KARELIA-AMMATTIKORKEAKOULU
Media-ala

Antton Hyvärinen

UXPIN SUUNNITTELUOHJELMISTO TEOLLISEN IOT:N VERKKOSOVELLUSTEN KEHITYKSESSÄ

Opinnäytetyö
Toukokuu 2019

Karelia AMMATTIKORKEAKOULU	**OPINNÄYTETYÖ** **Toukokuu 2019** **Media-ala** Tikkarinne 9 80200 JOENSUU +358 13 260 600 (vaihde)

Tekijä
Antton Hyvärinen

Nimeke
UXPin suunnitteluohjelmisto teollisen IoT:n verkkosovellusten kehityksessä

Toimeksiantaja
Process Genius Oy

Tiivistelmä

Tässä opinnäytetyössä esitellään UXPin suunnitteluohjelmisto teollisen IoT:n verkkosovellusten kehityksessä. UXPinia käytetään käyttöliittymäsuunnitelmien ja prototyyppien tuottamiseen. Työn tavoitteena oli käsitellä UXPinin perusominaisuuksia käyttökokemussuunnittelun kontekstissa. Opinnäytetyön tietoperusta käsittelee käyttäjälähtöisen käyttöliittymäsuunnittelun periaatteita ja käytettävyyteen ja käyttäjäkokemukseen vaikuttavia tekijöitä.

Opinnäytetyön toiminnallisessa osiossa on dokumentoitu UXPin prototyypin vuorovaikutteisuuden luomiseen liittyviä ominaisuuksia ja rajoitteita. Dokumentaation sisältöihin kuuluvat interaktioiden laukaisijat, toiminnot, funktiot, lausekesyntaksi ja muuttujien nimeämisrajoitukset. Sisällöt ovat muodoltaan taulukoita ja vapaamuotoista tekstiä. Dokumentaatiopohja on pyritty muotoilemaan siten, että sen voi julkaista toimeksiantajan projektinhallintajärjestelmässä.

Kieli suomi	Sivuja 53 Liitteet - Liitesivumäärä -

Asiasanat

uxpin, käyttäjäkokemussuunnittelu, prototyyppi, ux-prosessi

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2019
Degree Programme in Media

Tikkarinne 9
80200 JOENSUU
FINLAND
Tel. + 358 13 260 600 (switchboard)

Author (s)
Antton Hyvärinen

Title
UXPin design software for industrial IoT web application development

Commissioned by
Process Genius Oy

Abstract

This thesis showcases the UXPin design software for industrial IoT web development. UXPin is used to produce user interface designs and prototypes. The aim of this thesis was to address the basic features of UXPin in the context of user experience design. The theoretical part of the thesis discusses the principles of user-driven interface design, as well as factors that affect usability and user experience.

The functional part of the thesis was to gather documentation about the features and their constraints when creating interactions in UXPin. The contents of the documentation include interaction triggers, functions, functions, expression syntax and variable naming constraints. The contents are in the form of tables and text. The documentation base has been designed to be published in the client's project management system.

Language
Finnish

Pages 53
Appendices -
Pages of Appendices -

Keywords
uxpin, user experience design, prototype, ux-process

Sisältö

Sanasto

Dashboard	Kojelauta/koontinäkymä. UXPinissä dashboardiin sisältyvät projektinhallinta ja suunnittelujärjestelmät. (UXPin 2019a.)
Interaktio	Käyttäjän ja järjestelmän, käyttäjän ja elementin tai kahden elementin välinen vuorovaikutus, esimerkiksi tietokoneohjelmassa napin painaminen ja siitä seuraavan toiminnon aktivoituminen. (UXPin 2019b.)
IoT	Internet of things, eli asioiden internet. Termi viittaa älykkäisiin laitteisiin tai kokonaisuuksiin, jotka automaattisesti kommunikoivat keskenään internetin välityksellä. IoT laitteita tai kokonaisuuksia ovat esimerkiksi älykellot, älyjääkaapit tai älytalot. (Tikka 2014.)
Prototyyppi	Termi, jota käytetään tuotekehityksessä. Prototyyppi tarkoittaa varhaista versiota, jolla pyritään testaamaan tuotteen tai tuoteidean konseptia käytännössä, esimerkiksi verkkosovelluksen käyttöliittymää. (Kriik 2018.)
Trigger	UXPinin interaktion käynnistävä tekijä, eli laukaisija (UXPin 2019b).
UI	Englannin kielen sanoista *User Interface* tuleva lyhenne, joka tarkoittaa käyttöliittymää. Käyttöliittymä on rajapinta ihmisen ja koneen tai ihmisen ja ohjelmiston välillä. (Interaction Design Foundation 2018.)
UX	Englannin kielen sanoista *user experience* tuleva lyhenne, joka tarkoittaa käyttäjäkokemusta. Käyttäjäkokemus tarkoittaa niitä tuntemuksia, joita käyttäjä kokee, kun hän käyttää laitetta, ohjelmistoa tai palvelua. (Nielsen Norman Group 2019.)
UXPin	Graafisten käyttöliittymien suunnitteluun ja prototyypitykseen soveltuva ohjelmisto (UXPin 2019c).

1 Johdanto

1.1 Toimeksianto

Tämä opinnäytetyö käsittelee UXPin-suunnitteluohjelmistoa, sen perusominaisuuksia ja siihen liittyviä käytäntöjä teollisen internetin palveluiden muotoilussa. UXPin on työkalu, jota käytetään erilaisten digitaalisten käyttöliittymien suunnitteluun ja prototyyppien luomiseen. Opinnäytteen toiminnallisena osiona on laadittu dokumentaatiota UXPinin toiminnallisuuksista ja niihin liittyvistä käytännöistä toimeksiantajan projektinhallintajärjestelmään. Toimeksiantaja on Joensuussa ja Helsingissä toimiva Process Genius Oy, joka tuottaa ratkaisuja teollisen IoT:n tarpeisiin. Olen itse päätynyt harjoittelun kautta Process Geniuksen suunnittelutiimin työntekijäksi vajaat kaksi vuotta sitten.

Viime kesänä teimme vertailuarviota useille eri suunnitteluohjelmistoille, joita UXPinin lisäksi olivat Invision, Axure RP, Figma. Yksi vertailuarvioinnissa olleista ohjelmistoista oli myös Adobe XD, joka meillä oli ollut käytössä jo pidemmän aikaa. Syksyllä valitsimme UXPinin uudeksi työkaluksemme. Olennaisimpia kriteerejä valinnalle olivat mm. mahdollisuus luoda todenmukaisempia ja monipuolisempia prototyyppejä, prototyyppien helpompi jaettavuus, parempi dokumentaatio sekä tehokas kirjastointi käyttöliittymäelementeille. Suunnittelutiiminä tarvitsemme tietämystä hyvistä käytännöistä UXPinin parissa, jotta voisimme hyödyntää sitä tehokkaasti.

Opinnäytetyöni ydin on UXPin suunnitteluohjelmaan tutustuminen yleisellä tasolla, arviointi ja dokumentaation laatiminen siten, että se vastaisi toimeksiantajan suunnittelutiimin tarpeita ja hyödyttäisi mahdollisesti myös muitakin UXPinin käyttöä harkitsevia henkilöitä. UXPin on verrattain uusi suunnitteluohjelmisto, joten tässä opinnäytteessä esitellyt ominaisuudet tai toiminnallisuudet saattavat muuttua ajan kuluessa ja siksi dokumentaatiota

laatiessa on olennaista kiinnittää huomio erityisesti UXPinin interaktioiden osatekijöiden ja perustoiminnallisuuksien dokumentoimiseen.

1.2 Teollinen internet ja digitaaliset kaksoset

IoT, eli Internet of Things (esineiden internet) on terminä kuluttajalähtöinen, ja sillä viitataan älykkäisiin laitteisiin, kuten älykelloihin, älyjääkaappeihin, älykoteihin tai jopa älykkäisiin kaupunkeihin. Yhteistä näille laitteille tai kokonaisuuksille on se, että ne kommunikoivat keskenään internetin välityksellä. Kun puhutaan IIoT:sta (engl. Industrial internet Of Things) tai teollisesta internetistä, viitataan samaan teknologiaan, joka kuitenkin painottuu teollisuuteen ja sen tarpeisiin. (Tikka 2014.)

Teollisen internetin palveluihin liittyy useimmiten jonkinlaista analytiikkaa. Keräämällä dataa laitteista voidaan analysoida laitteiden, tai niiden käyttäjien, toimintaa ja tarkkailla niihin liittyviä prosesseja (Tikka 2014). Datan analysoinnilla voidaan optimoida energiankäyttöä, vähentää hukkakohteita ja jopa ennakoida riskejä.

Process Geniuksen tuotteiden kohdalla voidaan puhua ”digital twinistä”, eli digitaalisesta kaksosesta. Digitaalinen kaksonen on esimerkiksi responsiivisen verkkosovelluksen käyttöliittymään tuotu 3D-malli tehtaasta tai vaikka yksittäisestä laitteesta, jonka oheen tuodaan dataa reaalimaailmasta. Data tuodaan oikeista laitteista sensoreiden ja rajapintojen kautta käyttöliittymään ja liitetään oikeisiin paikkoihin 3D-mallissa. (Process Genius 2018.)

Tietokoneelle suurenkaan datamäärän käsittely ei ole ongelma, mutta ihmiselle massiivinen datamäärä on helposti liikaa, ellei se ole jäsennelty käytettävään

muotoon. *Digital Twinin* idea on nimenomaan tuoda data lähemmäs ihmistä. Aiemmin esimerkiksi kaikki koneisiin ja tavaroiden lähettämiseen liittyvät tarkastukset piti käydä tekemässä paikan päällä. Nykyään suurestakin tuotantoympäristöstä voidaan tuottaa digitaalinen kaksonen, josta voi tarkastaa kaikki edellä mainitut asiat lyhyellä vilkaisulla. (Process Genius 2018.)

1.3 Verkkosovellukset

Irmeli Sinkkonen jakaa teoksessaan *Helppokäyttöisen verkkopalvelun suunnittelu* verkkopalvelut kahteen luokkaan niiden sisältöjen perusteella; staattisiin ja dynaamisiin. Staattisella sisällöllä tarkoitetaan sellaista sisältöä, joka säilyy muuttumattomana pitkiä aikoja ja on kaikille käyttäjille sama. Dynaaminen sisältö on jatkuvassa muutoksessa eri käyttökertojen välillä. Muutoksia voi tehdä sivuston ylläpitäjä tai sivuston omat käyttäjät. Kun käyttäjät voivat tehdä muutoksia sisältöön, sisältöä voidaan kuvailla *toiminnalliseksi.*

Verkkosovelluksia kutsutaan *operatiivisiksi palveluiksi* ja niiden sisältö on aina toiminnallista. Esimerkiksi verkkopankit ja verkkokaupat ovat verkkosovelluksia. Sinkkosen mukaan raja verkkosivuston ja verkkosovelluksen välillä on kuitenkin häilyvä, koska useimmat verkkosovellukset ovat hybridejä, joissa on sivuja sekä staattiselle informaatiolle, että toiminnallisille prosesseille. Verkkosovellukset tukevat toimintoketjuja, joilla on jonkinlainen näkyvä vaikutus ulkomaailmassa. Tyypillisiä esimerkkejä verkkosovelluksista ovat verkkokaupat ja verkkopankit. (Sinkkonen 2009, 26.)

Suurin osa Process Geniuksen tuottamista palveluista on verkkosovelluksia. Process Geniuksen tuotteissa se voi tarkoittaa esimerkiksi laitteisiin tai ympäristöihin vaikuttamista verkkosovelluksen kautta. Hyvänä esimerkkinä verkkosovelluksesta ja aiemmin mainitusta *digitaalisesta kaksosesta* on Itä-

Suomen Yliopiston Sm4rtlab-hanke. Mallinnetun laboratorioympäristön avulla voi ohjata oikean laboratorioympäristön laitteita tietokoneella, tai vaikka AR-lasien avulla. (Elisa 2017.)

Progressiivinen verkkosovellus (PWA, progressive web application) pyrkii yhdistelemään natiivisovelluksen ja verkkosovelluksen parhaita ominaisuuksia ja parantaa palvelun käyttäjäkokemusta etenkin mobiililaitteilla. PWA:n voi toteuttaa siten, että se muistuttaa enemmän natiivisovellusta kuin verkkosovellusta, koska siitä voidaan karsia esimerkiksi verkkoselaimen omat käyttöliittymäelementit kokonaan pois. (Itewiki 2019.)

2 UI/UX -suunnittelu

2.1 Käyttöliittymät

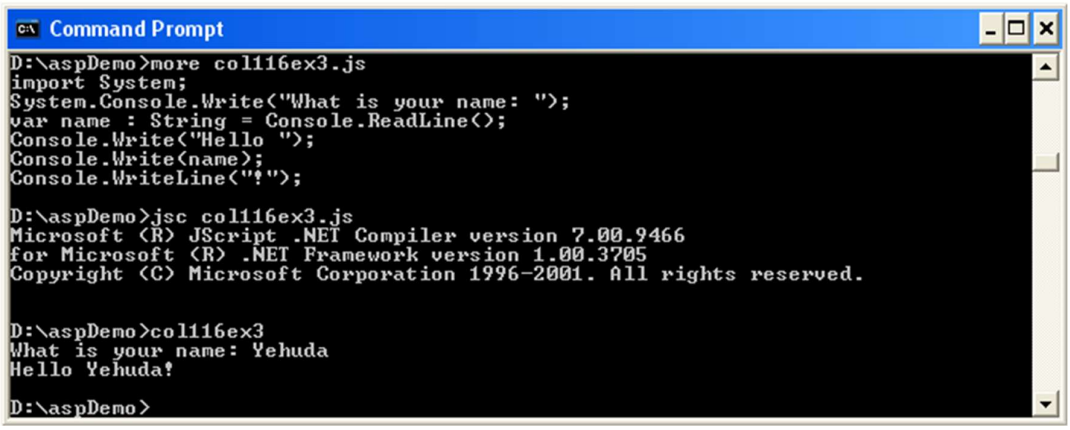
Käyttöliittymään ja käyttäjäkokemukseen viittaavat UI ja UX -lyhenteet tulevat sanoista *user interface,* eli käyttöliittymä, ja *user experience,* eli käyttäjäkokemus*.* UI ja UX -lyhenteet ja niiden merkitykset menevät monilta helposti sekaisin, eikä ihmekään, kun ne liittyvät niin vahvasti toisiinsa. Contrast.fi –sivustolla julkaistussa artikkelissaan Joonas Virtanen kuvailee hyvin UI:n ja UX:n eroa analogialla taideteoksen katsomisesta: taideteoksen sinussa herättämät tunteet ovat UX ja taideteoksen tekniikka, tyyli ja värit ovat UI. (Virtanen 2018.)

Käyttöliittymällä tarkoitetaan sitä laitteen, tai ohjelman osaa, jonka avulla käyttäjä vastaanottaa ja syöttää tietoa (Helsingin Yliopisto 2019). Käyttöliittymällä voidaan tarkoittaa myös välineitä tai kieliä, joiden avulla käyttäjä on yhteydessä ATK-laitteistoon. (Immonen 2003.) Erikoistapauksissa laitteen käyttöliittymä voi olla myös esimerkiksi ääni- tai liikeohjattu (Interaction Design Foundation 2018).

Tällaiset ratkaisut voivat olla hyödyllisiä tietyissä tapauksissa, esimerkiksi ääniohjattua käyttöliittymää voi käyttää turvallisesti ajaessa autoa tai sitä voi käyttää myös henkilö, jolla on heikko näkökyky.

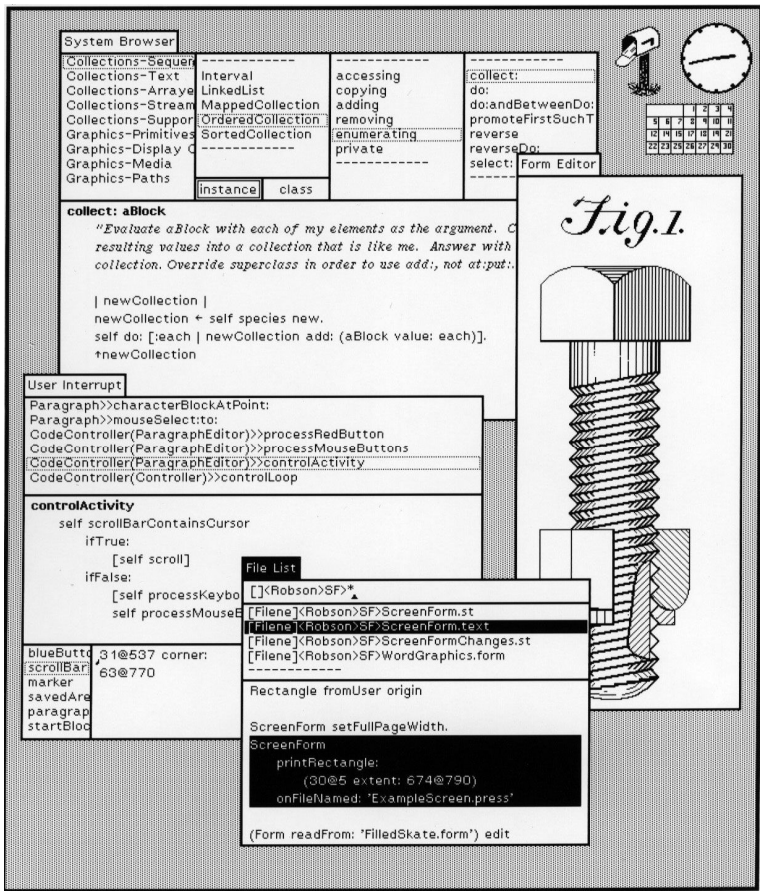
Käyttöliittymäsuunnittelijan työhön liittyy monia osa-alueita, joita ovat mm. visuaalinen suunnittelu, interaktioiden suunnittelu ja tiedon hierarkian suunnittelu. UXPinillä voi tehdä näitä kaikkia, joskin visuaalisen suunnittelun työkalut eivät vedä vertoja esim. Adoben suunnitteluohjelmistoille, kuten Photoshopille tai Illustratorille. Verkkosovellusten käyttöliittymäsuunnittelu on ennen kaikkea visuaalista suunnittelua ja se keskittyy siihen, miltä esimerkiksi verkkosovellus visuaalisesti näyttää vaikkapa värien, kuvien, animaatioiden tai typografian suhteen.

Tietotekniikan alkutaipaleella tietokoneilla pyöri vain ohjelmia, joiden käyttöliittymät olivat merkkipohjaisia (character-based user interface, CUI), joita ajettiin konsolin tai terminaalin kautta (Kuva 1). Perusidea tämäntyyppisissä ohjelmistoissa on yksinkertainen; käyttäjä kirjoittaa näppäimistön avulla ohjelmalle *syötteen* ja ohjelma ”tulostaa” *vasteen.* Tähän ei välttämättä tarvita edes minkäänlaista näyttölaitetta, vaan riittää että on tulostin, joka tulostaa paperille vasteen käyttäjän antamalle syötteelle. Komentoriviohjelmien vasteista käytetään edelleen nimitystä tuloste, vaikka tulosteet tulevatkin näytöllä näkyvään konsoli-ikkunaan eikä paperille. (Computer Hope 2017.)



Kuva 1. Esimerkki merkkipohjaisesta käyttöliittymästä. Konsoli-ikkunassa ajetaan yksinkertaista ohjelmaa, joka pyytää syötteeksi käyttäjän nimen ja tulostaa vasteeksi: ”Hello + (käyttäjän syöttämä nimi)+!”

Kehittyneemmissä ohjelmistoissa on kuitenkin käytön helpottamiseksi graafiset käyttöliittymät (graphic user interface, GUI), joita ohjataan jollain osoitinlaitteella, yleensä hiirellä. Kosketusnäyttöjen yleistymisen myötä tarve osoitinlaitteille on kuitenkin vähentynyt. Suunnittelijan on huomioitava tämä aspekti ja tiedettävä, millaisille laitteille käyttöliittymää ollaan toteuttamassa. Graafisissa käyttöliittymissä komennot on liitetty erilaisiin graafisiin elementteihin, esimerkiksi nappeihin, joita painelemalla ohjelma suorittaa erilaisia komentoja. Kun komennot on liitetty graafisiin elementteihin, käyttäjän ei tarvitse muistaa tai välttämättä edes ymmärtää monimutkaisia kirjoitettavia komentoja käyttääkseen ohjelmaa. Tästä seuraa se, että ohjelman oppiminen tapahtuu nopeammin, eikä tietokoneosaamista tarvitse juurikaan olla. (ITPRO 2019.)



Kuva 2. Xerox Alto -tietokoneen käyttöliittymä. Xerox Alto oli ensimmäinen tietokone, jossa oli graafinen käyttöliittymä. (Computer History Museum 2019.)

Nykypäivänä siis digitaalisista laitteista puhuttaessa ”käyttöliittymä” tarkoittaa digitaalisen palvelun visuaalista osaa, jonka kautta käyttäjä suorittaa toimintoja, mutta tulevaisuudessa se saattaa tarkoittaa jotain ihan muuta. Suunnittelijat kohtaavat jo nyt monenlaisia haasteita, kun digitaalisia palveluita pyritään tuomaan pienille tai jopa seinän kokoisille näytöille. Alustat saattavat vaihdella älykoruista, älyvaatteisiin, älypuhelimiin, tabletteihin, suurikokoisiin paneeleihin, projektorinäyttöihin tai jopa valaistuihin rakennuksiin. Käyttöliittymien tulee mukautua näyttökokojen lisäksi myös eri kielille, saavutettavaksi ihmisille, joilla on erilaisia käyttöä haittaavia vammoja tai sairauksia sekä alueille, joilla verkkokaistanleveydet vaihtelevat.

Jotkut innovaattorit uskovat, että perinteiset tietokoneet graafisine käyttöliittymineen tulevat katoamaan ja uudet käyttöliittymät tulevat olemaan läsnä kaikkialla ja sulautuvat ympäristöön. Tällaiset uudet teknologiat voisivat olla tilannetajuisia, huomaavaisia ja tarkkanäköisiä sekä pystyisivät vastaamaan käyttäjän tarpeisiin. Käyttäjälle ne antaisivat palautetta ympäristön kautta esimerkiksi loistaen, humisten, muotoa muuttaen tai ilmavirtaa puhaltaen. (Shneiderman, Plaisant, Cohen, Jacobs, Elmqvist, Diakopoulos 2017, 30.)

2.2 Erilaisia käyttöliittymäelementtejä

Käyttöliittymäelementit ovat käyttöliittymän yksittäisiä osia. Käyttöliittymää suunnitellessa tulisi olla johdonmukainen ja ennakoitava käyttöliittymän elementtejä valittaessa. Valtaosa käyttäjistä tuntee kyllä erilaiset elementit ja tietää, kuinka niiden tulisi toimia ja millaisiin yhteyksiin ne sopivat. Usability.gov -sivustolla on taulukko, jossa on lueteltu tyypillisiä käyttöliittymän elementtejä (Usability 2019). Käyttöliittymän elementtejä tyypillisesti ovat, mutta eivät kuitenkaan rajoitu seuraaviin:

Tulo-ohjaimet (engl. input controls): valintaruudut (engl. check-box), valintanapit (radio buttons), pudotuslistat (engl. dropdown lists), listalaatikot (engl. list boxes), napit (engl. buttons), vaihtokytkimet (engl. toggles), tekstikentät (text fields) ja päivämääräkentät (engl. date fields) (Usability 2019.)

Navigaatiokomponentit (engl. navigation components): murupolku (engl. breadcrumb), liukusäädin (engl. slider), tunnisteet (engl. tags), ja ikonit (engl. icons) (Usability 2019.)

Informatiiviset komponentit (engl. informational components): työkaluvihjeet (engl. tooltips), ikonit, edistymispalkit (engl. progress bar), ilmoitukset (engl. notifications), viestilaatikot (engl.message boxes) ja modaaliset ikkunat (engl. modal windows) (Usability 2019.)

Kuva 3. Esimerkkejä käyttöliittymäelementeistä. Elementit on ladottu piirtoalueelle UXPinin valmiista kirjastosta.

UXPinissä on todella laaja kirjasto valmiita käyttöliittymäelementtejä, joita voi halutessaan käyttää suunnittelussa. Suurimmassa osassa UXPinin tarjoamista käyttöliittymäelementeistä ei ole mitään toiminnallisuuksia, mutta on myös elementtejä, joilla on omat erityiset asetuksensa ja interaktionsa.

2.3 Käytettävyys

Käytettävyydellä tarkoitetaan järjestelmän laatua mittaava ominaisuutta. ISO 9241-11:2018 -standardi, *Ihmisen ja järjestelmän vuorovaikutuksen ergonomia,* tarjoaa kehyksen käytettävyyden käsitteen ymmärtämiseksi ja sen soveltamiseen tilanteisiin, joissa ihmiset käyttävät erilaisia vuorovaikutteisia järjestelmiä. Näihin järjestelmiin luetaan esimerkiksi erilaiset digitaaliset palvelut, fyysiset ympäristöt, tuotteet (niin teolliset kuin kuluttajatuotteetkin) ja palvelut (tekniset ja henkilökohtaiset palvelut). ISO 9241-11:2018 -standardi vapaasti suomennettuna

määrittelee käytettävyyden mittariksi, jolla mitataan järjestelmän, tuotteen tai palvelun käyttökelpoisuutta, tehokkuutta ja miellyttävyyttä tietyillä käyttäjillä tietyissä käyttötarkoituksissa. (International Organization for Standardization 2018.)

Jakob Nielsenin mukaan (Nielsen, 2012) käytettävyydelle voidaan asettaa tavoitteita ja tavoitteiden toteutumista voidaan arvioida viidellä laatua määrittelevällä tekijällä: opittavuudella, tehokkuudella, muistettavuudella, virheettömyydellä ja tyytyväisyydellä. Olen referoinut edellä mainittujen käytettävyystekijöiden määritelmät teoksesta *Designing The User Interface: Strategies for Effective Human-Computer Interaction* (Schneiderman ym. 2017, 33-34). Lisäksi olen poiminut esimerkkejä käytettävyystekijöiden arvioinnille teoksesta *Graafisen käyttöliittymän suunnittelu: Opas ohjelmistojen käytettävyyteen* (Koivunen & Nieminen 1995, 22-24).

Opittavuus mittaa sitä, kuinka helposti uusi käyttäjä oppii järjestelmän käytön ohjekirjan tai opastuksen avulla (Schneiderman ym. 2017, 33). Mitatut oppimisajat vaihtelevat riippuen järjestelmän koosta tai käyttäjäryhmistä. Monimutkaisen järjestelmän ylläpitäjälle voidaan hyväksyä pidempi oppimisaika, kuin järjestelmän peruskäyttäjälle. Järjestelmän muokattavuus omiin tarkoituksiin, joka edeltää varsinaista käyttöönottoa, on otettava myös huomioon oppimisajassa (esimerkiksi erilaiset tekstieditorit, joita käytetään ohjelmistosuunnittelussa tai kuvanmuokkausohjelmat, joissa voi olla useita eri työtiloja ja käyttäjien luomia kirjastoja). Useimmissa tapauksissa on kuitenkin turvallista olettaa, että käyttäjät eivät ole tietoteknisesti harjaantuneita ja siksi opittavuuteen panostaminen on tärkeää. (Koivunen & Nieminen 1995, 22-23.)

Tehokkuudella tarkoitetaan sitä, kuinka nopeasti järjestelmän oppinut käyttäjä suorittaa annetut tehtävät (Schneiderman ym. 2017, 34). Tehokkuuden mittaamista varten on olennaista määritellä haluttu taso, johon tähdätään. Hyväksyttävä tehokkuuden taso voidaan määritellä esimerkiksi keskiarvona

testikäyttäjien mittaustuloksista tai aikana, joka tietyn prosenttimäärän käyttäjistä on alitettava tehtävää suorittaessaan. (Koivunen & Nieminen 1995, 23.)

Muistettavuudella tarkoitetaan sitä, kuinka hyvin järjestelmän oppinut käyttäjä suoriutuu annetuista tehtävistä pidemmän käyttötauon jälkeen (Schneiderman ym. 2017, 34). Muistettavuus on tärkeää järjestelmissä tai järjestelmän toiminnoissa, joita käytetään vain harvoin. Jos järjestelemän oppiminen on helppoa uudelle käyttäjälle, on todennäköistä, että järjestelmä on myös helpommin muistettavissa. (Koivunen & Nieminen 1995, 23-24.)

Virheettömyydellä tarkoitetaan sitä, että järjestelmän tulisi olla sen verran selkeä ja johdonmukainen, jotta virheitä syntyisi mahdollisimman vähän ja vaikka niitä syntyisikin, niiden korjaamiseen ei kulu kauaa aikaa ja ne eivät vaikuta merkittävästi lopputulokseen (Schneiderman ym. 2017, 34). Virheiden määrään voidaan vaikuttaa käyttäjän selkeällä ohjeistuksella. On kahdenlaisia virheitä; *operaatiotason virheitä* ja *tavoitetason virheitä*. Operaatiotason virheillä tarkoitetaan esim. virheitä käyttäjän syötteissä, kun taas tavoitetason virhe voi syntyä, kun käyttäjä ymmärtää jotain väärin, eikä siksi pääse haluttuun tavoitteeseen. (Koivunen & Nieminen 1995, 24.)

Subjektiivisella tyytyväisyydellä mitataan, kuinka paljon käyttäjä piti käyttöliittymän eri aspekteista (Schneiderman ym. 2017, 34). Kun käyttäjä on käyttämäänsä järjestelmään tyytyväinen, hän käyttää järjestelemää mielellään ja yleensä myös tehokkaasti. Jos järjestelmän käyttö on epämukavaa, käyttäjä ei välttämättä halua tutustua järjestelmään syvällisemmin ja silloin järjestelmän koko potentiaali ei välttämättä nouse esille. Käyttäjätyytyväisyyttä voidaan arvioida kvalitatiivisilla tutkimusmenetelmillä, eli tekemällä haastatteluja tai erilaisia kyselyitä. (Koivunen & Nieminen 1995, 24.)

Edellä mainitut tekijät ovat suhteellisia toistensa kanssa ja ”huono” tulos yhden tekijän kohdalla voi vaikuttaa positiivisesti toiseen tekijään. Ohjelman pitkä

oppimisaika voi esimerkiksi näyttäytyä lisääntyneenä tehokkuutena, jos käyttäjällä on mahdollisuus muokata toiminnallisuuksia tai on käytettävissään useita pikakomentoja. Niiden opetteluun kuluu aikaa, mutta lopulta nopeuttavat tehtävien suorittamista. Käytettävyystekijöihin kannattaakin kiinnittää huomiota tapauskohtaisesti, sillä joskus yksi tekijä voi olla toista tärkeämpi. Suunnittelijoiden ja projektipäälliköiden olisi hyvä olla tietoisia tällaisista ”vaihtokaupoista” käytettävyystekijöiden kanssa, jotta osataan asettaa suunnittelulle projektiin sopivat päämäärät. (Shneiderman ym. 2017, 34.)

2.4 Käyttäjäkokemus

Käyttäjäkokemuksella tarkoitetaan kaikkia niitä tuntemuksia, joita loppukäyttäjä kokee, kun hän on vuorovaikutuksessa yrityksen, yrityksen palveluiden tai sen tuotteiden kanssa. Käyttäjäkokemukset ovat subjektiivisia, ajasta ja kontekstista riippuvaisia. Digitaalisista palveluista puhuttaessa on tärkeää erottaa kokonaisvaltainen käyttäjäkokemus käyttöliittymästä, vaikka käyttöliittymän osuus onkin merkittävä käyttäjäkokemuksen muodostumisessa. (Nielsen Norman Group 2019.) ISO 9241-11:2018 -standardissa (International Organization for Standardization 2018) on neljä ydinkohtaa käyttäjäkokemuksen määrittelylle:

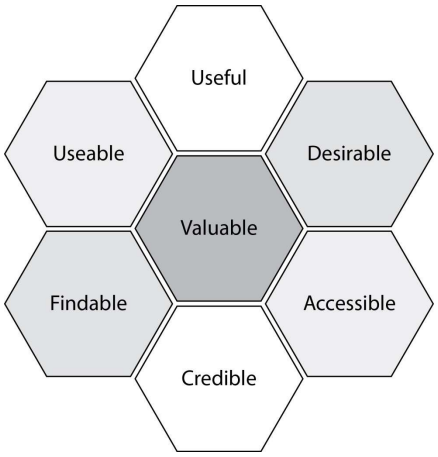
1. Käyttäjäkokemukseen kuuluvat kaikki käyttäjän kokemat tunteet, uskomukset, mieltymykset, havainnot, fyysiset ja psykologiset reaktiot, käyttäytyminen ja aikaansaannokset, jotka ilmenevät ennen käyttöä, käytön aikana ja käytön jälkeen.
2. Käyttäjäkokemus on seurausta tuotemerkistä, esityksestä, toiminnallisuudesta, suorituskyvystä, vuorovaikutteisesta käyttäytymisestä ja vuorovaikutteisen järjestelmän avustavista ominaisuuksista, käyttäjän sisäisestä ja fyysisestä tilasta, joka johtuu aiemmista kokemuksista, asenteista, taidoista ja persoonallisuudesta sekä käyttöyhteydestä.
3. Termiä "käyttäjäkokemus" voidaan käyttää myös viittaamaan osaamiseen tai prosesseihin, kuten käyttäjäkokemukseen, käyttäjäkokemuksen suunnitteluun, käyttäjäkokemusmenetelmään, käyttäjäkokemuksen

arviointiin, käyttäjäkokemuksen tutkimukseen, käyttäjäkokemuksen osastoon.

4. Ihmiskeskeinen suunnittelu voi hallita vain niitä käyttäjäkokemuksen osa-alueita, jotka johtuvat interaktiivisen järjestelmän suunnitelluista näkökohdista.

Käyttäjäkokemukseen vaikuttavat tekijät voi jakaa karkeasti pragmaattisiin ja hedonistisiin tekijöihin. Pragmaattisia, eli käytännönläheisiä tekijöitä ovat esimerkiksi aiemmin mainitut käytettävyystekijät tehokkuus, hyödyllisyys, virheettömyys jne. Hedonistisia, eli nautinnollisia, tekijöitä ovat käyttömukavuuteen liittyvät tekijät esimerkiksi käyttöliittymän houkuttelevuus, hieno visuaalinen ulkoasu ja hauskuus. (Hassenzahl ja Tractinsky 2006, 91-95.)

Käyttäjäkokemuksen käsitettä voi avata syvemmin erilaisilla kaavioilla ja malleilla. Eräs tunnetuimmista malleista on Peter Morvillen (2004) luoma hunajakennomalli (Kuva 3). Malli kuvaa käyttäjäkokemussuunnitteluun kuuluvia osa-alueita verkkopalveluissa, joita on Morvillen mukaan on seitsemän: käytettävyys (useable), hyödyllisyys (useful), löydettävyys (findable), saavutettavuus (accessible), luotettavuus (credible), haluttavuus (desirable) ja arvokkuus (valuable). Olennaisinta on, että verkkosivustolla on jonkinlaista arvoa käyttäjälleen, siksi arvokkuus on laitettu kennon keskelle. (Morville 2004.)



Kuva 4. Peter Morvillen hunajakennomalli (Morville 2004).

Käytettävyys. Helppokäyttöisyys on tärkeää, mutta ihmisen ja tietokoneen välisen vuorovaikutuksen rajapintakeskeiset menetelmät ja näkökulmat eivät koske kaikkia verkkosuunnittelun ulottuvuuksia. Käytettävyys on välttämätöntä, mutta se ei Morvillen mukaan riitä hyvään käyttökokemukseen. (Morville 2004.)

Hyödyllisyys. Verkkopalvelun tulisi vastata johonkin tarpeeseen. Jos verkkopalvelu ei ole hyödyllinen, sitä tuskin tullaan käyttämään kovinkaan paljoa, jos ollenkaan.

Löydettävyys. Käyttäjän tulisi helposti löytää haluamansa verkkopalvelu ja pystyä navigoimaan siellä vaivattomasti sekä löytää palvelusta haluamansa tieto/toiminnallisuus.

Saavutettavuus. Verkkopalvelun tulisi olla saavutettavissa ihmisille, joilla on erilaisia invalidisoivia vammoja tai sairauksia (yli 10% väestöstä). Saavutettavuus edistää liiketoiminta ja on myös eettisesti oikein. Myöhemmin käsittelemässäni UXPin suunnitteluohjelmistossa on työkaluja, joilla voi edesauttaa käyttöliittymän saavutettavuutta.

Luotettavuus. Tiedon pitäisi olla totuudenmukaista ja se pitäisi esitellä siten, että se on myös uskottavaa. Käyttäjän ei esimerkiksi pitäisi joutua pelkäämään omien tietojensa antamista verkkosivustolle.

Haluttavuus. Verkkopalvelun haluttavuuteen vaikuttavat tekijät ovat tunnesidonnaisia. Haluttavuuteen voivat vaikuttaa esimerkiksi verkkopalvelun ulkonäkö, maine muiden käyttäjien keskuudessa tai samaistuttava brändi.

Arvokkuus. Tärkein osa-alue, joka vastaa kysymykseen: mitä lisäarvoa verkkopalvelu tuo käyttäjälleen, jota hän ei muualta saisi?

Nielsen Norman Groupin verkkosivuilla käyttäjäkokemusta määrittelevässä julkaisussa on esimerkkinä verkkopalvelu, jolta käyttäjä voi etsiä elokuva-arvosteluja. Kyseinen verkkopalvelu voi soveltua käyttöliittymältään täydellisesti elokuva-arvostelun etsimiseen, mutta jos sivuston tietokanta sisältää arvosteluja

ainoastaan suurempien studioiden elokuvilta, se on käyttäjäkokemukseltaan huono sellaiselle käyttäjälle, joka haluaa löytää pienen budjetin indie-elokuvien arvosteluja. (Nielsen Norman Group 2018.) Esimerkin tapauksessa indie-elokuvista kiinnostunut käyttäjä on jäänyt käyttäjäpersoonien määrittelyvaiheessa kohderyhmän ulkopuolelle. Käyttäjäpersoonien määrittely on olennainen osa käyttäjäkeskeistä suunnittelua, jolla pyritään luomaan käyttäjälle hyvä kokemus.

2.5 Käyttäjälähtöinen suunnittelu

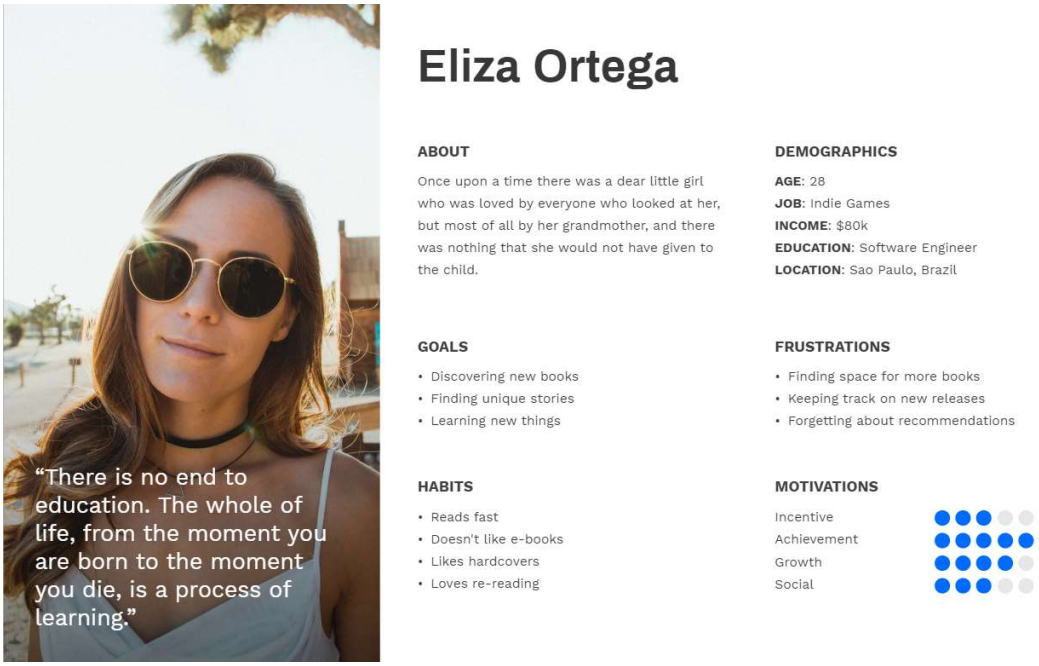
Käyttäjälähtöisessä, tai käyttäjäkeskeisessä (user-centered-design, UCD), suunnittelussa on kyse suunnittelumenetelmistä, joilla pyritään varmistamaan suunniteltavan järjestelmän, laitteen tai ympäristön käytettävyys ja hyvä käyttäjäkokemus. Käyttäjälähtöinen suunnittelu on iteratiivinen suunnitteluprosessi, jossa suunnittelijan huomio on käyttäjissä ja heidän tarpeissaan suunnitteluprosessin jokaisessa vaiheessa. Menetelmät rakentuvat yleensä liiketoiminnallisten tavoitteiden ympärille ja käyttäjätutkimukselle. (Sinkkonen, Nuutila, Törmä. 2009, 27.) Käyttäjäkeskeisen suunnittelun perusidea on selvittää, millaisia käyttäjät ovat. Käyttäjätutkimuksessa on huomioitava sekä nykyiset, että potentiaaliset tulevat käyttäjät. (Interaction Design Foundation 2019.)

Syntagmin toimitusjohtaja, UX-strategi ja kirjailija William Hudson kertoo käyttäjälähtöisen suunnittelun prosessista Interaction Design Foundationin artikkelin ohessa julkaistussa videossa. Hudson kannustaa suunnittelijoita pyrkimään suoraan vuorovaikutukseen loppukäyttäjien kanssa ja havainnoimaan heitä työssään. (Interaction Design Foundation 2019.) Käyttäjälähtöisen suunnitelman edut näkyvät käyttäjälle hyvänä käyttökokemuksena, mutta menetelmät helpottavat myös suunnittelijan työtä erilaisissa

päätöksentekovaiheissa, kun päätökset perustuvat tutkittuun tietoon, eikä umpimähkäiseen arvailuun (Sinkkonen ym. 2009, 27).

Suunnittelua voi konkretisoida luomalla *käyttäjäpersoonia* ja *tarinoita*, joiden on tarkoitus antaa eväitä hyvän käyttäjäkokemuksen luomiseen. Käyttäjäpersoona on esimerkki kuvitteellisesta käyttäjästä, joka on tavalla tai toisella tekemisissä palvelun kanssa (Aalto Yliopisto 2011). Kun käyttäjäpersoonalle annetaan kasvojen ja nimen lisäksi tavoitteet, motivaattorit ja mahdolliset ärsytystä aiheuttavat tekijät, on suunnittelijan helpompi kokea empatiaa loppukäyttäjää kohtaan ja ikään kuin katsoa maailmaa heidän kauttaan. Käyttäjiin viittaaminen “peruskäyttäjäjinä” tai “ylläpitäjinä” luo helposti mielikuvan kasvottomasta massasta, joka on kuvauksena suppea ja suunnittelijan on vaikeampi samastua heihin. (Interaction Design Foundation 2019.)

Käyttäjäpersoonien avulla päätöksien tekeminen suunnittelussa on tarkempaa, kun pystytään realistisemmin arvioimaan loppukäyttäjien tarpeita. Käyttäjäpersoonien luominen on iteratiivinen prosessi. Olennaista käyttäjäpersoonien luonnissa on, että se tapahtuu todellisia käyttäjiä koskevien havaintojen kautta, eikä arvausten kautta. (Interaction Design Foundation 2019.)



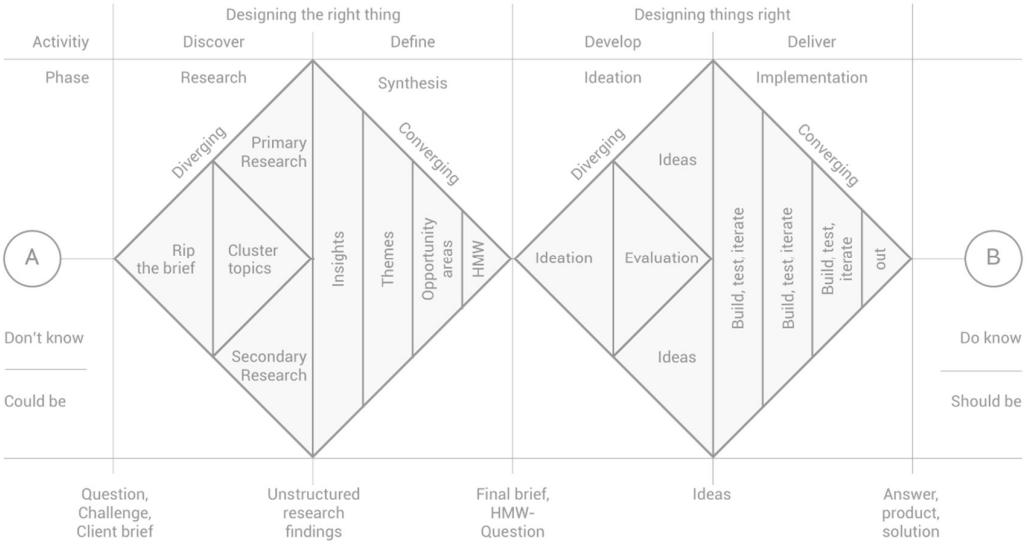
Kuva. 5. UXPinin peruspohja käyttäjäpersoonalle.

Käyttäjäpersoonien määrittelyn jälkeen voidaan aloittaa tarinoiden luominen. Tarinoilla tarkoitetaan erilaisia skenaarioita, joita ovat *toimintatarinat* ja *käyttötarinat*. Toimintatarinat kertovat, miten käyttäjä on toiminut ennen uutta palvelua ja käyttötarinoissa kuvaillaan, millaisia toimintamahdollisuuksia uusi palvelu tarjoaa. (Sinkkonen ym. 2009, 171-173.) Koottujen käyttötarinoiden pohjalta voidaan lähteä rakentamaan rautalankamalleja ja sitä kautta prototyyppiä. Ideaalitilanteessa prototyypin avulla voidaan aloittaa käytettävyysarviointi ja testaus jo projektin varhaisessa vaiheessa.

2.6 UX-suunnittelu kaksoistimantti-prosessimallin mukaisesti

Brittiläinen Suunnitteluneuvoston (Design Council) kehittämä *Suunnitteluprosessin kaksoistimantti,* on tunnettu käyttäjälähtöisen suunnittelun

prosessimalli. Sen kuvaama prosessi koostuu neljästä päävaiheesta: *löydä, määrittele, kehitä* ja *toimita.*



Kuva 6. Dan Nesslerin jatkokehittämä versio *Käyttäjäkokemuksen Suunnitteluprosessin Kaksoistimantti* -kaaviosta. Ensimmäinen timantti keskittyy tutkimukseen ja toinen suunnitteluun. (Nessler 2016.)

Löytämisvaiheen tavoitteena on tunnistaa ongelma, mahdollisuus tai tarve johon suunnittelun tulisi vastata. Tässä vaiheessa suunnittelijat pyrkivät katsomaan ongelmaa laatikon ulkopuolelta, etsimään inspiraatiota sekä asettumaan käyttäjän asemaan. Olennaisimpia metodeja ovat markkinatutkimus, käyttäjätutkimus, johtamis- ja suunnittelu ja tutkimusryhmien suunnittelu.

Määrittelyvaiheessa pyritään jäsentelemään edellisen vaiheen tutkimustuloksia ja löytämään projektin pääkohdat. Millä on eniten merkitystä? Mikä vaatii ensimmäisenä huomiota? Tässä vaiheessa tavoite on muodostaa selkeä, luova kuvaus, joka kehystää olennaisimman suunnitteluhaasteen. Olennaista

määrittelyvaiheessa on luoda projektille ”product backlog” eli tuotteen kehitysjono, joka kuvaa työtehtäviä, joita projekti sisältää.

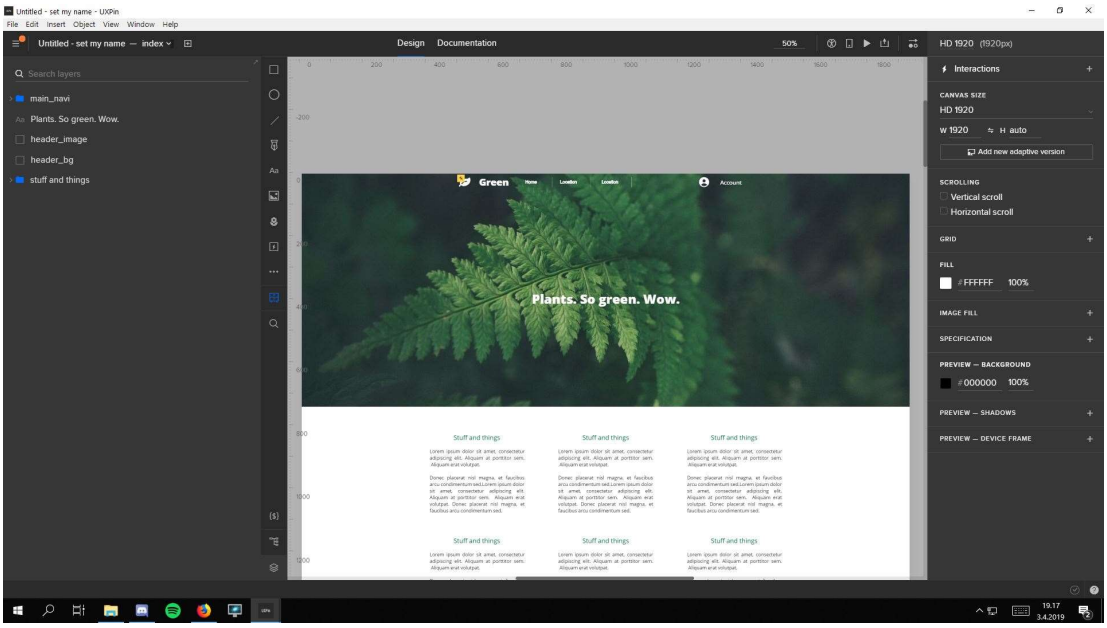
Kehitysvaiheessa pyritään luomaan ratkaisuja aiemmin määriteltyyn ongelmaan. Erilaisia ratkaisuja testataan prototyyppien avulla ja testaamisen pohjalta tehdään iterointia ja jatkojalostetaan ratkaisuja. Tyypillistä tälle vaiheelle on brainstormaus, prototyypitys, monialainen yhteistyö, visuaalinen suunnittelu, kehitysmetodit ja käyttäjätestaaminen.

Toimitus on Kaksoistimantin viimeinen vaihe, jossa valmis tuote viimeistellään ja julkaistaan. Vaiheeseen kuuluu lopputestaus, hyväksyntä ja julkaisu, sekä arviointi ja palautesilmukat.

3 UXPIN

3.1 Mikä UXPin on ja mihin sitä käytetään?

UXPin on suunnittelun ja prototyypityksen työkalu, jota käytetään palvelun muotoilussa. Se on progressiivinen verkkosovellus, joka mahdollistaa useamman käyttäjän yhtäaikaisen työskentelyn saman projektin parissa, ja sitä kautta suunnitelmien tehokkaan jakamisen. Kaikki projektin sisällä tapahtuvat muutokset päivittyvät internetin kautta nopeasti myös muiden käyttäjien nähtäväksi. Samat muutokset päivittyvät myös prototyyppeihin jopa niiden ajon aikana.



Kuva 7. UXPinin suunnittelunäkymä Windows 10 käyttöjärjestelmällä. Vasemmassa reunassa on tasohierarkia ja oikeassa reunassa valitun elementin tai projektin pohjan yksityiskohdat. Yläreunassa välilehdet ”Design” ja ”documentation”.

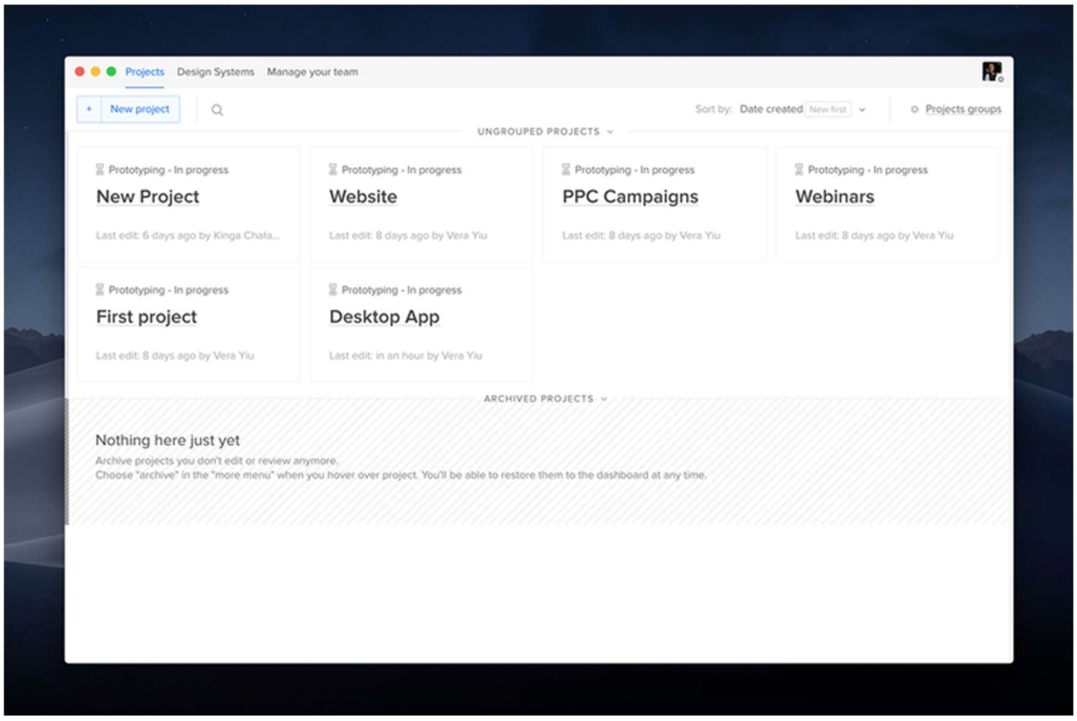
UXPinistä on sekä työpöytäversio että verkkosovellusversio. UXPinillä on myös oma mobiilisovellus, UXPin Mirror, joka helpottaa prototyyppien ajamista älypuhelimella. UXPiniä ja sen työkaluja käsittelevät luvut on kirjoitettu työpöytäsovellusta silmällä pitäen ja jotkut ominaisuudet saattavat poiketa tai puuttua kokonaan verkkosovellusversiosta.

3.2 Dashboard

3.2.1 UXPinin projektinhallinta

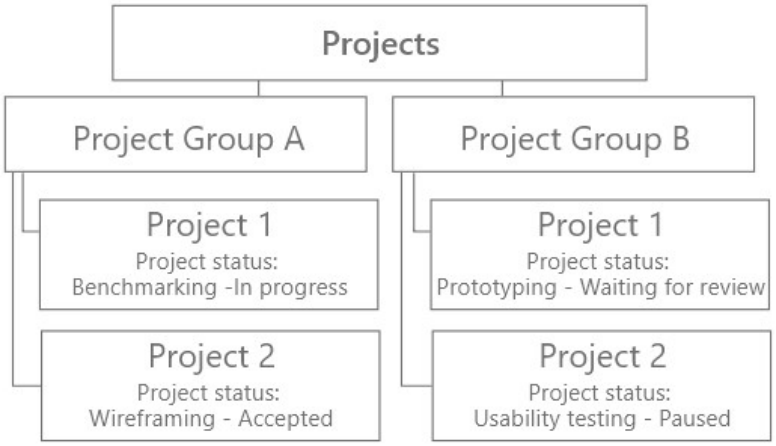
Dashboard, eli vapaasti käännettynä ”kojelauta”, on eräänlainen koontinäkymä kaikista projekteista ja suunnittelujärjestelmistä. Dashboardin yläpalkissa on kaksi välilehteä: projects (projektit) ja design system (suunnittelujärjestelmä).

Projektinäkymässä voi luoda ja hallinnoida suunnitteluprojekteja. Näkymään voi luoda ja nimetä projektiryhmiä, jotka auttavat hallinnoimisessa, kun projekteja on useita. Projektinäkymän lajitteluperusteeksi voi määritellä nimen (A-Z tai Z-A), luomispäivän (uusin ensin tai vanhin ensin), muokkauspäivämäärän (uusin ensin tai vanhin ensin). Lajitteluperuste vaikuttaa ainoastaan projektiryhmien sisäiseen järjestykseen, eli lajitteluperusteesta huolimatta eri projektiryhmien projektit eivät sekoitu keskenään. Jokaisen projektin sisällä voi olla useita eri prototyyppejä ja jokaisessa prototyypissä voi olla useita eri sivuja.



Kuva 8. UXPinin projektinäkymä iOS käyttöjärjestelmällä (UXPin 2019a).

Projektien ryhmittelyn lisäksi jokaiselle projektille voi lisätä suunnitteluprossin eri työvaiheita (eng. project stage), jotka kertovat missä vaiheessa tuotantoa projekti on. UXPinin tarjoamat oletustilat ovat ”benchmark” (vertailuarviointi), ”wireframing” (rautalankamalli), ”prototyping” (prototyyppi) ja ”usability testing” (käytettävyystestaus). Työvaiheita voi lisätä, poistaa tai niiden nimiä voi muokata vapaasti. Jokaiselle työvaiheelle voi myös määrittää statuksen, jotka oletusarvoisesti ovat ”in progress” (työn alla), ”waiting for review” (odottaa arviointia), ”accepted” (hyväksytty) ja ”paused” (pysäytetty). Projektin sisäisessä näkymässä käyttäjä näkee viimeisen merkityn työvaiheen ja sen statuksen.



Kuva 9. UXPin projektinäkymän hierarkia, työvaiheet ja työvaiheiden tilat.

Näen järkevän projektienhallintajärjestelmän olennaisena osana UXPiniä, koska UXPin ei tarjoa mahdollisuutta tallentaa projekteja paikallisille kovalevyille, eikä siten anna käyttäjälleen mahdollisuutta luoda esimerkiksi omia kansiorakenteita. Kovalevyille tallentaminen onkin ryhmätyöskentelyn kannalta hankalampaa, kun projektien osaset ovat hajautettuna useammalle tietokoneelle, eivätkä näin ole kaikkien projektiin osallistujien saavutettavissa.

3.2.2 UXPinin suunnittelujärjestelmät

Yksi UXPinin valttikorteista on ehdottomasti sen laajat suunnittelujärjestelmät (engl. design system). Suunnittelujärjestelmän idea on rakentaa kirjastoa komponenteista, joita suunnittelussa käytetään. Suunnittelujärjestelmiä voi olla useita ja yhdessä suunnittelujärjestelmässä on seuraavat luokat: colors (värit), typography (typografia), assets (resurssit) ja UI-patterns (käyttöliittymäelementit). Color- ja typography -luokat ovat aika itseselitteisiä, eli värejä ja typografiaa. Assets-luokkaan voi tallentaa kuvatiedostoja, ja UI patterns -luokkaan voi lisätä UXPinissä luotuja käyttöliittymäelementtejä, esimerkiksi valikkorakenteita tai nappeja. Jokainen näistä luokista voi pitää sisällään useamman alaluokan, esimerkiksi värit-luokassa voisi olla luokat ensisijaisille ja toissijaisille väripaleteille.

Suunnittelujärjestelmän luominen on erinomainen tapa edistää johdonmukaista suunnittelun periaatetta. Johdonmukaisen suunnittelun periaate tarkoittaa, että samankaltaiset asiat esitetään käyttäjille samalla tavalla tilanteesta riippumatta. Tämä voi olla haastava, kun saman suunnitelman parissa työskentelee useampi eri henkilö, joilla saattaa olla erilaiset näkemykset esimerkiksi elementtien graafisesta ilmeestä tai tiedon esittämistavoista. Suunnittelujärjestelmän laatiminen projektin alkuvaiheessa tuo kaikille suunnittelijoille samat elementit, joista lähteä rakentamaan suurempaa kokonaisuutta.

3.3 Piirtoalue ja mukautuvat versiot

UXPinin piirtoalueelle voi määrittää useita eri katkopisteitä (engl. breakpoints), jotka mahdollistavat saman prototyypin käytettävyyden useammalla eri näyttökoolla. UXPinissä näitä katkopisteitä kutsutaan mukautuviksi versioiksi

(engl. adaptive versions) Prototyypissä voi olla esimerkiksi omat katkopisteensä älypuhelimen ja vaikka full-hd näyttöjen mitoille. Katkopisteen ideana on, että jos näyttökoko on pienempi kuin nykyinen piirtoalue, siirrytään pienempään piirtoalueeseen, joka on lähimpänä laitteen todellista kokoa.

Esimerkkinä jos käynnistän prototyypin, jonka isoin näyttökoko on full-hd (1920x1080px), näen full-hd piirtoalueen, mutta jos prototyyppiä ajetaan pienemmällä näytöllä tai jos ikkunaa pienentää yhtään, vaihtaa prototyyppi pienenpään piirtoalueeseen, olipa se sitten tablet-laitteen näyttökoko tai älypuhelimen näyttökoko.

Uusi mukautuva versio luodaan piirtoalueen ominaisuudet-paneelista ” Add new adaptive version” -painikkeesta, joka nähtävissä on suunnittelunäkymän oikeassa reunassa silloin kun mikään piirtoalueen elementti ei ole valittuna. Se tekee kopion edellisen näyttökoon piirtoalueen sisällöstä, mutta piirtoalueen koko on erilainen ja sen seurauksena elementit täytyy järjestellä uudestaan. On tyypillistä, että joitain elementtejä piilotetaan kokonaan pienemmiltä näytöiltä, koska niiden jättäminen saattaisi haitata käyttäjäkokemusta.

3.4 Työkalut

UXPin ei erottele työkaluja eri kategorioihin, mutta niiden toiminnallisuuksien kuvailu vaatii mielestäni jaottelua monipuolisuuden vuoksi. Näkisin UXPinissä kolmen eri luokan työkaluja, joita voisi kuvata seuraavilla luokilla: *suunnittelutyökalut, sisältötyökalut* ja *saavutettavuustyökalut.* Suunnitelutyökaluille yhteistä on, että niillä luodaan elementtejä ”tyhjästä” piirtämällä erilaisia muotoja ja yhdistelemällä niitä. Sisältötyökalujen avulla haetaan sisältöjä UXPinin kirjastoista, kuvapankeista tai sitä generoidaan automaattisesti tiettyjen ennalta määriteltyjen sääntöjen mukaan. ja

Saavutettavuustyökalujen avulla voidaan arvioida käyttäjäkokemuksen laatua saavutettavuuden mittapuulla.

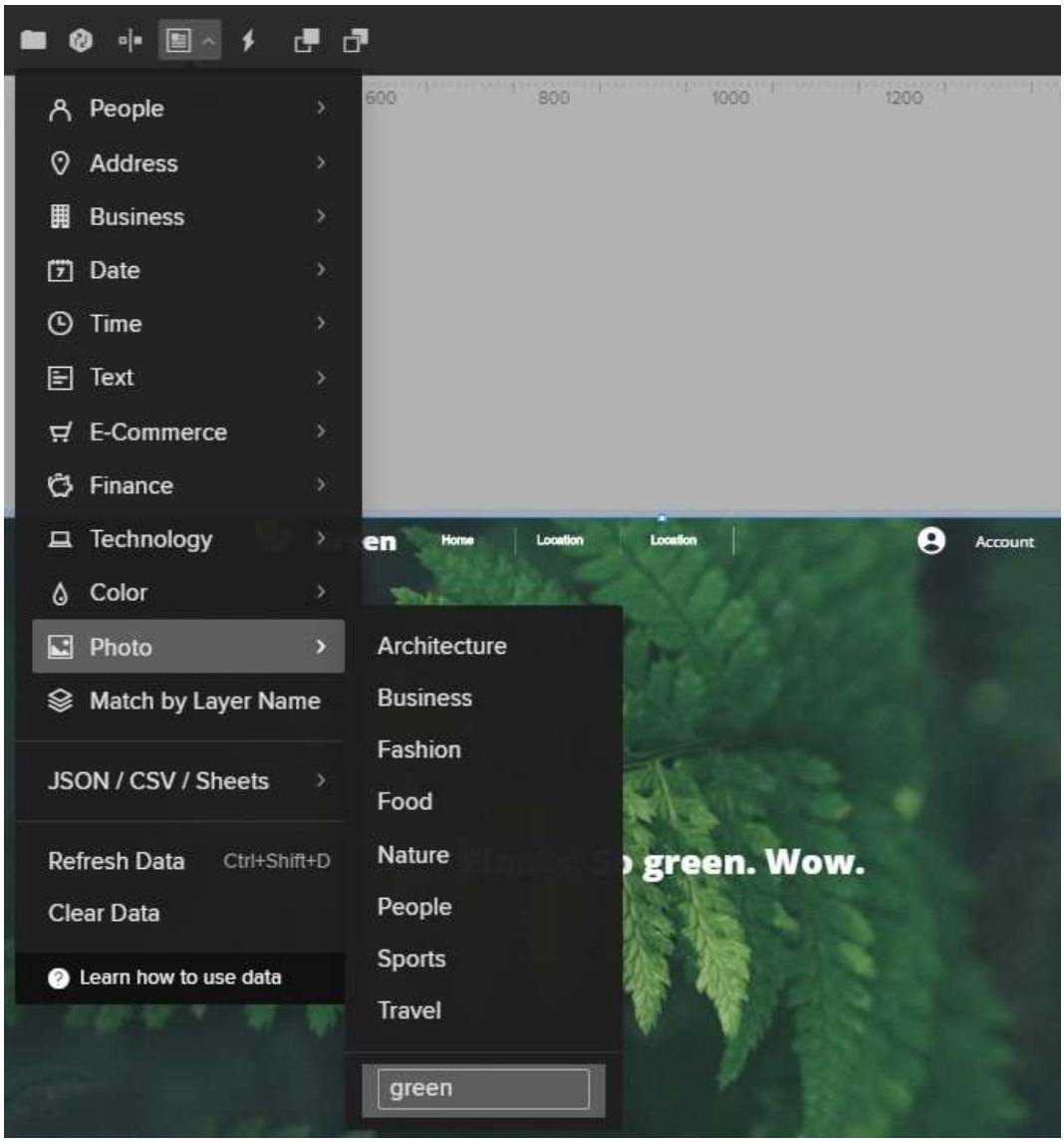
Suunnittelutyökaluihin kuuluvat perinteiset *suorakulmiotyökalu* (engl. rectangle), *ovaalityökalu* (engl. oval), *viivatyökalu* (engl. line), *polkutyökalu* (engl. path), *tekstityökalu* (engl. text), jolla voi luoda muokattavan tekstielementin ja *hotspot-työkalu*, jolla voi piirtää alueen, johon voi lisätä interaktioita. Hotspotit ovat elementtejä, jotka eivät näy prototyypissä, mutta voivat silti olla interaktiivisia alueita.

UXPin ei ole piirto- tai kuvanmuokkausohjelmisto, vaikka sillä niitä tukevia työkaluja onkin, vaan sen pääpaino on rautalankamalleissa ja prototyyppien luomisessa. Koska graafiseen suunnitteluun soveltuvien työkalujen kirjo UXPinissä ei ole kovin monipuolinen, UXPin tarvitsee rinnalleen mahdollisesti yhden tai useamman ohjelmiston laadukkaan grafiikan tuottamiseksi.

Sisältötyökaluihin kuuluvat *kuvatyökalu* (engl. image), jolla voi lisätä kuvan tietokoneelta, *ikonityökalu* (engl. icon), jolla voi lisätä ikonin UXPinin valmiista ikonikirjastosta sekä *hae kaikista resursseista*-työkalu *(*engl. search all assets), joka nimensä mukaisesti etsii hakusanan perusteella kaikista luoduista suunnittelujärjestelmistä sekä UXPinin oletuskirjastoista.

Yläpalkissa on myös *Täytä datalla* -työkalu (engl. fill with data). Se on voimakas sisältötyökalu, jolla voi täyttää objekteja, kuten suorakulmioita, ovaaleja tai tekstielementtejä, valitsemallaan datalla. UXPin generoi tiettyjä sisältöjä automaattisesti ja toisia sisältöjä se hakee ulkopuolisista lähteistä. Fill with data -toiminnon avulla voi täyttää elementtejä mm. kuvilla, nimillä, nimimerkeillä, sähköposteilla, päivämäärillä ja kellonajoilla. Kuvassa (Kuva 9.) näkyvä saniaiskasvi on lisätty fill with data -toiminnallisuuden ”photo” valinnalla, joka hakee kuvia Unsplashista joko kategoriakohtaisesti tai hakusanalla. Unsplash on

kuvapankki, josta voi ladata kuvia ilmaiseksi sekä omaan että kaupalliseen käyttöön (Unsplash 2019). Esimerkin kuvassa näkyvä saniainen on haettu hakusanalla ”green”.



Kuva 10. Fill with data -toiminnallisuuden valikkorakenne.

UXPinissä on kaksi **saavutettavuustyökalua** (engl. accessibility), *kontrastitaso* ja *värisokeustila.* Kontrastitaso korostaa tekstit, joilla on liian pieni kontrastiero verrattuna tekstin taustaan. Värisokeustilassa käyttäjä voi valita useammasta eri

tilasta, joilla voi simuloida yleisimmät värisokeuden muodot. Värisokeustilassa käyttöliittymän värit muuttuvat siten, että ne matkivat värisokean näkemää värispektriä.

3.5 Elementtien tilat - states

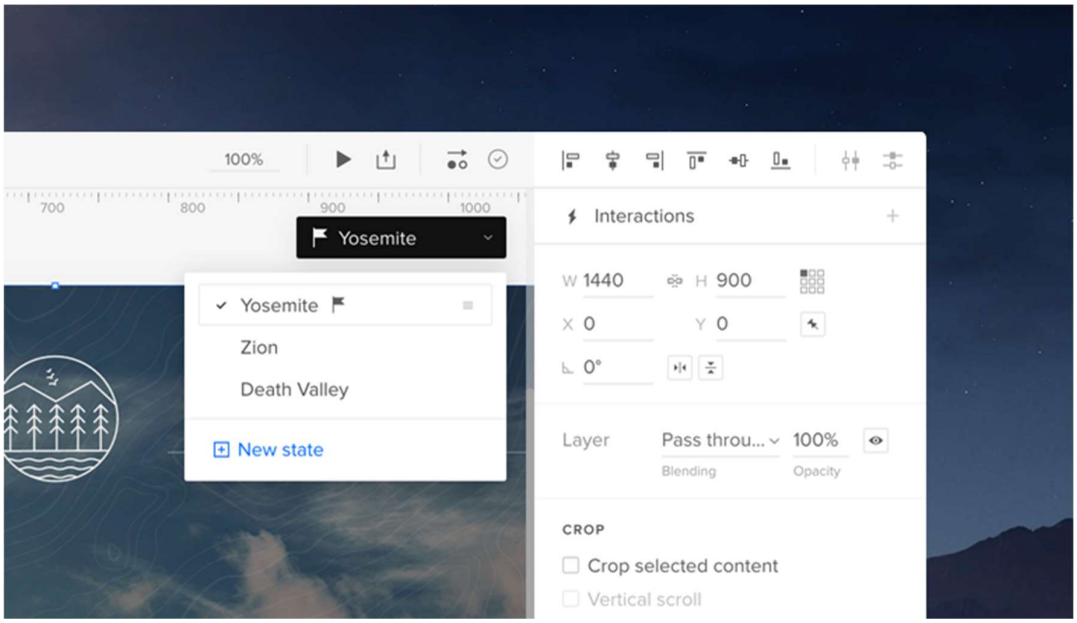
UXPinissä elementillä voi olla useampi eri tila (engl. state), joilla kaikilla on omat ominaisuutensa ja interaktionsa. Tyypillisesti tiloja käytetään esimerkiksi napeissa esittämään vaikkapa oletustilaa, tilaa kun hiiri on napin päällä ja tilaa kun nappia on klikattu. Elementtien tiloja voi kuitenkin hyödyntää myös monimutkaisemmissa komponentteja, kuten pudotusvalikkoja, tai erilaisia navigaatio-komponentteja. (UXPin 2018a.)

Kuva 11. Esimerkki log in -napin eri tiloista. Vasemmalta oikealle luettuna: Base-state, State 1 ja State 2. (UXPin 2018a.)

Jokaisella tiloja käyttävällä elementillä on perustilansa (engl. base-state). Perustila toimii tietynlaisena pohjana muille tiloille ja muut tilat perivät siltä joitain ominaisuuksia, kuten sisällön ja muodon. Perustila on merkitty lippuikonilla tilavalikkoon. Aina, kun jotain perustilan ominaisuutta päivitetään, muutokset tapahtuvat myös muissa tiloissa samoissa kohdin, ellei ko. ominaisuuksia on muokattu ennen perustilan muutosta. Jos jokin menee pieleen, voi tilan palauttaa

aina vastaamaan elementin perustilaa tilalistan ”reset all changes to base state” -napista. Jos perustilaan lisää interaktion, se periytyy myös muihin tiloihin, paitsi jos interaktion lisää johonkin muuhun kuin perustilaan, se on käytössä ainoastaan siinä tilassa. (UXPin 2018a.)

Jos kyseessä on useista käyttöliittymäelementeistä koostuva komponentti, jolla on tiloja, kaikki komponentin sisältämät elementit ovat myös kaikissa tiloissa. Jos elementin poistaa komponentista, se poistuu myös kaikista komponentin tiloista. Jos kuitenkin haluaa, että jokin elementti ei näy tietyssä tilassa, sen voi piilottaa tasohierarkian puolelta.



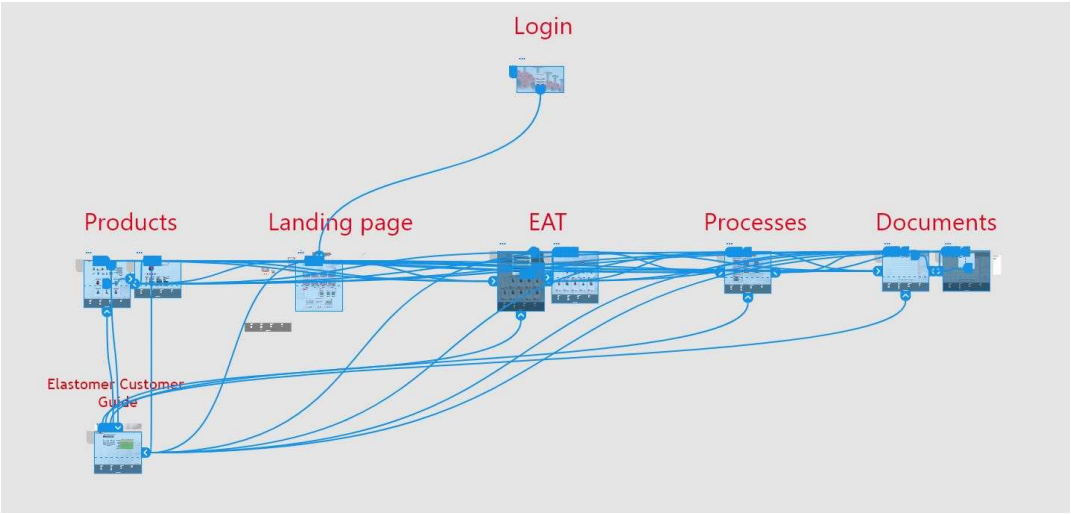
Kuva 12. Esimerkki state-listasta. Esimerkissä on kolme tilaa, joista perustila, eli base-state, on merkitty lippuikonilla. (UXPin 2018a.)

3.6 Interaktiot

3.6.1 Interaktioiden laukaisijat, eli triggerit

Muista prototyyppien luomiseen sovelutuvista suunnitteluohjelmistoista, kuten Adobe Experiencestä, Sketchistä, Figmasta, Invisionista ym. UXPin eroaa prototyyppien luomisessa siten, että siinä ei linkitetä ”artboardeja”, eli piirtoalueita, keskenään interaktioiden demonstroimiseksi. UXPinissä elementit voivat olla vuorovaikutuksessa keskenään ja interaktioille voi luoda omat lausekkeensa ja ehtonsa, joka tuo prototyyppiä lähemmäs lopullisen tuotteen toimintaa. (UXPin 2017.)

Esimerkiksi Adobe XD -suunnitteluohjelmistossa prototyypit luodaan siten, että kunkin elementin interaktio vaihtaa piirtoalueen aina toiseen piirtoalueeseen (engl. artboard), eli toisin sanoen kuva vaihtuu toiseen kuvaan. Jos haluaisin, että prototyypissä nappia painamalla piirtoalueelle piirretty ympyrä vaihtaa väriä, minun olisi luotava uusi piirtoalue, joka olisi väriä lukuun ottamatta identtinen kopio toisesta. Sitten olisi luotava linkki napista kyseiseen piirtoalueeseen. Jos interaktioita on paljon, piirtoalueita tulee nopeasti todella monta, ja prototyypin toimintalogiikan jäsentely käy pian haastavaksi. (UXPin 2017.)



Kuva 13. Adobe XD:n käyttöliittymä prototyypitys tilassa. Kaikki piirtoalueet on valittuna ja siniset viivat kuvaavat interaktioiden yhteyksiä artboardista toiseen.

UXPinissä ei tarvitse luoda uusia piirtoalueita, vaan monimutkaisetkin interaktiot voi toteuttaa yhdellä ja samalla piirtoalueella. Edellä kuvatun ympyrän väriä muuttavan napin interaktion voisi luoda väritoimintoa (engl. action→color) käyttämällä. Logiikkana olisi tällöin ”tämän napin toiminto muuttaa tämän ympyrän väriä” eikä ”tämän napin toiminto siirtää sinut näkymään, jossa tämä ympyrä on eri värinen” kuten Adobe XD:n ja monien muiden suunnitteluohjelmistojen kohdalla. Tämä ei ole kuitenkaan ainoa tapa, jolla kyseinen nappi voitaisiin UXPinissä toteuttaa. Vastaavan voisi tehdä myös määrittelemällä ympyräelementille kaksi tilaa, joissa on eri värit ja nappi vaihtaisi ympyrän tilan toiseen, jolloin väri muuttuisi. (UXPin 2017.)

Interaktion perusrakenteeseen kuuluu laukaisija (engl. trigger), joka käynnistää interaktion, ehto (engl. condition), jolla tarkastetaan täyttyvätkö toiminnon suorittamiselle asetetut ehdot ja toiminto (eng. action), joka suoritetaan. Elementeille tai piirtoalueelle itselleen voi laittaa useampia interaktioita, joilla on omat laukaisijansa, ehtonsa ja toimintonsa. (UXPin 2017.)

Interaktioiden triggerit voidaan jakaa kahteen luokkaan: **elementti-triggerit,** jotka käynnistävät toiminnon, kun käyttäjä on vuorovaikutuksessa elementin kanssa, ja **piirtoalue-triggerit**, jotka käynnistävät toiminnon kun piirtoalueen tila muuttuu. Ohessa on taulukko, johon on kirjattu kaikki triggerikategoriat, triggerit ja niiden kuvaukset. (UXPin 2017.)

Taulukko 1. UXPin interaktion laukaisijat.

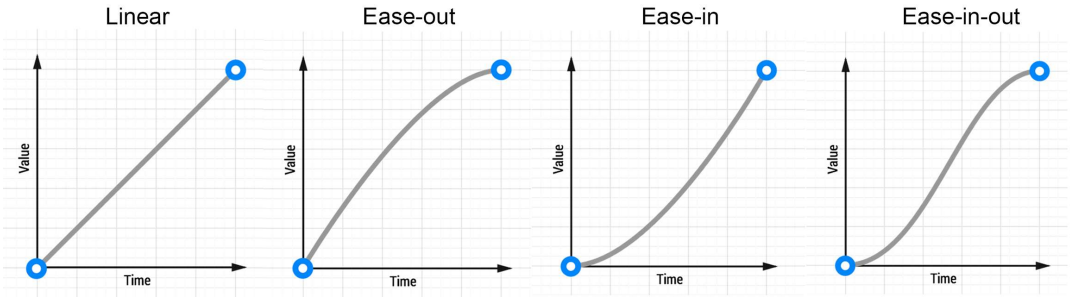
Kategoria	**Laukaisija**	**Kuvaus**
Click or tap (klikkaus tai sormipainallus)	Click or tap (klikkaus tai painallus)	Toiminto käynnistyy, kun interaktiivista elementtiä klikataan hiiren vasemmalla painikkeella tai painetaan sormella, jos kyseessä on kosketusnäyttö.
	Douple-click/tap (tuplaklikkaus/painallus)	Toiminto käynnistyy nopealla tuplaklikkauksella tai tuplapainalluksella, jos kyseessä on kosketusnäyttö.
	Right-click (klikkaus hiiren oikealla painikkeella)	Toiminto käynnistyy klikkaamalla hiiren oikealla painikkeella.
Mouse (hiiren liikkeet)	Hover (kursori on interaktiivisen elementin päällä)	Toimintoa suoritetaan kun hiiri on interaktiivisella alueella.
	Mouse in (kursori osuu interaktiiviseen elementtiin)	Toiminto käynnistyy sillä hetkellä, kun kursori osuu interaktiiviselle alueelle.
	Mouse out (kursori poistuu interaktiivisen elementin alueelta	Toiminto käynnistyy sillä hetkellä, kun kursori poistuu interaktiiviselta alueelta.
Gestures (eleet)	Swipe left (pyyhkäise vasemmalle)	Elementin pyyhkäisy vasemmalle käynnistää toiminnon.
	Swipe right (pyyhkäise oikealle)	Elementin pyyhkäisy oikealle käynnistää toiminnon.
	Swipe up (pyyhkäise ylös)	Elementin pyyhkäisy ylöspäin käynnistää toiminnon.
	Swipe down (pyyhkäise alas)	Elementin pyyhkäisy alaspäin käynnistää toiminnon.
Canvas (piirtoalue)	Key press (näppäimistösyöte)	Toiminto käynnistyy määriteltyä näppäimistön nappia painamalla.
	Window is scrolled to (näkymän vierittäminen)	Toiminto käynnistyy, kun näkymä on vieritetty määritellyn elementin kohdalle.
	Page is loaded (sivu latautuminen)	Toiminto käynnistyy heti, kun sivu on latautunut.

3.6.2 Toiminnot ja animaatio

Toiminto on se tapahtuma, jonka laukaisija käynnistää interaktiossa. Ilman toimintoja, prototyypissä ei ole vuorovaikutteisuutta. UXPinillä on monipuolinen toimintokirjasto, jonka jokaisella toiminnolla on omat asetuksensa. Lähes kaikilla toiminnoilla muuttujatoimintoja lukuun ottamatta on asetukset animaatiolle. Animaatioasetuksiin sisältyvät animaation keventäminen, kesto, viive ja toisto.

Paul Lewiksen mukaan luonnossa mikään ei liiku tasaisesti yhdestä pisteestä toiseen, vaan yleensä liike alkaa hitaasti ja kiihtyy tai hidastuu loppua kohden. Koska suoraviivaista liikettä ei luonnossa esiinny, aivomme ovat virittyneet siten, että ne odottavat kiihtyvyyttä tai hidastuvuutta ja tätä pitäisi hyödyntää animaatiossa. (Lewis 2019.)

Käyttöliittymäsuunnittelussa animaatioiden liikettä voidaan kuvata *easing*-termillä (Kuva 13). *Linear* tarkoittaa suoraviivaista liikettä. *Ease-in* tarkoittaa liikettä, joka lähtee hitaasti käyntiin ja kiihtyy loppua kohden. *Ease-out* tarkoittaa liikettä, joka hidastuu lopussa. *Ease-in-out* tarkoittaa liikettä, joka alkaa hitaasti, kiihtyy ja lopussa hidastuu. (Lewis 2019.)



Kuva 14. Easing esitettynä graafeina. Vaakataso kuvaa aikaa ja pystytaso liikkeen etenemistä. (Lewis 2019.)

UXPinin toimintokategorioita ovat linkit, animaatio, sisältö, lomake-elementit, järjestely ja elementtien tilat. Alla on excel-taulukko (Taulukko 2.), johon on listattu kaikki UXPin interaktioiden perustoiminnot. Taulukossa on edellä mainitut kategoriat ja kuvaus kustakin toiminnosta.

Erityisesti API-pyyntö, eli pyyntö ohjelmistorajapinnasta (engl. API-request) on mielenkiintoinen, koska se mahdollistaa kommunikoimisen prototyypin ja älykkäiden laitteiden välillä. Käytännössä API-pyyntö toimintona mahdollistaa IoT järjestelmän prototyypittämisen. UXPinin verkkosivuilla on lyhyt esimerkki API-pyyntöjen käytöstä, jossa prototyypin avulla vaihdetaan älyvalon väriä. UXPin ei ole vielä julkaisut yksityiskohtaisempia ohjeistuksia tai tutoriaaleja API-pyyntöjen käyttämiseen, koska ominaisuus on suhteellisen uusi. Jään innolla odottamaan UXPiniltä jonkinlaista tutoriaalia ko. toiminnon hyödyntämiseen. (UXPin 2019d.)

Taulukko 2. UXPinin interaktioiden toiminnot.

Kategoria	Toiminto	Kuvaus
Links (linkit)	Go to page (mene sivulle)	Interaktio avaa sivun UXPin prototyypistä.
	Go to URL (mene URL -osoitteeseen)	Interaktio avaa URL-osoitteen selaimessa.
	Go back (palaa takaisin)	Interaktio avaa edellisen ladatun sivun.
	API request (API-pyyntö)	Pyyntö ohjelmistorajapinnasta.
Animate (animaatio)	Move to (määritelty sijainti)	Siirtää elementin määriteltyyn pisteeseen x-y koordinaatistossa.
	Move by (määritelty siirtymämatka)	Siirtää elementtiä määritellyn matkan x ja/tai y koordinaatistossa.
	Opacity (läpinäkyvyys)	Määrittelee elementin läpinäkyvyydelle arvon
	Size (koko)	Määrittelee elementin koon.
	Color (väri)	Määrittelee elementin värin.

	Rotate (kierrä) (by/to)	Asettaa elementin kallistuskulman määriteltyyn asteeseen (to). Muuttaa elementin kallistuskulmaa tietyn astemäärän verran (by).
	Advanced (edistynyt animaatio)	Animaatio, jolla voi vaikuttaa useisiin elementteihin vaiheittain.
Content (sisältö)	Set Content (aseta sisältö)	Lisää halutun elementin tai muuttujan sisällön kohde-elementtiin.
Form element (lomake-elementti)	Focus (keskitä)	Kohdistaa valinnan kohteena olevaan lomake-elenttiin.
	Enable (ota käyttöön)	Ottaa lomake-elementin käyttöön.
	Disable (poista käytöstä)	Poistaa lomake-elementin käytöstä.
	Check (lisää valinta)	Lisää valinnan valintalaatikko -elementtiin (checkbox).
	Uncheck (poista valinta)	Poistaa valinnan valintalaatikko-elementistä.
Arrange (järjestä)	Bring to front (tuo eteen)	Nostaa elementin tasohierarkian ylimmäiseksi.
	Send to back (vie taakse)	Pudottaa elementin tasohierarkian alimmaiseksi.
State (tila)	Next state (seuraava tila)	Vaihtaa elementin tilaksi seuraavan tilan.
	Previous state (edellinen tila)	Vaihtaa elementin tilaksi edellisen tilan.
	Set State (aseta tila)	Asettaa elementille määritellyn tilan.

3.6.3 Lausekkeet

Lausekkeiden avulla suunnittelija voi luoda monimutkaisempia toiminnallisuuksia omiin prototyyppeihinsa. Käytännössä lausekkeet sallivat suunnittelijan kirjoittaa kevyttä JavaScript koodia elementteihin tai piirtoalueeseen, joka mahdollistaa laskennallisten ominaisuuksien hyödyntämisen ja sitä kautta huomattavasti kehittyneemmän prototyypin tuottamisen. (UXPin 2019e.)

Lausekkeissa voi käyttää mm. aritmeettisia ja trigonometrisia funktioita, jotka on lainattu JavaScriptistä. Lausekkeilla voi käsitellä muuttujia, jotka voivat olla esim. merkkijonoja ja numeroita. Lausekkeilla voi validoida käyttäjien syöttämiä tietoja, esimerkiksi tarkistaakseen, täyttääkö salasana tietyt kriteerit. (UXPin 2019e.)

UXPinissä lausekkeet noudattavat tiettyä syntaksia, eli lauseoppia. Ohessa esimerkkejä lausekkeiden mahdollisista tietotyypeistä ja niiden merkintätavoista. (UXPin 2019e.)

- numerot – kokonaisluvut kuten **5, 10, 15;**
- merkkijonot – merkkijonot alkavat ja loppuvat lainausmerkkeihin esim. **”teksti”;**
- muuttujat – muuttujat, joita on lisätty prototyyppiin esim. **$muuttujan_nimi;**
- elementin sisältö – määriteltyn elementin sisältö voidaan määritellä esim. **‘Box’**;
- funktiot – mikä tahansa tuetuista funktioista esim. **length(merkkijono);**
- totuusarvomuuttujat – **true** tai **false** -arvot

3.6.4 Muuttujat

Muuttujien avulla voidaan tallentaa prototyyppiin käyttäjän syötteitä ja hyödyntää niitä myöhemmin prototyypissä. Muuttujia voi hyödyntää esimerkiksi dynaamisissa tekstielementeissä, käyttäjäkohtaisissa asetuksissa tai tiedonkeruussa. Muuttujiin tallennetut arvot ovat globaaleja, eli ne toimivat kaikilla prototyypin sivuilla. (UXPin 2018b.)

UXPinissä muuttujien nimeämisillä on tiettyjä rajoitteita. Muuttujan nimen tulisi alkaa kirjaimella, koostua ainoastaan aakkosista tai numeroista, ei saa olla välilyöntejä kokonaismerkkimäärä on 25. UXPinin sivuilla listataan myös seuraavat lausekkeita varten varatut fraasit, joita ei voi käyttää nimeämisessä*:*

do, if, in, for, let, new, try, var, case, else, enum, eval, null, this, true, void, with, await, break, catch, class, const, false, super, throw, while, yield, delete, export, import, public, return, static, switch,typeof, default, extends, finally, package, private, continue, debugger, function, arguments, interface, protected, implements ja *instanceof*. Jos muuttujalle annetaan nimeksi edeltävässä listassa ollut fraasi, UXPin lisää automaattisesti muuttujan nimen perään alaviivan ja numeron. (UXPin 2018b.)

3.6.5 Interaktioiden ehdot - conditions

Interktion ehdoilla tarkoitetaan sääntöjä, joiden avulla määritetään, pitääkö interaktion toiminto suorittaa vai ei. Jokaisella ehdolla on ”if-then” -rakenne. Jos if-ehto täyttyy, interktion sisältämä toiminto suoritetaan. Jos if-ehto ei täyty, siirrytään suorittamaan then-ehdon sisältämä toiminto, jos sellaista on. (UXPin 2017.)

Ehtolauseessa verrataan kahta tekijää keskenään jollain tietyllä perusteella. Ehtona voisi olla esimerkiksi ”muuttuja1:n arvo on oltava suurempi tai yhtä suuri kuin muuttuja2: arvo. ” Alla taulukko, jossa esimerkkejä interaktioiden ehdoista Taulukko 3.).

Ehtolauseita voi hyödyntää prototyypeissä, jossa käyttäjän odotetaan syöttävän tietoja jossain tietyssä muodossa. Esimerkkitapauksena vaikka käyttäjätunnuksen ja salasanan syöttäminen. Kirjautumiselle voidaan asettaa *regex* vertailuperiaate, eli verrataan kielisääntöön, jonka mukaan käyttäjätunnukseksi annetun syötteen täytyy olla sähköpostimuotoinen. Salasanaa voidaan verrata muuttujaan, joka on aiemmin alustettu käyttäjän syötteellä tai alustettu muuten etukäteen.

Taulukko 3. Interaktioiden ehdot kuvauksineen.

Kategoria	**Vertailuperiaate**	**Kuvaus**
Arvojen vertailu	equals (on yhtä kuin)	Tarkastaa, onko arvo yhtä suuri kuin vertailtava arvo.
	doesn't equal (on erisuuri)	Tarkastaa onko arvo erisuuri kuin vertailtava arvo.
	is greater (on suurempi)	Tarkastaa, onko suurempi kuin vertailtava arvo.
	is greater or equal (on suurempi tai yhtä suuri)	Tarkastaa, onko arvo suurempi tai yhtä suuri kuin vertailtava arvo.
	is less (on pienempi)	Tarkastaa onko arvo pienempi kuin vertailtava arvo.
	is less or equal (on pienempi tai yhtä suuri)	Tarkastaa onko arvo pienempi tai yhtä suuri kuin vertailtava arvo.
Sisällön vertailu	contains (sisältää)	Tarkastaa sisältääkö esim. merkkijono jonkun tietyn merkin tai merkkiyhdistelmän.
	doesn't contain (ei sisällä)	Tarkastaa puuttuuko merkkijonosta joku tietty merkki tai merkkiyhdistelmä.
	starts with (alkaa)	Tarkastaa alkaako esim. merkkijono jollain määritellyllä merkkijonolla.
	ends with (loppuu)	Tarkastaa loppuuko esim. merkkjono johonkin tiettyyn merkkijonoon.
Kielisäännöt	matches regex (vastaa kielisääntöä)	Tarkastaa, vastaako vertailtava merkkijono määriteltyä kielisääntöä.
	doesn't match regex (ei vastaa kielisääntöä)	Tarkastaa, eroaako vertailtava merkkijono määritellystä kielisäännöstä.
Totuusarvot	is empty (on tyhjä)	Tarkastaa, onko muuttuja alustamaton.
	is not empty (ei ole tyhjä)	Tarkastaa, onko muuttuja alustettu.
	is true (on tosi)	Tarkastaa, onko totuusmuuttujan arvo tosi.
	is false (on epätosi)	Tarkastaa, onko totuusmuuttujan arvo epätosi.

3.6.6 Funktiot

UXPinillä on valmis kirjasto JavaScriptistä lainattuja funktioita, joita voi käyttää prototyyppien elävöittämisessä. Funktiot lisätään lausekkeisiin ja ne noudattavat aiemmin kuvattua syntaksia. Funktioilla voi käsitellä numero ja/tai merkkijono muuttujia ja suorittaa niiden avulla erilaisia laskutoimituksia. Yksinkertaisten plus, miinus, kerto- ja jakolaskujen lisäksi voi suorittaa trigonometrisia tai aritmeettisia laskukaavoja. Funktioiden avulla voi myös käsitellä päivämääriin, verkkoselaimeen ja prototyyppiin liittyviä tietoja.

Alle olen koonnut taulukon funktioista ja niiden kuvauksista. Laadin funktioille värijärjestelmän, jotta oikeantyyppisten funktioiden etsiminen kävisi nopeammin ja taulukkoa voisi käyttää eräänlaisena lunttilappuna.

Taulukko 4. UXPin tukemat funktiot (sivulla 44).

Väriselitteet funktioille

Numeroita käsittelevät funktiot
Merkkijonoja käsittelevät funktiot
Numeroita ja merkkijonoja käsittelevät funktiot
Selaimen tietoja käsittelevät funktiot
Prototyypin tietoja käsittelevät funktiot
Aikaa ja päivämääriä käsittelevät funktiot

Funktio	**Kuvaus**
abs (number)	Palauttaa numeron absoluuttisen arvon.
acos (number)	Palauttaa numeron kaarikosinin.
asin (number)	Palauttaa numeron arkussinin.
atan (number)	Palauttaa numeron arkustangentin.
avg (number, number,…)	Palauttaa valittujen numeroiden keskiarvon.
canvasHeight()	Palauttaa piirtoalueen korkeusmitan.
canvasWidth()	Palauttaa piirtoalueen leveysmitan.
capitalize (string)	Lisää ison alkukirjaimen tekstiin.

ceil (numer)	Palauttaa pienimmän integer-muuttujan arvon, joka on suurempi tai yhtäsuuri kuin annettu numero.
concat (string, string,…)	Yhdistää valitut merkkijonot ja palauttaa uuden merkkijonon.
contains (string, searchString)	Tarkistaa, sisältääkö merkkijono haettua merkkijonoa.
cos (number)	Palauttaa numeron kosinin.
date (format?)	Palauttaa päivämäärän määritellyssä formaatissa.
dateAdd(date, amount, unit?, format?)	Lisää annetun aikamäärän määriteltyyn yksikköön (päivät, viikot, kuukaudet, vuodet) päivämäärässä.
dateDiff (firstDate, secondDate, unit?)	Palauttaa ajankohtien eron määritellyssä yksikössä (päivät, viikot, kuukaudet, vuodet)
dateIsAfter (firstDate, secondDate, precision?)	Tarkistaa, onko päivämäärä toisen päivämäärän jälkeen annetun yksikön perusteella (päivät, viikot, kuukaudet, vuodet).
dateIsBefore (firstDate, secondDate, precision?)	Tarkistaa, onko päivämäärä toista päivämäärää ennen annetun yksikön perusteella (päivät, viikot, kuukaudet, vuodet).
dateIsBetween (date, startDate, endDate, precision?)	Tarkistaa, onko päivämäärä määritellyllä aikavälillä annetun yksikön perusteella (päivät, viikot, kuukaudet, vuodet).
dateSubtract (date, amount, unit?, format?)	Vähentää annetun määrän ajasta yksikön mukaan (päivät, viikot, kuukaudet, vuodet).
deviceOS()	Palauttaa käyttöjärjestelmän tyypin.
endsWith (string, searchString)	Tarkastaa, loppuuko merkkijono annetulla merkkijonolla. Palauttaa totuusarvomuuttujan (boolean: true/false).
execute (string)	Suorittaa merkkijonon lausekkeen.
floor (number)	Palauttaa suurimman integer-muutujan, joka on pienempi tai yhtäsuuri kuin annettu numero.
getActiveState(string)	Palauttaa elementin aktiivisen tilan.
format (string|number, format)	Palauttaa merkkijonon tai numeron määritellyssä formaatissa.
indexOf (string, seachString)	Palauttaa haku merkkijonon sijainniksi oletus merkkijonon.
isBetween (number, start, end)	Tarkastaa, onko numero määritellyllä lukualueella.

isNotBetween(number, start, end)	Tarkastaa, onko numero määritellyllä lukualueella.
left (string, number)	Palauttaa tietyn määrän merkkejä tekstistä, lukien vasemmalta oikealle.
length (string)	Palauttaa merkkijonon merkkien määrän numerona.
log (number)	Palauttaa numeron luonnollisen logaritmin.
lowerCase (string)	Palauttaa merkkijonon pienaakkosina.
max (number, number, …)	Palauttaa maksiminumeron annetuista numeroista.
median (number, number, …)	Palauttaa mediaaninumeron annetuista numeroista.
min (number, number, …)	Palauttaa miniminumeron annetuista numeroista.
now (format?)	Palauttaa ajankohtaisen päivämäärän määritellyssä formaatissa.
pageName()	Palauttaa avoinna olevan sivun nimen.
pageNumber()	Palauttaa sivun järjestysnumeron sivustokartassa.
pagePath()	Palauttaa polun avoinna olevalle sivulle.
pow (base, exponent)	Palauttaa määritetyn eksponentin tehon.
prototypeName()	Palauttaa prototyypin nimen.
random (min, max)	Palauttaa satunnaisen luvun määritellyltä lukualueelta.
regex (string, pattern)	Tarkastaa, täyttääkö syötetty merkkijono lausekkeen ehdot.
replace (string, pattern, replacement)	Vaihtaa merkkijonon toiseen.
right (string, number)	Palauttaa määrittellyn määrän merkkejä merkkijonosta, lukien oikealta vasemmalle.
round (number, precision?)	Pyöristää luvun lähimpään kokonaislukuun.
sin (number)	Palauttaa numeron sinin.
sqrt (number)	Palauttaa numeron neliöjuuren.
startsWith(string, searchString)	Tarkastaa, alkaako merkkijono määritellyillä merkeillä. Palauttaa tootusarvon. (boolean: true/false)
substring(string, start, end)	Palauttaa osan merkkijonosta määriteltyjen aloitus ja lopetuspisteiden välillä tai merkkijonon loppuun asti.
tan (number)	Palauttaa numeron tangentin.
time (format)	Palauttaa sen hetkisen ajan määritellyssä formaatissa.
toNumber (value)	Palauttaa elementin numerona.

toString (value)	Palauttaa elementin merkkijonona.
trim (string)	Poistaa välilyönnit merkkijonon alusta ja lopusta.
trunc (number)	Palauttaa luvun kokonaisluvun poistamalla mahdolliset desimaalit.
upperCase (string)	Palauttaa merkkijonon suuraakkosina.
windowHeight ()	Palauttaa selaimen näkymisportin korkeuden sekä vierityspalkin korkeuden, jos sellaista on.
windowScrollX()	Palauttaa sen x-akselin arvon, johon sivu on sillä hetkellä vieritetty.
windowScrollY()	Palauttaa sen y-akselin arvon, johon sivu on sillä hetkellä vieritetty.
windowWidth()	Palauttaa selaimen näkymisportin leveyden, sekä vierityspalkin leveyden, jos sellaista on.

3.7 Design handoff

Design handoff tarkoittaa sitä prosessia, kun suunnittelija luovuttaa suunnitelman eteenpäin kehittäjälle (Babarczi 2018). UXPin:llä voi tehokkaasti jakaa prototyyppejä kehittäjätiimin jäsenille tai asiakkaille. Prototyypin yhteyteen UXPin tarjoaa erilaisia tapoja selventää suunnittelijan ajatuksenjuoksua kehittäjälle.

Prototyypin yläpaneelissa on pudotusvalikon ”Actions” (toiminnot), ”Sitemap” (sivukartta) ja ”breakpoints” (katkaisupisteet). Actions valikossa on seuraavat vaihtoehdot: muokkaa prototyyppiä (engl. edit prototype), mene projektiin (engl. go to project), aloita live-esitys (engl.start live presentation) ja kirjaudu ulos (engl. log out). Sivukartta-pudotusvalikon alla on listattu kaikki prototyypin sisältämät sivut ja sen kautta voi siirtyä sivulta toiselle. Live-esityksen aloittaminen avaa prototyypin verkkoselaimessa ja antaa käyttäjälle linkin, jonka kautta kuka tahansa voi päästä katsomaan prototyypin ajamista. Katsojat eivät voi itse ajaa prototyyppiä live-esityksen aikana, mutta he voivat tarkkailla esittelijän toimia ja kommentoida esimerkiksi mikrofonin avulla. Lisäksi yläpaneelissa on neljä

välilehteä, simulaatio (engl. simulate), kommentti (engl. comment), tekniset tiedot (engl. spec) ja dokumentaatio (engl. documentation).

Kuva 15. UXpin prototyypin välilehdet. Kuvassa dokumentaatio-välilehti on aktiivisena.

Simulate-välilehdellä ajetaan prototyyppiä. **Comment**-välilehdellä käyttäjä voi lisätä mihin tahansa kohtaan pisteen, johon voi kirjoittaa kommentin. Kommentin lisääminen aloittaa keskustelun, jossa asiakas, suunnittelija ja kehittäjä voivat käsitellä esimerkiksi johonkin tiettyyn elementtiin liittyvää ongelmaa. Kaikki kommenttiketjuun lisätyt henkilöt saavat sähköpostiin ilmoituksen, aina kun kommenttiketju päivittyy. Kun ongelma on ratkaistu, suunnittelija voi merkitä kommenttiketjun tilaksi ”resolved” eli ratkaistu.

Spec-välilehti sisältää käyttöliittymän tekniset tiedot. Spec-tilassa voi tarkastella jokaisen elementin yksityiskohtaisia tietoja, joihin kuuluvat esimerkiksi objektien mitat ja sijainti (korkeus, leveys, x-y -koordinaatit), kirjasinperheet, väripaletti ja mahdolliset ikonit. Prototyypissä käytetyt ikonit voi myös ladata suoraan käyttöliittymästä joko alkuperäisessä tiedostomuodossaan tai .png -tiedostona, joka on vakio riippumatta alkuperäisestä tiedostomuodosta. UXPin tukee seuraavia kuvaformaatteja: .svg, .jpg, .png, ja .gif (myös animoidut gif-tiedostot). (Gremillion 2018.)

Document-välilehdellä suunnittelija voi lisätä elementti- tai sivukohtaisia selvennyksiä suunnitelmaan. Osion oikeanpuoleisessa paneelissa on tekstinkäsittelytyökalu, johon voi kirjoittaa muistiinpanoja tai selitteitä

suunnitelmalle. Jokaisen kappaleen voi *pinnata,* eli kiinnittää johonkin tiettyyn elementtiin. Tekstiä voi myös tyylitellä esimerkiksi lihavoinnilla tai kursiivilla. Tämän ominaisuuden tarkoitus on helpottaa kommunikointia suunnittelijoiden, ohjelmoijien ja asiakkaiden välillä. Dokumentaatiota voi toki myös hyödyntää omina muistiinpanoina. (UXPin 2017.)

Kuva 16. Esimerkki dokumentoinnista. Vasemmalla kaksi tekstielemnttiä ja oikealla niihin liitetyt muistiinpanot.

Vaikka prototyypistä saakin paljon suunnittelupuolen teknisiä tietoja irti, animaatioiden yksityiskohtaisia tietoja ohjelmoija ei voi prototyypistä nähdä**.** Näitä voisi kirjata ylös dokumentaatio-osiossa animaatiota sisältäviin elementteihin. Ohjelmoijan kannalta olennaisimpia tietoja animaatiosta ovat sen kesto millisekunneissa (engl. duration), viive (engl. delay), toisto (engl. loop) ja animaation keventämisen (engl. easing) eri metodit. (UXPin 2017.)

4 Confluence ja UXPin dokumentaatiopohja

Opinnäytteen yhtenä osa-alueena oli laatia UXPinia käsittelevä sivu toimeksiantajan Confluence-wikiin. Confluence on organisaatiowikiohjelmisto, jonka on tehnyt Atlassian. Confluencen lisäksi yksi Atlassianin tunnetuimmista tuotteista on myös Jira, joka on tehtävienhallintaan tarkoitettu verkkosovellus, joka on myös Process Geniuksen käytössä. (Wikipedia 2018.)

Ideana on, että Confluence-sivua päivitetään sitä mukaa, kun UXPiniin tulee uusia ominaisuuksia tai muutoksia. Dokumentaatiopohjaan sisältyvät seuraavat opinnäytteessä esitellyt asiat:

- Värikoodattu taulukko funktioille
- Taulukot triggereille, toiminnoille ja ehtojen vertailuperiaatteille
- Rajoitukset muuttujan nimeämisessä
- Esimerkit lausekkeen syntaksista

5 Pohdinta

Ennen tätä opinnäytetyötä minulla oli jo jonkin verran kokemusta ja hajua siitä, mitä UX suunnittelijan työhön kuuluu. Opintojen ja työharjoittelun kautta olin saanut ensimmäiset kosketukseni käyttöliittymäsuunnittelijan rooliin ja matkan varrella olin kokeillut useita tarkoitukseen sopivia ohjelmistoja. Opinnäytetyön aloitusvaiheessa olin käyttänyt UXPinia vain joitain kuukausia. Tämän opinnäytetyön kautta olen oppinut paljon käyttäjäkokemussuunnittelusta ja osaamiseni UXPinin käyttäjänä on kasvanut merkittävästi. Opinnäytetyöni myötä olen alkanut arvioimaan arkipäiväisten ympäristöjen, laitteiden ja ohjelmistojen käytettävyyttä kriittisemmällä silmällä ja olen löytänyt UXPinista ominaisuuksia, joita en aiemmin tiennyt olevan olemassa.

Opinnäytetyöprosessissa koin suurimmaksi haasteeksi aiheen sopivan rajaamisen. Käynnistysvaiheessa pallottelin usean eri aiheen välillä, jotka kaikki pyörivät UI/UX suunnittelun ja siihen liittyvien teknologioiden ympärillä. Lopulta päädyttyäni sopivaan aiheeseen, jouduin tarkentamaan rajausta monta kertaa. Haasteina olivat myös ajankäyttö ja tehokkaan kirjoitusrutiinin omaksuminen. Työssäkäyvänä nuorena miehenä voi toisinaan olla haastavaa löytää aikaa ja motivaatiota kirjoittaa akateemista tekstiä, etenkin kun sellaisen tuottamisesta ei ole aiempaa kokemusta tällaisessa mittakaavassa.

Työn pääaihe, eli UXPin, on monipuolinen, jatkuvan kehitystyön alla oleva ohjelmisto. Aihe on siksi ajankohtainen ja mielestäni tämän työn puitteissa sen ominaisuuksien kuvailu on riittävää. Tavoitteessani julkaista organisaatiowiki Confluencessa, tuli ongelmia, kun minulla ei ollut kaikkia julkaisuun tarvittavia käyttöoikeuksia. Tällä hetkellä olennaisimmat sisällöt, joita siihen halusin, on kuitenkin laadittu valmiiksi, joten en pidä julkaisun epäonnistumista kovin suurena ongelmana.

Herää kysymyksiä, onko Confluence ylipäätään oikea paikka dokumentaatiopohjalle, ja kannattaako sitä julkaista sellaisenaan lainkaan? UXPin nimittäin vastikään (14. toukokuuta 2019) julkaisi oman wikisivustonsa, *UXPin Docsin*, käyttöohjeille ja dokumentaatiolle, ja sitä varmasti tullaan päivittämään ahkerasti tulevaisuudessa (UXPin 2019c). Harmikseni, julkaisu tapahtui päivä seminaarin jälkeen, jossa esittelin lähes valmiin opinnäytetyöni. Vastaavaa keskitettyä ohjekokoelmaa UXPinillä ei aiemmin ollut ja en tiedä onko hirveästi mieltä lähteä ”kilpailemaan” sellaisen kanssa, ellei halua tehdä pelkkää käännöstyötä.

Työn osioissa, jotka käsittelevät käyttäjäkokemussuunnittelua, on useita teemoja, joista voisi kirjoittaa paljon ja monesta eri näkökulmasta. Teknologia kehittyy ja

sen myötä kehittyvät myös suunnittelumenetelmät ja niihin liittyvät käytännöt. Tämän takia erilaisista UX suunnittelun prosessimalleista, uusista trendeistä tai käytännöistä voisi varmaan kirjoittaa useamman opinnäytetyön melkein joka vuosi. Omasta mielestäni kiinnostavia, ja todennäköisesti myös Process Genius Oy:lle hyödyllisiä tutkimuksen aihealueita voisivat olla virtuaalitodellisuusalustojen sovellusten UX-suunnittelun tutkiminen tai niiden toteuttamiseen soveltuvien työkalujen vertailuarviointi. Virtuaalitodellisuutta ja lisättyä todellisuutta tullaan näkemään tulevaisuudessa varmasti enemmän myös teollisuuden saralla.

Lähdeluettelo

Anne. 2011. Käyttäjäpersoonat herättävät käyttäjät eloon. IT innovaatiopalvelumalli. 20.10.2011. https://blogs.aalto.fi/itainnovaatiopalvelu/2011/10/20/kayttajapersoonat-herattavat-kayttajat-eloon/. 09.01.2019.

Babarczi, K. 2018. UXstudio. Design handoff guide: What makes good designer-developer collaboration? https://uxstudioteam.com/ux-blog/design-handoff/. 11.04.2019.

Computer History Museum. 2019. Xerox Alto. https://www.computerhistory.org/revolution/input-output/14/347. 22.05.2019.

Computer Hope. 2017. CUI. https://www.computerhope.com/jargon/c/cui.htm. 25.04.2019.

Elisa. 2017. Sm4rtlab – Maailman ensimmäinen todellisuutta laajentava laboratiorio. https://hub.elisa.fi/sm4rtlab-maailman-ensimmainen-todellisuutta-laajentava-laboratorio/. 16.10.2018.

Gremillion, B. 2016. Importing Assets. https://www.uxpin.com/studio/user-guide/importing-assets/ 02.02.2019.

Hassenzahl, M. Tractinsky, N. 2006. User experience – a research agenda. Teoksessa Behaviour & Information Technology, Vol. 25, No. 2. Taylor & Francis Group. 91-95.

Immonen, J. 2003. Graafiset käyttöliittymät. http://cs.joensuu.fi/~jimmonen/gkl_moniste/gkl_v202.html. 18.11.2018.

Interaction Design Foundation. 2018. What is User interface (UI) Design?

https://www.interaction-design.org/literature/topics/ui-design. 18.11.2018.
Interaction Design Foundation. 2019. User Personas. https://www.interaction-design.org/literature/topics/user-personas. 20.02.2019.
International Organization for Standardization. 2018. ISO 9241-11:2018 Ergonomics of human-system interaction – Part 11: Usability: Definitions and Concepts. https://www.iso.org/standard/63500.html. 09.11.2018.
ITPRO. 2019. What is a graphical user interface? https://www.itpro.co.uk/operating- systems/30248/what-is-a-graphical-user-interface. 07.02.2019.
Koivunen, M. Nieminen, M. 1995. Ohjelmiston käytettävyys. Teoksessa Kalimo, A. (toim.). Graafisen käyttöliittymän suunnittelu: Opas ohjelmistojen käytettävyyteen. Espoo: Suomen ATK-kustannus Oy, 22-24.
Kriik, G. 2018. Prototyypit ohjelmistokehityksessä. https://www.arter.fi/prototyypit-ohjelmistokehityksessa/.29.05.2019.
Nielsen, J. 2012. Usability 101: Introduction to Usability. Nielsen Norman Group. https://www.nngroup.com/articles/usability-101-introduction-to-usability/. 10.11.2018.
Nieminen, P. 2018. Selviytymisopas: Komentorivin käyttö. Jyväskylän Yliopisto. http://users.jyu.fi/~nieminen/ohj1/materiaalia/tyokaluohjeet/komentorivi_selviytyminen.html. 18.11.2018.
Shneiderman, B. Plaisant, C. Cohen, M. Jacobs, S. Elmqvist, N. Diakopoulos, N. 2017. Designing The User Interface: Strategies for Effective Human-Computer Interaction. Harlow: Pearson.
Sinkkonen, I. Nuutila, E. Törmä, S. 2009. