.5.2021. Saatavissa https://www.npmjs.com/

OpenJS Foundation, 2021. NodeJS. Viitattu 27.5.2021. Saatavilla https://nodejs.org/en/

Tekijänkoikeuslaki 61/404. Suomen laki. Viitattu 26.5.2021. Saatavissa https://www.finlex.fi/fi/laki/ajantasa/1961/19610404

Tietosuoja-asetus 2016/679. Euroopan parlamentin ja neuvoston asetus. Viitattu 26.5.2021. Saatavissa https://eur-lex.europa.eu/legal-content/FI/TXT/HTML/?uri=CELEX:32016R0679&from=FI

Warhammer Community, 2021. Combat Roster. Viitattu 30.5.2021. Saatavissa https://www.warhammer-community.com/combat-roster/