Tietotekniikan ohjelmointityö: PoEMap

Emil Keränen, Jyväskylän yliopisto

Itsearviointi ja jälkiselvitys

Sisällysluettelo

Tiivistelmä

PoEMap on yksinkertaisesti selitettynä virtuaalikauppa kartta-esineille web-sovelluksen muodossa Path of Exile -videopeliä varten. Karttojen ostaminen ja myyminen tapahtuu pelin sisäisillä virtuaalivaluutoilla, joten työssä ei tarvitse käsitellä oikeita rahatapahtumia.

Sovellus on tehty *Microsoft Visual Studio 2017* -ohjelmalla C#-ohjelmointikielellä hyödyntäen Microsoftin .NET Core -kehystä. Pohjana on Visual Studion oma *web-application* -malli, joka luo ohjelman lokaaliin isännöintiin vaadittavat tiedot sekä muutamia esimerkkisivuja.

Kuvaus, tausta ja tavoitteet

Path of Exile on toiminta-rooli (*action role-playing*) -peli, jossa tarkoituksena on taistella kymmeniä ellei satoja vihollisia vastaan samanaikaisesti. Kyseessä on siis hyvin nopeatempoinen peli, joka kiteytyy tarinan jälkeen loppupelin kartta-tavaroihin, joilla luodaan vaikeita, väliaikaisia ja usein hyvin tuottavia alueita verrattuna tarinassa käytyihin alueisiin.

Kartat ovat siis hyvin merkittävä osa kyseisen videopelin loppupelin kulkua, jonka takia sovellukseni keskittyy ainoastaan niiden ostamiseen ja myymiseen. Kaikki tapahtuu virtuaalivaluutoilla eikä ostaminen ole automatisoitua, vaan sovelluksestani löytyy myyjän nimi ja hinta, joiden avulla voi pelin sisällä esimerkiksi kuiskata kyseiselle pelaajalle.

Päätavoitteeni oli luoda kauppapaikka, joka on yksinkertainen käyttää ja josta pystyy hyvin nopeasti ostamaan haluamansa kartan. Onnistuin mielestäni melko hyvin, mutta käytännössä ohjelma on hyvin raskas ylläpitää enkä usko ottavani sitä käyttöön. Ohjelma vaatisi myös paljon jatkokehitystä, jota en ainakaan lähiaikoina pysty toteuttamaan.

Käytännön toteutus

Sovelluksen visuaalinen kokonaisrakenne löytyy Github-sivultani, mutta kirjoitan rakenteen lyhyesti myös tähän dokumenttiin.

Ohjelman toiminnan mahdollistaa Path of Exile -pelin tekijöiden, Grinding gear games -yhtiön, käyttöönottama *Public stash tab API*, joka antaa tiedon kaikkien pelaajien julkisten kätköjen (stash) sisällöistä. Kätköt ovat siis pelin pankki, jonne voi laittaa haluamiaan tavaroita talteen. *Public stash tab API* jakaa tiedon tietyin väliajoin json-muodossa sivustolla (http://www.pathofexile.com/api/public-stash-tabs), josta kuka tahansa voi käydä katsomassa tai käyttämässä sitä haluamallaan tavalla. API toimii “update-stream” periaatteella eli jokainen uusi päivitys seuraavan id:n avulla sisältää tietyn määrän muuttuneita kätköjen sisältöjä. Jos muuttuneita kätköjä on liikaa, niin ne näkyvät seuraavassa päivityksessä.

Ohjelma parsii saadun JSON-tiedon *Newtonsoft*-kehyksen avulla, joka muuttaa ne C#:in tunnistamiksi olioiksi. Tämän jälkeen tarvittavat tiedot ja niiden suhteet voidaan määrittää ja lisätä ne paikalliseen SQLite-tietokantaan. Olioiden suhteet toimivat seuraavasti: pelaajalla on yksi tai useampi *stash* eli kätkö, jossa voi olla yksi tai useampia *mappeja* eli karttoja. Jokaisella kartalla on yksi *currency*- eli valuutta-olio, joka pitää sisällään tiedon kartan hinnasta eli käytetystä valuutasta ja sen määrästä. Hakusivu käyttää tätä olemassaolevaa tietokantaa kyselyiden luomiseen hakusuodattimien perusteella.

