Lauri Lahtiranta

INTERAKTIIVINEN KARTTA: SUUNNITTELU JA KEHITTÄMINEN

Tietojenkäsittelyn koulutusohjelma

2018

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

INTERAKTIIVINEN KARTTA: SUUNNITTELU JA KEHITTÄMINEN

Lahtiranta, Lauri
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Helmikuu 2018
Ohjaaja: Nieminen Hans
Sivumäärä: 34
Liitteitä: 0

Asiasanat: karttapalvelut, verkko-ohjelmointi, verkkopalvelut

---

Työssä käydään läpi interaktiivisen kartan sovelluskehitystä. Suunnittelussa katsotaan mitä työkaluja sovelluksen tuottamiseen käytetään, vaatimusmäärittelyä, mitä hyvän sovelluksen luominen vaatii sekä tämän sovelluksen suunnittelun eri vaiheita. Toteutus osiossa käydään läpi ohjelman toiminnallisuus sekä mitä haasteita ja oppimista sovelluksen kehittämisen aikana tuli. Vertailussa käyn läpi osittain mallina toimineen HUSin interaktiivisen kartan eroja omaan kartta sovellukseeni.

INTERACTIVE MAP: DESIGN AND DEVELOPMENT

Lahtiranta, Lauri
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Technology
February 2018
Supervisor: Nieminen Hans
Number of pages: 34
Appendices: 0

Keywords: mapinfo, map services, programming

In this work we will go through interactive map software development. In design we will look at what tools were used to develop this interactive map, software requirements specification, what it takes to create a good application and different phases of design in for this program. In implementation section we will go through the functionality of the program and what challenges and studies I met during the development of this program. In comparison I will measure my interactive map program against HUS interactive map and see what differences it has compared to the interactive map I made.

SISÄLLYS

1 JOHDANTO

Tarkoituksenani on luoda interaktiivinen kartta, jota muokkaamalla lopullinen sovellus tehdään. Interaktiivinen kartta on tilattu Satakunnan sairaanhoitopiiristä Medbit Oy:ltä ja tulisi koskemaan Satakunnan keskussairaalaa, sekä mahdollisesti Turun yliopistollista keskussairaalaa.

Medbit on yksi Suomen suurimmista julkisomisteisista sosiaali- ja terveysalan-ICT ratkaisutoimittajista. Medbit tarjoaa keskitettyjä ja ulkoistettuja ICT-palveluja sosiaali- ja terveydenhuollon ammattilaisten arjen helpottamiseksi.

Interaktiivisen kartan tarkoitus on tarjota parempia toimintoja kuin perinteinen karttakuva. Interaktiiviset ominaisuudet tarjoavat enemmän informaatiota käyttäjälle kuin perinteinen kartta. Esimerkiksi käyttäjä voi napsauttaa haluamaansa kohdetta, jolloin avautuu uusi ikkuna, jossa on lisätietoja tai jolloin kohde siirtää käyttäjän näkymän uuteen sijaintiin.

Työssä käydään läpi ohjelman tekoon käytetyt teknologiat, suunnittelu ja eri ohjelman versiot, miten se toteutettiin ja lopuksi verrataan sitä HUSin (Helsingin ja Uudenmaan sairaanhoitopiiri) käyttämään interaktiiviseen karttaan. Kehitystyökalut ja teknologiat käydään lyhyesti läpi, mitä kukin teknologia on ja mihin sitä tarvitaan. Suunnittelussa kerron, miten ohjelma suunniteltiin ja miten se kehittyi eri versioittain. Toteutuksessa katsotaan, kuinka ohjelma toimii. Lopuksi vertaan ohjelmaa HUS käyttämään interaktiiviseen karttaan, joka toimi eräänlaisena esimerkkinä sovellukselle.

Omana tarkoituksenani on oppia paremmaksi HTML-, JavaScript- ja CSS-ohjelmoijaksi sekä paremmin ymmärtää, mitä Web-sivujen suunnittelu eri laitteille vaatii. Lisäksi toimittaa toimiva interaktiivinen kartta Medbitille ja sen asiakkaalle.

2 SUUNNITTELU

2.1 Kehitystyökalut ja teknologiat

Aluksi käytin sovelluksen pohjana vain HTML-, CSS- ja JavaScript-tekstitiedostoja. Tekstitiedostopohjan käyttö oli minulle aluksi ketterää, kun halusin nopeasti testata useita eri kirjastojen toimintoja. Tein pieniä muokkauksia tiedostoon samalla, kun opiskelin kieltä ja tutustuin sen kirjastoihin. Kun minulla alkoi olla jokin pohja, jonka päälle rakentaa sovellus, siirsin projektin Visual Studio 2017 Community Edition-ohjelmistoon.

Ohjelmointikielinä käytin yleisesti Web-suunnittelussa käytettyjä kieliä HTML, CSS ja JavaScript. JavaScript-kirjastoina käytin Jqueryn lisäksi Semantic ui sekä Konvajs.

2.1.1 Visual Studio

Microsoftin kehittämä Visual Studio on integroitu kehitysympäristö. Sitä käytetään tietokonesovellusten, Web-sivujen, Web-sovellusten, Web-palveluiden ja mobiilisovellusten kehittämiseen. Visual Studiossa on IntelliSenseä tukeva koodi editori sekä työkalut koodin optimointia varten. Sisään rakennettu debuggeri toimii sekä lähde että kone tason debuggerina. Visual Studio tukee useita eri ohjelmointikieliä ja sen koodi editori sekä debuggeri tukevat vaihtelevasti lähes jokaista ohjelmointi kieltä. Visual Studio otettiin käyttöön sovelluksen versioinnin, virheidenkorjaustyökalujen, sekä sen muiden ohjelmisto kehitystä helpottavien ominaisuuksien vuoksi. (Wikimedia Foundation, Inc., 19.2.2018)

2.1.2 JavaScript

JavaScript on Netscapen kehittämä dynaaminen ohjelmointikieli, joka on tarkoitettu enimmäkseen Web-ympäristöön. JavaScript onkin CSS ja HTML kanssa tärkeä osa WWW-sivustojen sisällön tuotannossa. Sitä käytetään interaktiivisten Web-sivujen sekä pelien tuottamiseen. Suurin osa maailman netti sivuista käyttää JavaScriptia ja kaikissa moderneissa selaimissa se on sisäänrakennettu. Koko sovellus perustuu JavaScriptiin. Ilman JavaScriptiä sovellus ei toimi. (Wikimedia Foundation, Inc., 9.5.2017)

2.1.3 HTML ja Canvas

HTML on lyhenne sanoista Hypertext Markup Language. HTML-normia ylläpitää vuonna 1994 perustettu kansainvälinen yritysten ja yhteisöjen yhteenliittymä World Wide Web Consortium. HTML on standardoitu ohjelmointikieli Web-sivujen ja Web-sovellusten tuotannossa. Web-selaimet saavat HTML dokumentteja Web-palvelimilta tai paikalliselta levyltä ja renderöivät sen multimedia Web-sivuiksi. HTML-elementit ovat HTML-sivujen rakennuspalikoita. (Wikimedia Foundation, Inc., 3.2.2018)

Canvas elementti on kaksiulotteisten muotojen renderöinti alusta. Itse Canvas koostuu piirrettävästä alueesta, jolla on korkeus ja leveys. JavaScript koodilla voidaan käyttää Canvas aluetta ja sen piirtämisominaisuuksia, jolla voidaan luoda dynaamisesti generoitua grafiikkaa. Sitä käytetään muun muassa taulukoiden, animaatioiden, pelien ja kuvien muodostamisessa. Canvas on alun perin Applen Mac OS X WebKit komponentti, jota käytettiin Applen Dashboardin widgetteihin sekä Safari selaimeen. Canvas kartta sovelluksessa luo piirtämispohjan, jonka päälle muotoja voidaan piirtää. (Wikimedia Foundation, Inc., 13.2.2018)

2.1.4 CSS

CSS eli Cascading Style Sheets on kieli, jota käytetään HTML-sivujen ulkoasun kuvaamiseen. CSS on suunniteltu erottamaan rakenne ja sisältö toisistaan, kuten sijoitus, värit ja fontit. Tämä erittely parantaa sisällön käytettävyyttä, tarjoaa joustavamman ja paremman kontrollin esittelyn pääpiirteisiin, antaa usean HTML-sivun jakaa saman muotoilupohjan erilliseen .css-tiedostoon ja vähentää monimutkaisuutta ja toistoa rakenteelliselta sisällöltä. Muotoilun ja sisällön erottaminen mahdollistaa saman HTML-sivun näyttämisen eri tyyleillä tai näkymään eri tavoin eri laitteilla vaikuttaen kuvan kokoon. CSS puolelta tulee sovellukseen tyylittelyt sekä animaatioita. (Wikimedia Foundation, Inc., 20.2.2018)

2.1.5 JSON

JSON eli JavaScript Object Notation on avoin standardi tiedosto tyyppi joka lähettää data olioita jotka koostuvat atribuutin arvopareista ja taulukon data tyypeistä. Se on tavallinen dataformaatti, jota käytetään asynkronisessa selain-palvelin kommunikaatiossa, sekä vaihtoehtona XML ja Ajax järjestelmille. JSON on itsenäinen data formaatti. Se on kehitetty JavaScriptistä, mutta 2017 lähtien monet ohjelmointi kielet ovat sisältäneet koodia jolla generoida ja jäsentää JSON formaatin dataa. Kaikki sovelluksen sijainti-, osasto- ja rakennustiedot ovat JSON tiedostoissa. (Wikimedia Foundation, Inc., 25.11.2018)

2.1.6 Konva

Konva on Canvas-elementin käyttöön tehty JavaScript-kirjasto, joka tarjoaa korkean suoritus kyvyn animaatioissa, transitioissa, työpöydän ja mobiililaitteiden tapahtumien käsittelyssä ja monessa muussa. Konva on Canvaksen yksi monista kirjastoista, jotka antavat sille uusia ominaisuuksia ja helpottavat sen ominaisuuksien käyttöä. Muita vastaavan tyyppisiä kirjastoja, kuten Konva, on muun muassa Fabric.js, Paper.js sekä Easeljs. Itse päädyin käyttämään Konvaa sen hyvän dokumentaation ja tutoriaalien vuoksi. Tutoriaalien avulla opin nopeasti, kuinka Konvan useita toimintoja käytettiin. (Konva, 5.2.2018)

2.1.7 Semantic

Semantic UI on JavaScript- ja CSS-kirjasto. Se tarjoaa kehittäjille valmiita osia websivujen rakentamiseen. Semantic on suunniteltu tukemaan monia laitteita antaen Web-sivujen skaalautua. Semantic on tuotantovalmis ja yhteensidottu useiden ohjelmointikehyksenä käytettävien JavaScript-kirjastojen kanssa. Se voidaan integroida esimerkiksi seuraaviin kehyksiin kuten React, Angular, Meteor ja Ember. Sovelluksessa Semantic luo pohjan valikoille sekä muulle käyttöliittymälle. (Semantic, 20.2.2018)

2.2 Vaatimusmäärittely

Ohjelmiston vaatimusmäärittelyssä luodaan yhdessä asiakkaan kanssa dokumentti, jossa kuvataan ohjelmisto projektin vaatimuksia ja tavoitteita. Dokumentissa määritetään miten lopullisen ohjelmiston pitäisi toimia ja millä tavoin nämä toiminnallisuudet voidaan saavuttaa. Vaatimusmäärittely jaetaan kahteen osaan toiminnallisiin ja ei- toiminnallisiin vaatimuksiin, joista ei- toiminnallinen koostuu laadullisista- ja resurssivaatimuksista. (Wikimedia Foundation, Inc., 9.5.2017).

Tähän sovellukseen virallista vaatimusmäärittelyä ei tehty. Asiakas pyysi sovelluksen näyttävän ja omaavan samoja ominaisuuksia kuin HUSin interaktiivinen kartta. Lisätoiveina oli interaktiivisuus myös kartan ja nimikenttien välillä sekä rakennusten tietokentät, jotka sisältäisivät kuvan kohteesta sekä sen tietoja. Uusia haluttuja ominaisuuksia tulisi sitä mukaan, kun projekti etenisi ja asiakas saisi uusia prototyyppimalleja testattavaksi. Medbitin puolelta toiveena oli, että sovelluksella olisi hyvä uudelleen käytettävyys pohjana toisiin interaktiivisiin karttaratkaisuihin.

2.3 Käyttöliittymän perusteet

Käyttöliittymän tarkoitus on ennakoida mitä käyttäjä haluaa tehdä ja varmistaa että käyttäjällä on siihen helppo pääsy ja ymmärrys. Käyttöliittymä tuo yhteen käytettävyyden suunnittelun, visuaalisen suunnittelun sekä informaatio arkkitehtuurin.

Käyttäjät ovat tottuneet tietyn tyyppisin käyttöliittymiin ja miten ne käyttäytyvät. Käyttöliittymä tulisi suunnitella siten että käyttäjä osaa ennakoida mitä tapahtuu, kun hän käyttää tiettyjä toimintoja. On tapauksia joissa usea elementti voi olla hyvä ratkaisu sisällön esittämiseen. Kyseisissä tapauksissa kannattaa miettiä mitä niissä voidaan myös menettää. Esimerkiksi joskus elementit jotka säästävät tilaa voivat asettaa suuremman taakan käyttäjälle, jos käyttäjä joutuu arvailemaan mitä alasvetovalikko sisältää.

Tärkeää on ymmärtää käyttäjiä, mitä käyttäjä haluaa saavuttaa, kuinka hyvin osaa käyttää sekä tottumukset ja suosimiset. Kun ymmärtää käyttäjän tarpeet tulee huomioida seuraavia asioita käyttöliittymän suunnittelussa. (usability.gov, 21.5.2014).

Hyviä tapoja käyttöliittymän suunnittelussa

Seuraavassa listaa hyvistä käyttöliittymän suunnittelussa käytettävistä tavoista.

-**Pidä käyttöliittymä helppona ymmärtää**. Parhaat käyttöliittymät ovat lähes näkymättömiä käyttäjälle. Ne välttävät turhia elementtejä ja käyttävät selvää kieltä tekstikentissä ja viesteissä.

-**Luo yhtenäisyyttä ja käytä tuttuja käyttöliittymä elementtejä**. Kun käyttää tunnettuja elementtejä käyttöliittymässä se tuntuu mukavammalta ja asiat tulee hoidettua nopeammin. Kun käyttäjä oppii käyttämään jotain hänen tulisi pystyä käyttämään samaa opittua asiaa muualla sovelluksessa.

-**Tarkoituksen mukainen malli**. Mieti elementtien välisiä yhteyksiä sekä rakennetta. Elementtien tarkka asettelu voi auttaa huomion keräämisessä tärkeimpiin kohteisiin sekä luettavuutta.

-**Käytä oikein värejä ja tekstuuria**. Huomion suuntaamisessa kohteeseen värit, tekstuuri ja efektit ovat avuksi.

-**Käytä typografiaa luodaksesi hierarkiaa ja selkeyttä**. Harkinta fonttien tyypeissä, koossa ja järjestelyssä vaikuttaa luettavuuteen ja ymmärrettävyyteen.

-**Pidä huolta, että järjestelmä kertoo mitä tapahtuu**. Aina tarjoa informaatiota käyttäjille toiminnoista, sijainneista, virheistä tai muusta tarpeellisesta mitä sovelluksessa tapahtuu.

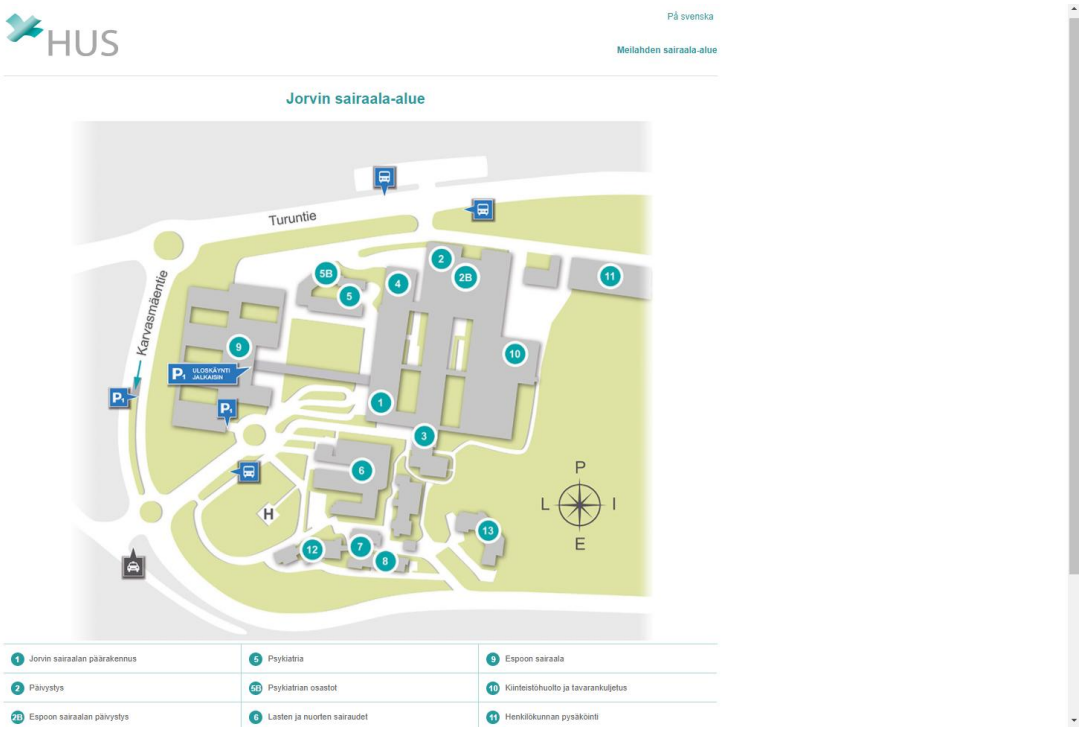
-**Mieti oletusarvoja**. Ajattele ja ennakoi, mitä ihmiset haluavat saavuttaa sovellusta käyttäessä. Suunnittele kentät ja valikot siten, että niissä on oletuksena se, mitä käyttäjä todennäköisesti tarvitsee. (usability.gov, 21.5.2014).

2.4 Sovelluksen käyttöliittymä

Alkuperäisenä projektimallina toimi Satakunnan sairaanhoitopiirin sivuilta löytyvä PDF-muotoinen kartta. Tämän kartan pohjalta rakennettiin(ks. kuva 2) image-maps -pohjaisen interaktiivisen karttaprototyypin. Image-maps on kuvalinkitystä. Kuvaan luodaan alueita, joille muodostetaan linkkejä. Tämän prototyypin pohjalta rupesin kehittämään omaa kartta sovellustani. Prototyyppi piirsi kartalle kuvan rakennuksesta tai ovesta, kun osastoa, poliklinikkaa tai jotain muuta nimeä napsautti valikosta. Tämä myös toimi toiseen suuntaan eli jos rakennusta painoi kartalla, se korosti valikos-

ta ne osastot, jotka kuuluivat rakennukseen. Tästä ratkaisusta käytin samaa keinoa linkittääkseni muodot (rakennukset ja ovet) valikon teksteihin ja tietokenttiin.

Alkuperäisen ratkaisun ongelma oli kuitenkin sen soveltuvuus mobiililaitteille, sen uudelleen käytettävyys ja mahdolliset lisäinteraktiivisuudet, kuten tietokentät muoto- ja painaessa. Muina ongelmina oli karttakuvan muoto, joka oli vertikaalisti useita kertoja pidempi kuin horisontaalisesti. Ohjelmasta tuli myös luoda englanninkielinen versio, joten kaksikielisyys tuli olla mukana alusta asti suunnittelussa. Sovelluksen käyttöliittymän mallina toimii HUSin Jorvin sairaalan alueen kartta (ks. kuva 1).



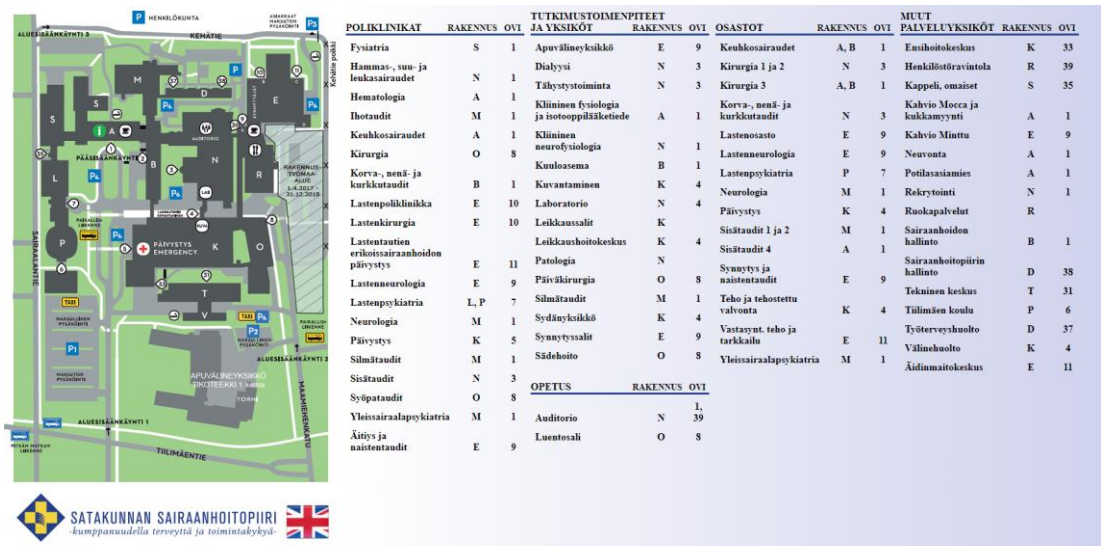
Kuva 1 HUS Jorvin sairaala-alue (hus.koje.fi)

2.4.1 Versio 1: Image-maps Canvakseen

Ensimmäisessä vaiheessa tarkoitukseni on luoda sama pohja uudelleen, mutta ilman image-mapsia käyttäen Canvasta pohjana. Projekti alkoi ensin tutustumalla Canvas elementin käyttämiseen littyvään JavaScrip, joka oli minulle täysin uusi asia. Can-

vaksella minun olisi tarkoitus piirtää projektin tarvittavat muodot kartalle. Jonkin aikaa Canvasta käytettyäni huomasin, että Canvakseen löytyy myös kirjastoja, jotka helpottavat sen käyttöä. Päädyin käyttämän Konvaa. Aluksi piirsin jokaisen muodon erikseen. Se ei ollut hyvä menetelmä, sillä siitä kertyi paljon koodia. Päädyin piirtämään kaikki muodot yhden funktion kautta siten, että vain sisältö tuodaan muusta tiedostosta.

Näin ensimmäinen versio oli valmis. (ks. kuva 2)



POLIKLINIKAT	RAKENNUS	OVI
Fysiatria	S	1
Hammas-, suu- ja leukasairaudet	N	1
Hematologia	A	1
Ihotaudit	M	1
Keuhkosairaudet	A	1
Kirurgia	O	8
Korva-, nenä- ja kurkkutaudit	B	1
Lastenpoliklinikka	E	10
Lastenkirurgia	E	10
Lastentautien erikoissairaanhoidon päivystys	E	11
Lastenneurologia	E	9
Lastenpsykiatria	L, P	7
Neurologia	M	1
Päivystys	K	5
Silmätaudit	M	1
Sisätaudit	N	3
Syöpätaudit	O	8
Yleissairaalapsykiatria	M	1
Äitiys ja naistentaudit	E	9

TUTKIMUSTOIMENPITEET JA YKSIKÖT	RAKENNUS	OVI
Apuvälineyksikkö	E	9
Dialyysi	N	3
Tähystystoiminta	N	3
Kliininen fysiologia ja isotooppilääketiede	A	1
Kliininen neurofysiologia	N	1
Kuuloasema	B	1
Kuvantaminen	K	4
Laboratorio	N	4
Leikkaussalit	K	
Leikkaushoitokeskus	K	4
Patologia	N	
Päivsäkirurgia	O	8
Silmätaudit	M	1
Sydänyksikkö	K	4
Synnytyssalit	E	9
Sädehoito	O	8

OPETUS	RAKENNUS	OVI
Auditorio	N	1, 39
Luentosali	O	8

OSASTOT	RAKENNUS	OVI
Keuhkosairaudet	A, B	1
Kirurgia 1 ja 2	N	3
Kirurgia 3	A, B	1
Korva-, nenä- ja kurkkutaudit	N	3
Lastenosasto	E	9
Lastenneurologia	E	9
Lastenpsykiatria	P	7
Neurologia	M	1
Päivystys	K	4
Sisätaudit 1 ja 2	M	1
Sisätaudit 4	A	1
Synnytys ja naistentaudit	E	9
Teho ja tehostettu valvonta	K	4
Vastasynt. teho ja tarkkailu	E	11
Yleissairaalapsykiatria	M	1

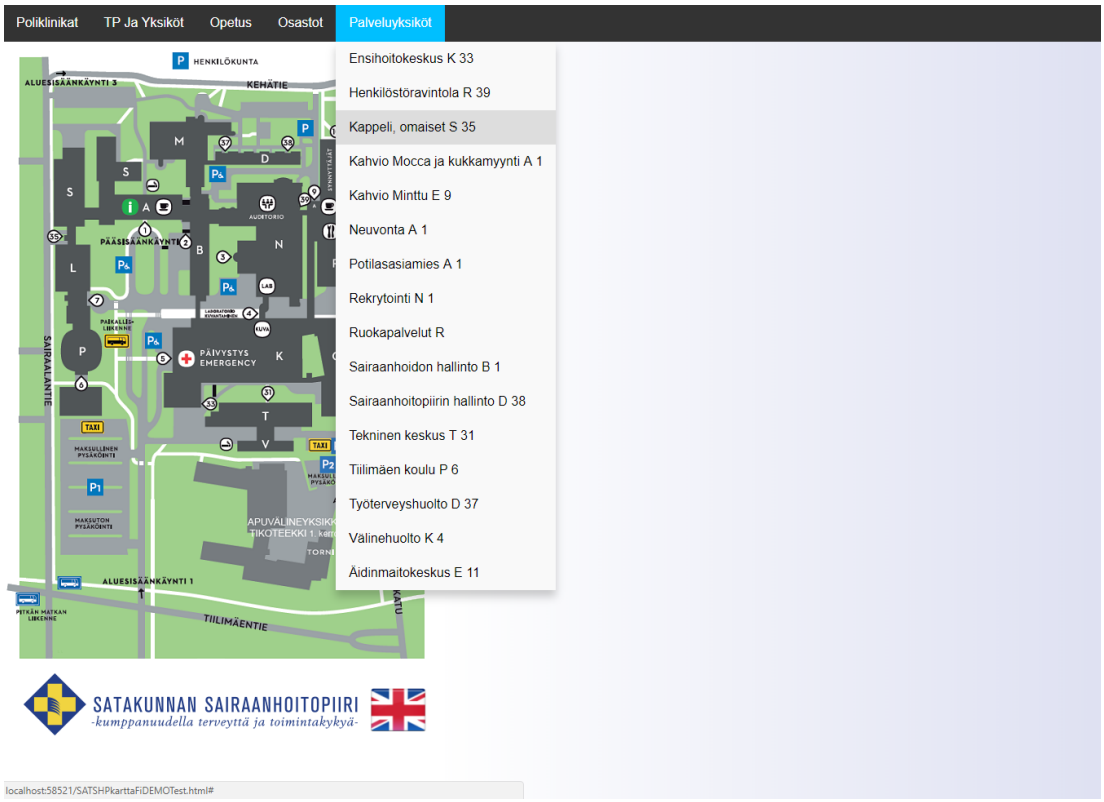
MUUT PALVELUYKSIKÖT	RAKENNUS	OVI
Ensihoitokeskus	K	33
Henkilöstöravintola	R	39
Kappeli, omaiset	S	35
Kahvio Mocca ja kukkamyynti	A	1
Kahvio Minttu	E	9
Neuvonta	A	1
Potilasasiamies	A	1
Rekrytointi	N	1
Ruokapalvelut	R	
Sairaanhoidon hallinto	B	1
Sairaanhoitopiirin hallinto	D	38
Tekninen keskus	T	31
Tiilimäen koulu	P	6
Työterveyshuolto	D	37
Välinehuolto	K	4
Äidinmaitokeskus	E	11

Kuva 2 Pekka Lukkarin tekemä pohja, jostaa vaihdoin image-maps:n canvakseen

2.4.2 Versio 2: Alasvetovalikko

Seuraavassa vaiheessa halusin yrittää saada ohjelman näkymään paremmin mobiililaitteiden ruudulla, sillä ensimmäisen version käytettävyys mobiililaitteilla oli heikko. Näkymää joutui vetämään paljon edestakaisin, eivätkä sen muutkaan toiminnallisuudet suosineet mobiililaitteita. Yritin parantaa tätä sijoittamalla yläpalkin, jonka alasvetovalikoista voisi valita tarvittavat yksiköt. Halusin myös lisätä paremmat skaalautuvuudet mobiililaitteille sekä myös mobiililaitteen sormipainallustoiminnot. Alasvetovalikot eivät toimineet halutulla tavalla ja ne peittivät kartan näkymän täysin.

Vaikka käytettävyys mobiililaitteilla oli nyt parempi, niin yleinen käytettävyys heikkeni. (ks. kuva 3)



Kuva 3 Alasvetovalikko versio

2.4.3 Versio 3: Sivupalkki

Tässä versiossa kokeilin, jos osittain läpinäkyvä sivupalkki olisi parempi ratkaisu. Suunnittelin sivupalkin avautumaan painamalla vasemmassa yläreunassa olevaa pientä SATSHP-logoa. Sivupalkista tulisi rakennuslinkkilista, joita napsauttaessa sivupalkki sulkeutuisi ja rakennuksen ja oven muoto piirtyisi kartalle. Käytettävyyskokeiluissa kuitenkin huomasin, että sivupalkin aukeaminen kartan päälle on häiritsevää. Yritin myös, että sivupalkki työntäisi karttaa sivulle, mutta tämä ei ole hyvä ratkaisu mobiililaitteille niiden pienen horisontaalisen näytön tilan vuoksi. (ks. kuva 4)

Kuva 4 Sivupalkki versio

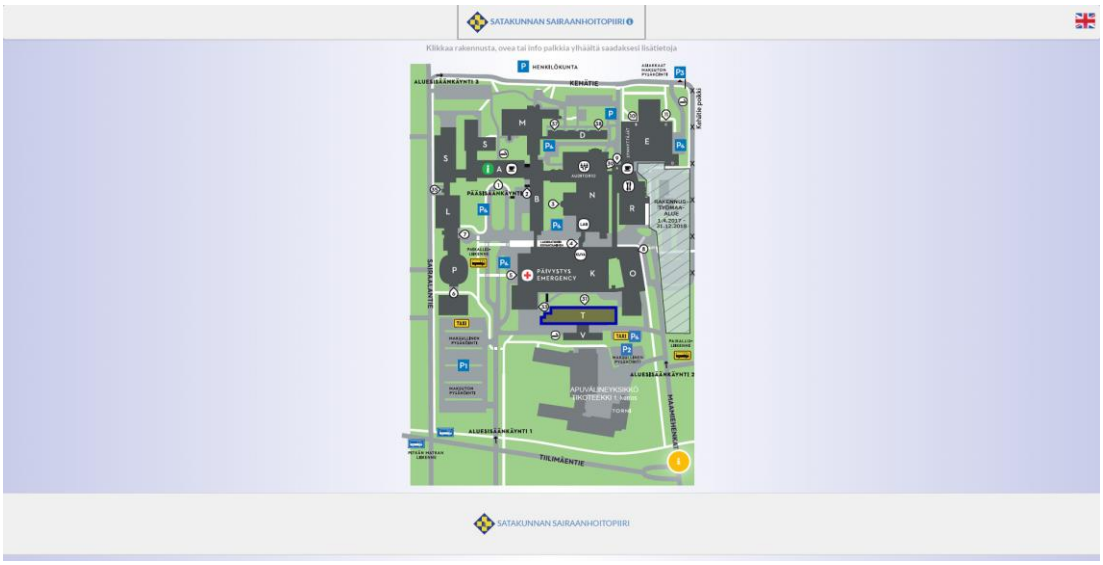
2.4.4 Versio 4: Haitarimalli

Asiakkaalta kun ei ollut tullut mitään erityisiä ulkonäkötoiveita sovellusta varten, laadin suunnitelman jonka hahmottelin paperille. Se käyttäisi version kaksi tavoin yläpalkkia, mutta sen toiminnallisuus olisi haitarimalli. Haitarimallia napsauttaessa avautuisi ensin yksiköt ja niitä napsauttaessa niiden kohteet. Laittaisin myös yläpalkkiin SATSHP logon, tekstin sekä kielenvaihtokuvake, joka oli tähän asti ollut ohjelman alareunassa parempaa näkyvyyttä ajatellen. Jouduin myös miettimään mahdollisten painettavien näppäinten ja kuvakkeiden kokoa ottaen huomioon mobiilikäyttäjät. Esimerkiksi kielenvaihtokuvakkeen oli aluksi aivan liian pieni. Huomion keräämiseksi kohtaan, josta haitarimallin voi avata laitoin pomppivan nuolikuvakkeen. Lisäksi kartan sekä yläpalkin väliin kirjoitin aputekstin, joka neuvoo käyttäjää napsauttamaan yläpalkkia tai jotakin rakennusta. Kartan ja muiden elementtien keskittäminen näkymän keskelle paransi ulkoasua isoilla näytöillä.

Väriteemana, kuten edellisissäkin versioissa, käytettäisiin SATSHP käyttämiä värejä. Pohjaväriksi useita värimalleja kokeiltuani päädyin hieman kellertävän valkoiseen, muun muassa musta pohjaväri sai sivut vaikuttamaan erittäin synkiltä.
Päädyin käyttämään myös alapalkkia eli footteria, johon sijoitin SATSHP logon.

2.4.5 Versio 4: Viimeistelyt

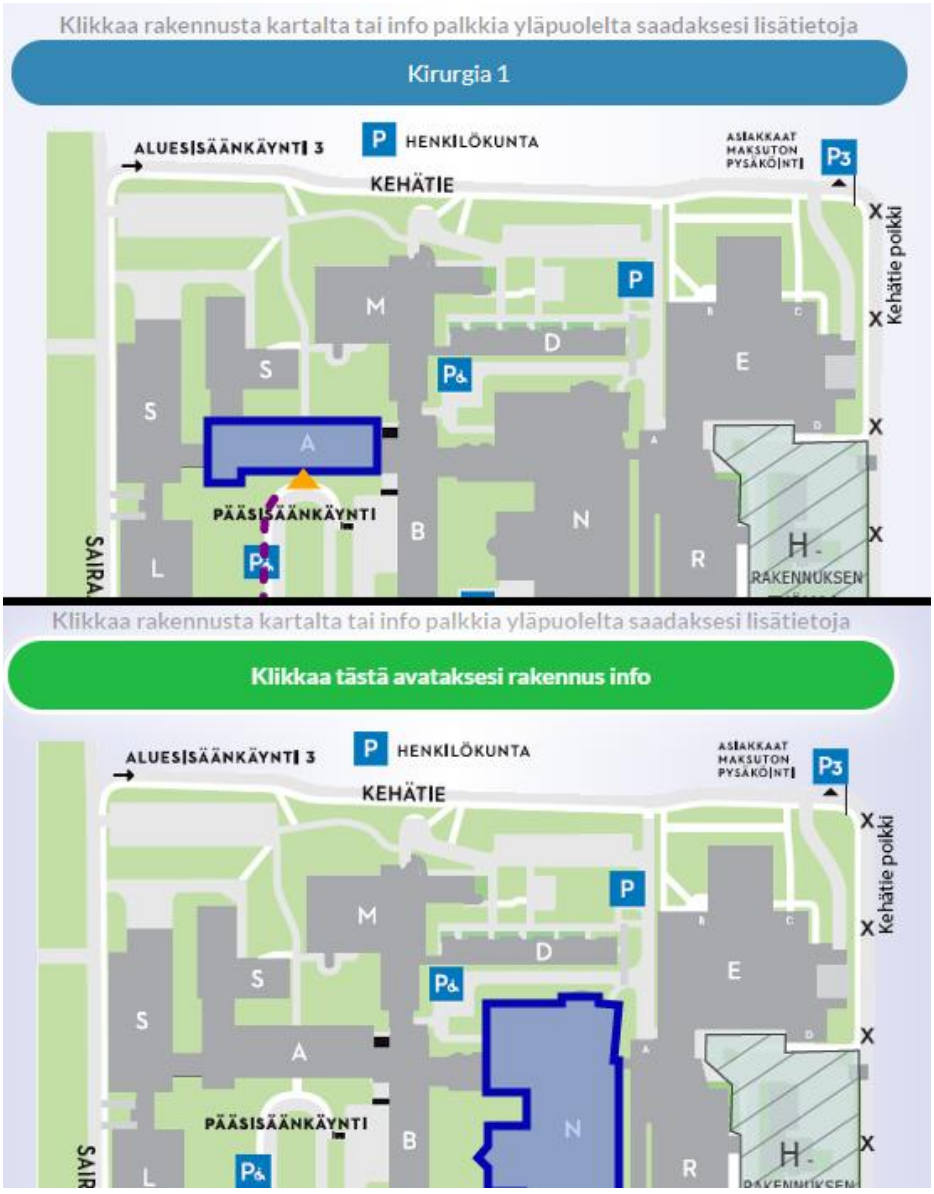
Asiakkaalta tulleiden muokkaus ehdotusten johdosta muutamaa osio otettiin muokattavaksi. Asiakas toivoi, että haitarivalikko olisi enemmän napin kaltainen. Muokkaamalla haitarivalikkoa näppäinmäiseksi lisäämällä siihen reunatyylittelyn sekä korvaamalla pomppivan nuolikuvakkeen info kuvakkeella saatiin valikosta helpommin huomattava.
Muina toivomuksina oli, että rakennusten info-näppäin olisi kirkkaampi ja isompi. Käyttämällä info-näppäimen ympärillä valkoista reunaa ja laittamalla hehkun CSS box-shadow tyylittelyn, sain aikaan halutun tehosteen. (ks. kuva 5)



Kuva 5 Lopullisen version pohja malli ennen viimeisimpiä tyylittelyjä

Asiakkaalta tulleen uuden kartan mukana tuli uusia muutoksia sekä toiveita. Asiakas toivoi reititystä parkkipaikoilta oville, joiden kautta mennään valittuun kohteeseen. Reitityksen toteutan lisäämällä purppuran katkoviivan parkkipaikan ja oven välille (ks. kuva 6). Valittaessa reitti välkkyy hitaasti korostaen sen väriä pinkiksi, värimaa-

ilmaa tehdessä huomioin hyviä käyttöliittymän suunnittelun tapoja. Kartan yksinkertaistetun ulkoasun vuoksi ovien sijainnit poistettiin kartasta. Ovien poiston myötä joudun poistamaan mahdollisuuden napsauttaa ovia kartalla ja saada niistä tietoa. Ovet ovat vielä mukana, mutta näkyvät vain, kun valitaan haluttu kohde. Vaihdoin ovimuotojen ulkoasun nuoliksi, osoittamaan suuntaa jossa sisäänkäynti on. Myös keltainen info-näppäin muuttui vihreäksi ja koko kartan levyiseksi näppäimeksi kartan yläpuolelle sisältäen ohje tekstin. Karttaan tuli myös valitun kohteen nimi näkyviin kartan yläpuolelle jotta käyttäjä näkee heti minkä sijainnin hän on valinnut. Myös haitarivalikon listat muutettiin taulukko maisiksi. Rakennusten täyte värikin muuttui siniseksi.



Kuva 6 Kartan reititys sekä viimeisimmät tyyli muutokset

2.5 Sovellus

Sovelluksesta oli alusta asti Medbitin puolelta tarkoitus kehittää yleiskäytännöllinen, jotta sitä tai sen pohjaa voitaisiin myös käyttää tulevissa projekteissa. Ohjelma tulisi siis yrittää suunnitella siten, että sen jokainen komponentti voidaan erottaa toisistaan ja käyttää erikseen. Esimerkiksi, kun pohjaa käytettäisiin luomaan uusi kartta TYK-Sille, voitaisiin siitä poistaa SATSHP:n version käyttämät HTML, CSS tai muut tarpeettomat komponentit ja ohjelma toimisi silti. Tulisi myös päästä eroon pohjaversion HTML riippuvuudesta, sillä pohja versiossa image-maps tuli HTML puolelta. Jos image-maps - koodista ei olisi päästy eroon, ohjelman monikäyttöisyys olisi kärsinyt, sillä kaikki muokkaukset tarvitsisi aina tehdä HTML tiedostoon, joka jo muutenkin vaatii muita osia toimiakseen.

Kaksikielisyys sisältäen kielet englanti ja suomi on yksi suunnittelun tärkeistä osista. Kaikki tehdään siten, että niistä löytyy myös englannin kielinen versio. Tämä tarkoittaa, että esim. JSON-datatiedosto löytyy aina kaksin kappalein.
Rakennusten, ovien ja muiden tiedot tulevat kaikki JSON tiedosto tyypistä. Näin ollen tiedot ja osat jotka käyttävät tietoja ovat riippumattomia toisistaan ja tarvittaessa voidaan muokata vain JSON-muotoisina tiedostoina, kun halutaan päivittää dataa.
Toivomuksiin kuului myös, että sovellus toimisi kahteen suuntaan niin tekstivalikoista karttaan kuin kartasta tekstivalikkoihin. Valikkojen ja kartan tulisi myös toimia hyvin mobiilikäytössä.
Ohjelman koodi ei myöskään tulisi sisältää turhaa tai huonosti kirjoitettua koodia. Versioita toimitetaan asiakkaan testattavaksi prototyyppipohjalla. Aina kun uusi versio saadaan valmiiksi, se toimitetaan asiakkaan testattavaksi testiosoitteen kautta. Tätä ennen versiota testataan useamman kerran mahdollisten virheiden löytämiseksi.

3 TOTEUTUS

3.1 Valmis sovellus

Canvasta ja sen kirjastoa Konvaa tarvitaan työssä luomaan pohja kartan napsautettaville rakennuksille ja oville. Canvas luo tavallaan piirtämisalustan määritetylle alueelle tässä tapauksessa kartanalue, johon voidaan piirtää muotoja tai muita kaksiulotteisia elementtejä. Alueen ulkopuolelle canvaksella piirretyt elementit eivät näy. Rakennusten ja ovien koordinaatit ja muut tiedot tulivat aluksi tiedostosta pushShapes.js joka sisältää koordinaattien lisäksi myös kaiken muun tarvittavan tiedon sen taulukoista. Myöhemmin tiedot siirrettiin tulemaan JSON tiedostosta parempaa ylläpitoa ajatellen. Koordinaatteja käytetään ovien, reittien ja rakennusten sijaintien määrittämiseen. Ovissa on vain yksi x- ja y- koordinaatti, jonka avulla piirretään nuoli suuntaamaan kartan ovelle, tässä tapauksessa väriteemana käytetään SATSHP värejä. Rakennukset koostuvat useista x- ja y- koordinaatti pisteistä, joita hyödyntäen Konva.js piirtää linjan, joka muodostaa rakennuksen rajat. Kun rakennuksen rajat on piirretty, täytetään se värillä, jotta se erottuisi enemmän karttapohjakuvasta. Reitit piirretään samalla menetelmällä kuten rakennuksetkin (ks. kuvat 7 ja 8). Konva.js saa polun piste koordinaatteja joiden mukaan piirretään reitti haluttuun kohteeseen. Muu JSON data sisältää piirrettävien muotojen nimet, tekstikentän tiedot, kuvan sekä piirtämiseen käytetyt muuttujat.

```
function CreateBuildingsandItsMouseEvents() {

    for (var i = 0; i < buildings.length; i++) {
        var s = buildings[i];
        var links = document.getElementsByClassName(s.link);
        var poly = new Konva.Line({
            points: s.points,
            fill: 'rgba(100,149,237,0.4)',
            stroke: '#0404B4',
            strokeWidth: 5,
            baseStrokeWidth: 5,
            closed: true,
            name: s.name,
            desc: s.desc,
            descDep: s.descDep,
            img: s.img,
            id: s.link,
            opacity: 0
        });

        poly.on('mousedown touchstart', function () {

            document.getElementById("showSelectedDepartment").style.display = "none";

            var currentCircle = this;

            cardTitleNameText = currentCircle.attrs.name;
            cardDescriptionText = currentCircle.attrs.desc;
            cardDepartmentsText = currentCircle.attrs.descDep;
            cardImageSetter = currentCircle.attrs.img;

            document.getElementById("testimg").src = cardImageSetter;
            document.getElementById("cardTitle").innerHTML = cardTitleNameText;
            document.getElementById("cardDesc").innerHTML = cardDescriptionText;
            document.getElementById("cardDep").innerHTML = cardDepartmentsText;

            $('.selectedfield').removeClass('selectedfield');

            var polyfind = stage.find('Line');
            for (var i = 0; i < polyfind.length; i++) {
                polyfind[i].opacity(0);
            }

            var polyfind = stage.find('Arrow');
            for (var i = 0; i < polyfind.length; i++) {
                polyfind[i].opacity(0);
            }

            this.opacity(1);
            InfoButtonToggle();
            MoveFromBuildingToInfo();
            layer.draw();
            $('.' + this.id()).addClass("selectedfield");
        });

        poly.on('mouseenter', function () {
            stage.container().style.cursor = 'pointer';
        });

        poly.on('mouseleave', function () {
            stage.container().style.cursor = 'default';
        });

        layer.add(poly);
    }
}
```

Kuva 7 Rakennusten luonti

```
function CreateDoorPointer() {
    for (var i = 0; i < doors.length; i++) {
        var dor = doors[i];
        var circle = new Konva.Arrow({
            x: dor.x,
            y: dor.y,
            points: dor.points,
            pointerLength: 6,
            pointerWidth: 10,
            stroke: 'orange',
            fill: 'orange',
            strokeWidth: 5,
            name: dor.name,
            desc: dor.desc,
            img: dor.img,
            id: dor.link,
            opacity: 0
        });
        layer.add(circle);
    }
}

function CreatePaths() {
    for (var i = 0; i < paths.length; i++) {
        var pa = paths[i];
        var route = new Konva.Line({
            points: pa.points,
            stroke: 'Purple',
            strokeWidth: 5,
            lineJoin: 'round',
            dash: [5, 10],
            lineCap: 'round',
            id: pa.link,
            name: "Path",
            opacity: 0
        });
        layer.add(route);
    }
}
```

Kuva 8 Reititysten ja ovi paikkojen luonti

Suurimmasta osasta käytettävistä JSON tiedostoista löytyy myös englanninkielinen toinen versio. Kieli data ladataan ohjelmaan valittaessa kieli, tämä vaihtaa pohja HTML tiedostoa, joka taas lataa eri kielen versiot datasta. Kielen vaihtokuvake sijaitsee sovelluksen oikeassa yläreunassa.

Muodot piirretään silmukan kautta piirtäen niin monta muotoa, kuin niitä löytyy JSON tiedoston taulukosta. Samalla jokaista muotoa kohden määritetään sille hiirenpainallustoiminnot silmukan sisällä. Toiminnot vaikuttavat, miten muoto toimii, kun sitä napsautetaan hiirellä. Rakennuksen napsauttaminen näyttää info-näppäimen kartan yläpuolelle (ks. kuva 9). Info-näppäintä napsauttamalla tulee näkymään tietokenttäkortti kartan alapuolelle, jossa on kuva liittyen sijaintiin ja nimi, tekstikenttä sekä mitä kohteita (osasto, poliklinikka, jne) se sisältää.

Kuva 9 Piirretty rakennus

Sovelluksen yläreunassa käytetään Semantic UI haitarimallista alasvetovalikkoa, jonka muokkasin toimimaan navikointipalkin tavoin yläpalkkina. Haitarien avautuessa ne eivät tule kartan kuvan päälle, vaan työntävät niitä alemmaksi. Itse päähaitari sisältää useita alihaitareita jotka avautuessaan näyttävät yksikön sisältämät kohteet. Yksiköt ovat poliklinikat, tutkimustoimenpiteet ja yksiköt, opetus, osastot sekä muut palvelu yksiköt. Semantic Ui käytetään myös monissa muissa elementeissä sovelluksessa, kuten kuvien pienentämiseen, kuvakkeisiin, tietokentän kortti pohjaan sekä joihinkin efekteihin.

Toimipaikat kunkin yksikön sisällä toimivat linkkeinä kartan rakennuksiin, oviin ja polkuihin (ks. kuva 10). Kohdetekstit HTML puolelle tulevat JavaScript-funktiosta. Jokainen toimipaikka on linkki, sen kohteen määrittää elementille annettu luokka. Jos kohde tekstit eivät tulisi JavaScript puolelta JSON tiedoston kautta, tämä aiheuttaisi riippuvuuden HTML puoleen luokkalinkitysten vuoksi. Kun napsauttaa esim. Muut palveluyksiköt ja sen kohdetta sairaanhoitopiirin hallinto, niin sairaanhoitopiirin hallinto tekstin taustaväri korostuu ja näkymä siirtyy karttaan käyttäen JavaScript funktiota värittäen rakennuksen, reitin ja oven. Lisäsin myös animaation, joka vilkuttaa rakennusta, reittiä ja ovea hetken näkymän siirtymän jälkeen antamaan sille lisää huomiota. Animaatio toimii käyttämällä JavaScriptin timeout ominaisuutta, joka

mahdollistaa toimintojen ajastamisen. Navikointipalkin keskelle on määritetty kenttä, jossa on SATSHP logo sekä teksti. Lisäksi animoin CSS tiedoston kautta info-kuvakkeen pomppimaan SATSHP tekstin jälkeen, jotta käyttäjä ymmärtäisi kyseessä olevan painettava näppäin. (ks. kuva 11 haitarivalikon luonti)

OSASTOT

TOIMIPAIKAT	RAKENNUKSET	OVET
Keuhkosairaudet	A B	1
Kirurgia 1	A	1
Kirurgia 2	N	3
Kirurgia 3	A B	1
Korva-, nenä- ja kurkkutaudit	B	1
Lastenosasto	E	9
Lastenneurologia	E	9
Lastenpsykiatria	P	7
Neurologia	M	1
Päivystysosasto	N	3

Kuva 10 Haitarivalikon sisällä oleva taulukko sisältäen toimipaikat

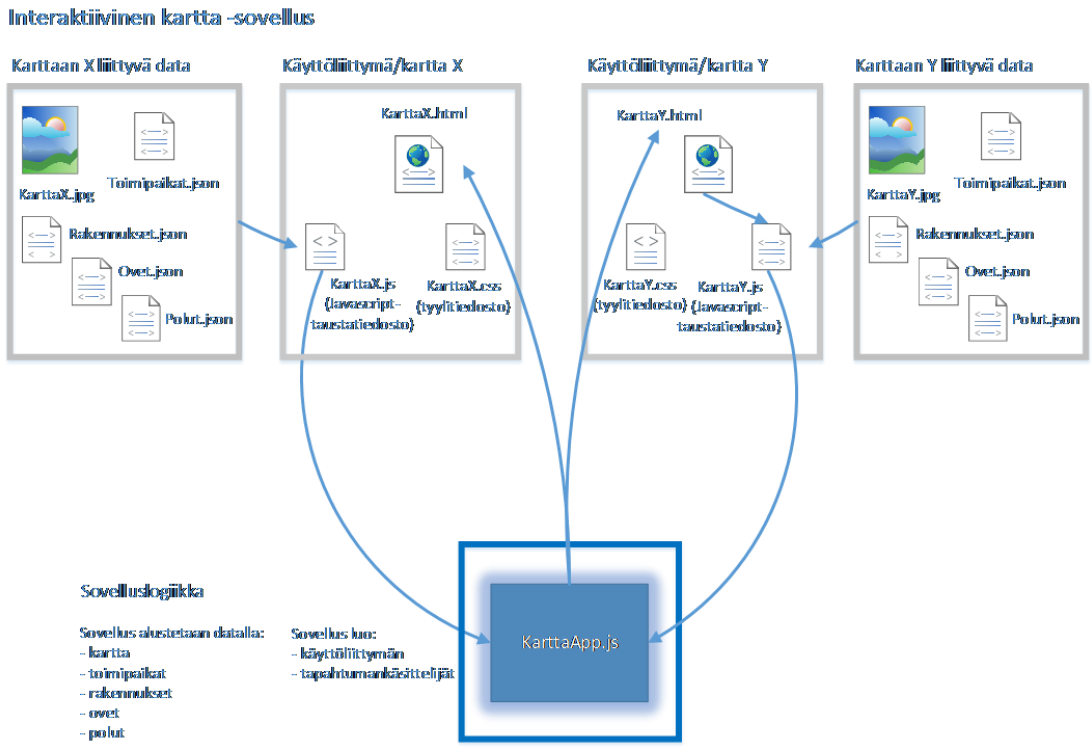
```
function CreateAccordionTitles($container) {
    if (!$container.length)
        return;

    Object.keys(toimipistedata).forEach(function (key) {

        var component =
            '<div class="title styleSetSubAccordion">' +
            '<i class="dropdown icon"></i>' +
            toimipistedata[key][0].title +
            '</div>' +
            '<div class="content styleSetSubAccordion">' +
            '<div class="ui two column divided grid myPadding ' + toimipistedata[key][0].classname + '\">' +
            '<table class="ui compact unstackable celled table">' +
            '<thead>' +
            '<tr><th>' + _tekstit.tblHeaderToimipaikat + '</th>' +
            '<th>' + _tekstit.tblHeaderRakennukset + '</th>' +
            '<th>' + _tekstit.tblHeaderOvet + '</th>' +
            '<th>' + _tekstit.tblHeaderKerrokset + '</th>' +
            '</tr></thead>' +
            '<tbody>';

        for (var i = 0; i < toimipistedata[key].length; i++) {

            console.log("Floors: " + toimipistedata[key][i].floors);

            var txtp = '<tr class="' +
                toimipistedata[key][i].buildings + " " +
                toimipistedata[key][i].door + " " +
                toimipistedata[key][i].path +
                '\">' +
                '<td class="imagemaps-linkki part1 column ' +
                toimipistedata[key][i].buildings + " " +
                toimipistedata[key][i].door + " " +
                toimipistedata[key][i].path +
                '\" href="#">' + toimipistedata[key][i].location + '</td>' +
                '<td class="imagemaps-linkki part2 column ' +
                toimipistedata[key][i].buildings + " " +
                toimipistedata[key][i].door + " " +
                toimipistedata[key][i].path +
                '\" href="#">' + toimipistedata[key][i].buildings + '</td>' +
                '<td class="imagemaps-linkki part3 column ' +
                toimipistedata[key][i].buildings + " " +
                toimipistedata[key][i].door + " " +
                toimipistedata[key][i].path +
                '\" href="#">' + toimipistedata[key][i].door + '</td>' +
                '<td class="imagemaps-linkki part4 column ' +
                toimipistedata[key][i].buildings + " " +
                toimipistedata[key][i].door + " " +
                toimipistedata[key][i].path +
                '\" href="#">' + toimipistedata[key][i].floors + '</td>' +
                '</tr>';
            component = component + txtp;
        }

        component = component +
            '</tbody >' +
            '</table >' +
            '</div>' +
            '</div>';

        $container.append(component);
    });
}
```

Kuva 11 Haitarivalikon luonti

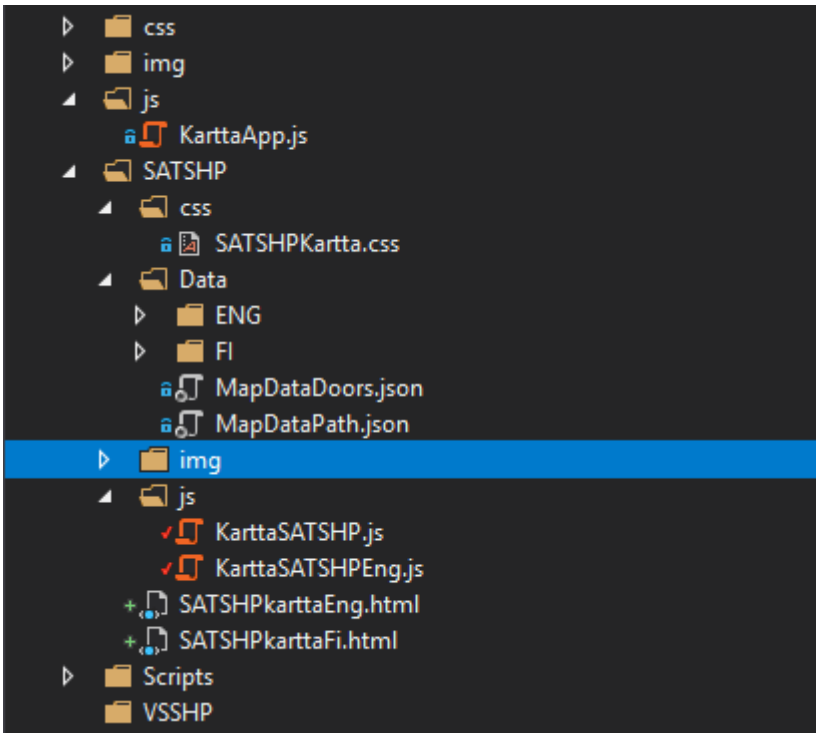
Sovelluksen alapalkki, jonka keskellä on SATSHP teksti ja logo toimii linkkinä SATSHP:n sivuille. Alapalkin sisälle avautuu tietokenttäkortti info-näppäintä painaessa. Kortti sisältää tiedot kohteen nimestä, tekstikentästä sekä kuvan (ks. kuva 12). Kortti voidaan myös sulkea kortin alareunassa olevasta sulje napista, joka kuin info painike luodaan JavaScriptin puolella välttäen riippuvuuksia. Kortti sulkeutuu myös, kun painetaan haitarivalikon toimipaikkaa. Kortin tiedot vaihtuvat, jos se on auki ja eri rakennusta napsautetaan.

Kuva 12 Rakennuksen info kenttä

Kuvista 13 ja 14 käy selväksi sovelluksen rakenne. HTML tiedosto kutsuu JavaScript ja CSS tiedostoja. CSS tiedosto muokkaa HTML ulkoasua ja JavaScript kutsuu funktioita sekä lataa JSON ja kuvatiedostot. Yhdessä nämä muodostavat sovelluksen jolla on hyvä uudelleenkäytettävyys tulevissa interaktiivisissa karttasovelluksissa.



Kuva 13 Sovellusrakenne



Kuva 14 Visual Studion sovellusrakenne. KarttaApp on pääsovellus

3.2 Haasteet

Haasteissa kerron ja kuvailen tilanteita, jotka työn edetessä aiheuttivat ongelmaa ja joiden käyttöä jouduin opiskelemaan. Haasteet eivät ole huono asia, vaan ne pakottivat oppimaan jotain uutta.

Canvas

Aloitettuani tutustumisen Canvas JavaScript kirjastoon tuli käytännön ongelmia perus Canvaksen käytössä. Halusin merkitä ovet karttaan ympyröin, mutta Canvas itsessään ei omaa ympyrän hiirenkosketustoimintoa koordinaattien perusteella. Perus Canvas tunnisti hiiren siirtymisen muodon päälle, vain jos se oli piirretty käyttäen lineTo komentoa. Yritin ensin erilaisten muotojen käyttöä ovien merkitsemisessä, mikään näistä ratkaisuista ei tuntunut hyvältä. Canvaksen peruskäyttö oli hieman raskasta, joten päätin ottaa käyttöön Canvas kirjaston Konva.js.

Karttakuva

Karttakuvan sovittamisessa oli myös omat ongelmansa. Alussa määrittelin kaikki koordinaattipisteet rakennuksille ja oville perustuen meillä olevaan karttapohjaan. SATSHP päivitti karttoja rakennustyömaan sekä ulkoasumuutosten vuoksi ja meille lähetettiin uudet kartat. Huonoa tässä oli uuden kartan täysin eri suhde vanhaan nähden. Lisäksi karttojen reunojen paksuudet saattoivat olla eri jopa muuttaessa samaan kokoon suhteeseen. Jouduin muokkaamaan karttaa siten, että poistin laskelmieni mukaan oikean määrän reunustaa. Tämän jälkeen muutin kartan koon vanhaa vastaavaksi. Jos tämä ei olisi onnistunut, olisin joutunut laskemaan kaikki rakennusten, reittien ja ovien koordinaatit uudelleen. Uusien karttamuutosten edessä ylläpidollisesti tulee rasitteita laittaa karttakoordinaatit uudelleen datatiedostoihin.

JavaScript ja jquery

JavaScriptin käyttö ei myöskään ollut aina täysin ongelmatonta. Usein vaikeuksia syntyi, kun yrittää kohdistaa jotakin elementtiä, funktiota tai jotain muuta jonkin toisen sisältä. Tässä jouduin kysymään apua Stack Overflow-ohjelmointifoorumeilta. Ks. kuva 15 yrityksistä ratkaista ongelma.

Väärät Ratkaisut:

```
var item = this.name;
item.style.backgroundColor = "#ffcc00";

document.getElementsByClassName(this.name).style.backgroundColor = "#ffcc00";

highlight_target = "#ffcc00";

document.getElementsByClassName(${this.name }).style.backgroundColor = "#ffcc00";

document.getElementsByClassName(s.name).style.backgroundColor = "#ffcc00";
```

Oikea Ratkaisu:

```
$('.' +this.name()).css({ backgroundColor: "#ffcc00"});
```

Kuva 15 Vääriä ratkaisuja ja oikea ratkaisu

Semantic

Semantic aiheutti päänvaivaa taas tyylien vuoksi. Otin Semantic UI käyttöön kokonaisuudessaan, mutta en osannut ajatellakaan, että voisin valita sen myös osina GitHubin puolelta. Koko Semantic paketti vaikutti sovelluksen kaikkiin tyyleihin. Jotta sain omat tyylitykseni yliajamaan Semantic:n omat tyylit, jouduin käyttämään tyyleissäni !important komentoa perässä. Myöhemmin jouduin ottamaan vain ne osat Semantic UI:sta, jotka tarvitsin sen sijaan, että olisin ladannut koko paketin.

Debuggerit

Debuggerien eli virheiden löytämiseen ja korjaamiseen tarkoitettujen toimintojen käyttö oli aluksi haastavaa. Vaikka aikaisemmissa opinnoissani olinkin käyttänyt debuggereita, oli niiden käyttö silti jäänyt vähäiseksi tai puutteelliseksi. Työssä joka painottui Web-sovelluksen luomiseen, opettelin käyttämään Chromen sisäistä debuggeria. Hyvä ominaisuus Chromen sisäisessä debuggerissa oli muun muassa CSS tyylien nopea muuttaminen vain napsauttamalla niitä päälle tai pois.

Versionhallinta

Versiointi on sovelluksen uusien versioiden päivitysten lataamista johonkin varastoon, josta jokainen projektiin osallistuva henkilö voi niitä käyttää. Tarvittaessa varastoon voidaan myös ladata sovelluksen viimeisin toiminut versio, jos nykyinen versio on mennyt rikki. Vaikka tämä ei itsessään ollut mitenkään vaikea sisäistää, oli se silti aivan uusi ja välttämätön asia osata liittyen sovelluskehitykseen.

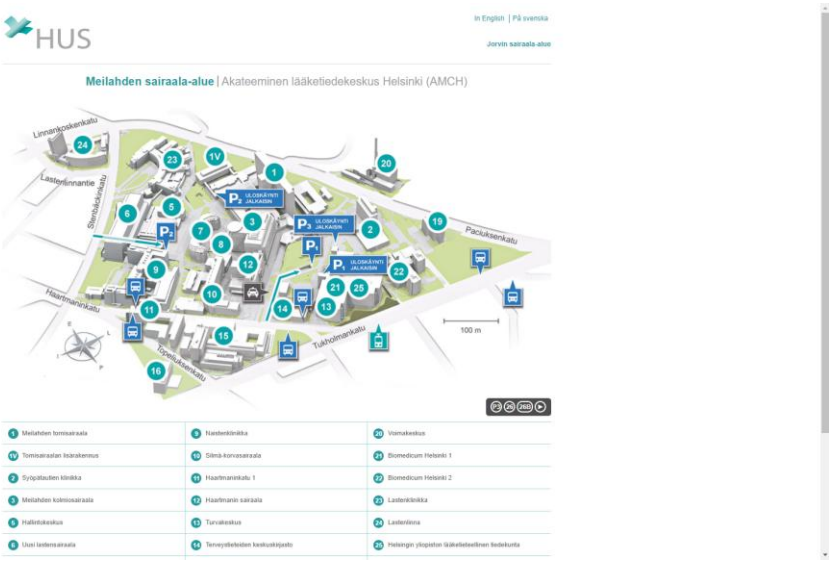
Selaimet

Sovelluksen tuli toimia eri selaimilla, mutta en osannut odottaa, miten eri tavoin sovellus reagoi kullakin selaimella. Joillain selaimilla jotkin tyylittelyt eivät toimineet, toisella taas jokin tiedosto puuttuu tai jokin antoi virhettä. Joissakin selaimissa jotkin JavaScript tai CSS toiminnot eivät olleet tuettuina.

4 VERTAAMINEN HUSIN INTERAKTIIVISEEN KARTTAAN

4.1 Johdanto

HUSin eli Helsingin ja Uudenmaan sairaanhoitopiirin interaktiivinen kartta toimi mallina (ks. kuva 16) siitä, minkä kaltaista karttasovellusta tekisin SATSHP:n ja TYKSin käyttöön. Tässä osiossa vertailen Helsingin ja Uudenmaan sairaanhoitopiirin karttasovellusta omaan karttasovellukseeni SATHSHP:lle. Tarkoituksena on antaa kuvaa, millaisen mallin pohjalta lähdin rakentamaan omaa ratkaisuani ja katsoa miten onnistuin rakentamaan oman sovellukseni verrattuna HUSin karttasovellukseen.



Kuva 16 HUS Meilahden sairaala-alue (hus.koje.fi)

4.2 Vertailu

HUSin kartta, kuten suunnittelemani karttasovellus sisältää samantyylisen ulkoasun. Ala- ja yläpalkin väliin on sijoitettu karttakuva, oikeassa yläreunassa sijaitsee kielen vaihto ja koko sovellus on keskitetty näkymän keskelle. Myös info-palkki aukeaa alapalkin sisään. Reitityksen valittuun kohteeseen löytyvät myös kummastakin sovelluksesta. Sivun värit myös vastaavat HUS käyttämiä värejä kuten SATSHP:lle suunnittelemassani karttasovelluksessa. Toisin kuin tekemässäni kartta sovelluksessa HUSin interaktiivisessa kartassa ei voi painaa kartasta kohteita siten, että ne antaisivat mitään lisätietoa. HUSin kartta toimii myös käyttäen useita kuvia HTML puolella canvaksen, imagemapsin tai jonkin muun piirtämisen sijaan. Kuvat ovat siis erillisiä valmiiksi piirrettyjä kuvia, jotka vaihtuvat kohdetta napsauttaessa. Itse HUSin karttakuva on hyvin suunniteltu kolmiulotteinen karttakuva, jonka suhteet tukeva paremmin laaja kuva näkymää. HUSin kartan kohteiden nimet on sijoitettu alas riveihin. Omassa sovelluksessani kohteita on enemmän, joten niiden sijoittaminen alas tai ylös riveihin ei toiminut käytettävyydeltään. Kohteita napsauttamalla HUSin kartassa saa myös kävely ohjeet kuhunkin kohteeseen, sekä halutessaan julkisenliikenteen reittiohjeet info palkin kuvakkeita napsauttamalla. Joistakin HUSin kartan kohteen info-palkkiin avautuvista kuvista on mahdollista katsoa panoraamakuva siltä alueelta. HUSin kartta on jaettu kolmeen osaan Meilahden sairaala-alue, sen sivualue sekä Jorvin sairaala-alue. Jorvin sairaala-alue toisin kuin Meilahden sairaala-alueen kolmiulotteinen kartta on kaksiulotteinen perinteisempi kaupunki kartta.

4.2.1 Mahdolliset parannukset HUSin karttaan

HUSin kartan käyttöohjeet on sijoitettu näkymän alapuolelle, jolloin käyttäjä ei välttämät näe niitä. Omasta kokemuksestani, koska en nähnyt HUSin käyttöohjeita yritin heti painaa kohteita kartasta ilman, että mitään tapahtui. HUSin kartan sivuilta ei huomaa ensinäkemältä kovin helposti, että sivulla on myös kaksi muuta karttaosiota.

Jorvin sairaalan-alue sijaitsee ylhäällä kirjoitettuna pienellä fontti koolla ilman mitään erikoistyylittelyä, joten sen huomaaminen vaati itseltäni kymmeniä sivuvierailukertoja. Jorvin sairaalan kartan ulkoasuun ei myöskään ole panostettu yhtä paljon kuin Meilahden sairaalan kartan ulkoasuun.

4.2.2 Mahdolliset parannukset omaan karttasovellukseen

Kartan skaalautuvuus aiheutti ongelmia, sillä jos skaalaisin karttaa tarvittaessa mobiililaitteille olisi rakennuksia vaikeampi painaa sormella. Tämän vuoksi kartta ei skaalaa mobiililaitteilla, mistä johtuu, ettei kartta mahdu mobiililaitteiden näytöille kokonaisuudessaan. Myös mobiililaitteilla karttaa liikuteltaessa henkilö voi vahingossa painaa rakennusta sormella, jolloin tiedot kentässä ja kartalla vaihtuvat. Jos kartasta otettaisiin pois kartan rakennusten napsauttaminen, voisi karttaa skaalata ja päästäisiin eroon ongelmista. Myös isojen näyttöjen kuva suhteen hyödyntäminen kenties avaamalla tietokenttä kartan viereen perustuen mikä kuvakoko on käytössä, olisi hyvä ratkaisu tyhjän horisontaalisen tilan täyttämiselle.

5 LOPUKSI

Sovellustyön aloittaminen oli helppoa, omasin jo jonkin verran kokemusta JavaScriptin, CSS:n, ja HTML:n käytöstä. Haastavimmat ja aikaa kuluttavimmat osat työtä tehdessä olivat ne pienet parannukset ja yhteensopivuussäädöt sekä uusien asioiden opiskelu. Sovellus onnistui täyttämään ne odotukset, joita asiakas toivoi. Lisäyksiä ja muutoksia tehtiin asiakkaan tahtiin samalla, kun yritin pitää sovelluksen uudelleen käytettävänä.

Sovelluksen lopputulos oli mielestäni hyvä ja se saavutti sille aluksi asetetut määritykset niin asiakkaalta kuin Medbitin puolesta. Sovellus parantaa selvästi käytettävyyttä verrattuna perinteiseen pdf-muotoiseen karttaan varsinkin mobiililaitteilla. Mobiilipuolen osion olisin voinut toteuttaa vielä paremmin liittyen eri kuvasuhteisiin ja kartan skaalautuvuuksiin.

Itse projektin aikana taitoni työskennellä yrityksessä ohjelmoijana paranivat huomattavasti. Ymmärrykseni miten ohjelmointiyrityksessä toimitaan tai käsitellään sovelluksia, sen kehityksen eri vaiheiden aikana parani huomattavasti pelkän koulussa opitun teorian lisäksi. Myös taitoni liittyen useisiin eri ohjelmointikieliin sekä Websovellusten suunnitteluun kasvoi. Jouduin joka päivä käyttämään eri kieliä ja yhdistämään niitä keskenään saadakseni työtä etenemään. Ongelman ratkontaa ja uuden oppimista tuli vastaan lähes toistuvasti, kun hain ja opiskelin tietoja milloin mistäkin sovelluskehitykseen kuuluvasta alueesta tai kielestä.

LÄHTEET

HUS

HUS. Viitattu 21.2.2018

http://hus.koje.fi

Konva, 5.2.2018

Konva. Viitattu 21.2.2018

https://konvajs.github.io/api/index.html

MEDBIT

Terveitä ICT-ratkaisuja, joissa on sydän mukana. Viitattu 21.2.2018

http://www.medbit.fi/keita-me-olemme/

Semantic, 20.2.2018

Semantic UI. Viitattu 21.2.2018

https://github.com/Semantic-Org/Semantic-UI

usability.gov, 21.5.2014

User Interface Design Basics. Viitattu 21.2.2018

https://www.usability.gov/what-and-why/user-interface-design.html

Wikimedia Foundation, Inc., 19.2.2018

Microsoft Visual Studio. Viitattu 21.2.2018

https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

Wikimedia Foundation, Inc., 13.2.2018

Canvas element. Viitattu 21.2.2018

https://en.wikipedia.org/wiki/Canvas_element

Wikimedia Foundation, Inc., 9.5.2017

JavaScript. Viitattu 21.2.2018

https://fi.wikipedia.org/wiki/JavaScript

Wikimedia Foundation, Inc., 25.11.2018

JSON, Viitattu 21.2.2018

https://en.wikipedia.org/wiki/JSON

Wikimedia Foundation, Inc., 3.2.2018

Viitattu 21.2.2018

https://en.wikipedia.org/wiki/HTML

Wikimedia Foundation, Inc., 20.2.2018

Cascading Style Sheets. Viitattu 21.2.2018

https://en.wikipedia.org/wiki/Cascading_Style_Sheets

Wikimedia Foundation, Inc., 2.10.2015

Ohjelmiston vaatimusmäärittely. Viitattu 21.2.2018

https://fi.wikipedia.org/wiki/Ohjelmiston_vaatimusm%C3%A4%C3%A4rittely