Oman työn arviointi

Ohjelmointityön tekeminen alkoi marraskuun 2018 aikoihin, jolloin tein ohjelman alustavan rakenteen. Aluksi tarkoitus oli ottaa API:sta vain kartat tietokantaan ja asettaa ominaisuudet, kuten myyjä ja hinta, suoraan kartta-olioon, mutta tajusin hetken kuluttua, että tarvitsen myös kätköjen tiedot. Lisäsin kätköt rakenteeseen ja jatkoin työn tekemistä.

En käyttänyt aluksi mitään tietokantaa, joten seuraavaksi ongelmaksi muodostui tietokannan käyttö. Ohjelman tietokannassa tapahtuu hyvin paljon muutoksia (lisäyksiä, poistamista ja korvaamista) ja kyselyitä ja se vaatii osien välisiä relaatioita, joten kokeilin aluksi Microsoft Azuren SQLServer-tietokannan käyttöä. Tämä olisi vaatinut *ConnectionStringin* käyttöä, joka vaikutti hieman riskialttiilta näin yksinkertaiseen työhön. *ConnectionStringiin* olisi joutunut tallentamaan käyttäjätunnuksen ja salasanan ja salaamaan ne, joten päädyin lopulta käyttämään vain SQLiten lokaalia tietokantaa.

Ohjelmointityön tekemisen aikana muutin rakennetta useaan otteeseen vähän liian hätiköidysti. Lisäsin kesken kaiken mm. Currency-olion pitämään yllä tietoa hinnasta, mutta myöhemmin muutin sen staattiseksi, joka hoiti vain hinnan parsimiset. Kohtasin muutamia ongelmia sen suhteen, joten palautin sen jälleen olioksi. Jos aloittaisin työn alusta, suunnittelisin sen paljon paremmin ja tarkemmin sekä noudattaisin alkuperäistä rakennetta ellei jokin todella hyvä ratkaisu tulisi eteen.

Alkuraportissa mainitsin haasteiksi mm. järkevän ja nopean JSON-tiedon purkamisen sekä hinnan mukaisen lajittelun, koska eri valuuttoja on jopa kymmeniä ja niiden arvo on täysin pelaajien määriteltävissä. JSON-tiedon purku on mielestäni toimiva, mutta vie yllättävän paljon muistia, koska jokaisesta kätköstä, kartasta ja hinnasta luodaan oma olionsa. Hintojen järjestely perustuu tällä hetkellä raakoihin arvioihin, eli ne eivät ole täysin varmoja. Jätin siis toteuttamatta esimerkiksi arvojen reaaliaikaisen noutamisen (https://poe.ninja)-sivustolta.

Ohjelmointityö on toteutettu siis Microsoftin .NET Core -kehyksen avulla, eli suurin osa käytetystä ajasta meni Microsoftin dokumentaatiota ja ohjeita lukiessa. Ohjeet olivat hyvin selkeitä ja sisälsivät paljon esimerkkejä, joiden avulla pääsin hyvin työssä liikkeelle. Hyödynsin myös muiden käyttäjien neuvoja ja ohjeita, jos vastaan tuli ongelmia. Muiden ohjelmointikoodia en kuitenkaan suoraan käyttänyt. Alkuperäisessä raportissa mainitsin Javascriptin, HTML ja CSS käytön, mutta en lopulta käyttänyt näistä muita kuin HTML todella pienissä määrin.

Olen mielestäni noudattanut melko hyvin ohjelmointikäytäntöjä työssäni. Kommentointini on ollut selkeää ja olen saanut tehtyä uudistuksia pienissä osissa siten, että ohjelma on aina kuitekin toiminut. Lisäksi olen debuggauksen avulla selvitellyt ohjelman kompastuskohtia, kuten muistin käyttöä, ja yrittänyt parhaani mukaan etsiä lisätietoa vastaavista ongelmista ja korjata niitä. Miinusta antaisin itselleni ainakin testauksen puutteesta. En ole tehnyt minkäänlaisia automaattisia testauksia, vaan olen debuggauksen ja käytön perusteella testannut ohjelman toimivuutta. Lisäksi alkuperäinen suunnitelmani oli hieman hätiköidysti tehty.

Aikatauluni ei pitänyt ollenkaan paikkaansa. Tarkoitus oli saada työ valmiiksi tammi-helmikuun vaihteessa 2019, mutta en millään saanut työtä palautuskelpoiseksi siihen mennessä. Lopulta palautus venyi huhtikuulle 2019. Oikealla suunnittelulla ja perehtymällä asioihin ajoissa olisin saanut tehtyä työn paljon aiemmin ja tehokkaammin. Nyt tuntimäärä ylittyi melko paljon verrattuna alkuperäiseen arvioon.

Tietotekniikan perusopintojen luennoilta opin hyvät perusteet ohjelmointiin, mutta syvempi oppiminen tapahtui mielestäni tämän ohjelmointityön aikana. Koin oppivani todella paljon uutta varsinkin .NET-kehyksen ympäristöön liittyen, joka parantaa esimerkiksi tulevaisuuden työmahdollisuuksiani runsaasti. Nämä asiat eivät kuitenkaan olleet luennoilla läsnä, joten kokemukseni eivät vastanneet täysin luennoilla käsiteltyjä asioita.

Tällä hetkellä ohjelma toimii ensimmäisellä ajamiskerralla moitteettomasti, mutta ohjelman uudelleenkäynnistyminen saattaa aiheuttaa SQLite-tietokantavirheen. Tämä virhe liittyy karttojen uniikki-id -rikkomukseen, joka todennäköisesti syntyy, kun tietty kartta on tallennettu tietokantaan, mutta seuraavalla API:n hakukerralla kartta on kerennyt jo vaihtaa omistajaa ja esiintyy uuden omistajan kätkössä.

Työssä käytetyt lähteet

Ohjelmointityö on pääasiassa tehty Microsoftin dokumentaation ja tutoriaalien avuin (https://docs.microsoft.com/en-us/). *Stash Tab API:n* toiminnallisuuksien ja rakenteen selvittämiseen käytin Path of Exilen omaa Wiki-sivustoa (https://pathofexile.gamepedia.com/Public_stash_tab_API). Näiden lisäksi käytin apuna monia eri käyttäjäfoorumeita, kuten Stack Overflow- ja Reddit-sivustoja (https://www.reddit.com/r/pathofexiledev/comments/48i4s1/information_on_the_new_stash_tab_api/).

Ylläpitoon vaaditut tiedot ja ohjeet

Ohjelma on melko raskas ylläpitää varsinkin muistin suhteen ja sen olisi tarkoitus olla käynnissä jatkuvasti, jotta tavaroista tai kartoista saadaan reaaliaikainen tieto. Muuten tietokannassa tulee esiintymään “haamutavaroita”, joita ei välttämättä ole enää olemassa tai myynnissä. Tietokantatiedosto tulee olemaan pahimmillaan useiden gigatavujen kokoinen, joten siihen täytyy myös varautua. Ohjelmaa ei ole testattu pitkillä aikaväleillä eli sen suoriutuvuudesta ei ole takuita, jos tietokanta on hyvin suuri.

Sovelluksen mahdollinen jatkokehitys

Sovellus on tällä hetkellä todella niukka, joten jatkokehitys on jopa suotavaa. Tärkeimpänä kehityskohteena on ottaa huomioon Path of Exilelle tyypillinen “karttakierrätys” eli kartat vaihtavat päivitysten yhteydessä tasoaan. Tämä pitäisi ottaa huomioon ohjelmassa, mutta tällä hetkellä se vaatisi päivitystä kolmen kuukauden välein, jolloin uusi päivitys usein tulee.

Toisena kehityskohteena on hakusuodattimien lisääminen ja kaikkien kartta-olion tietojen tallentaminen. Tällä hetkellä ohjelma tallentaa vain muutamat tarpeelliset ominaisuudet. Hakufiltterien lisäämisen yhteydessä ohjelman julkisivua voisi myös päivittää järkevämpään suuntaan, koska nyt se on tehty todella pelkistetysti valmiita muotoja käyttämällä.

Yhteenveto

PoEMap-ohjelmointityö perustuu lyhyesti *Public Stash Tab API:n* kulutukseen ja tästä saatavan tiedon parsimiseen ja tallentamiseen. Tallennettu tieto saadaan nopeasti SQLite-tietokannasta käyttäjän asettamien suodattimien avulla. Ohjelman olisi tarkoitus olla käynnissä jatkuvasti, jotta käyttäjä voisi ostaa reaaliajassa halvimmat tarvitsemansa kartat, mutta tällä hetkellä ainakaan minulla ei ole resursseja pitää ohjelmaa käynnissä täyspäiväisesti. Ohjelmointityö toimi siis pääasiassa todella hyvänä käytännön oppituntina Microsoftin teknologioiden, kuten .NET-kehyksen ja *Entity Frameworkin*, käytöstä ja hyödyntämismahdollisuuksista.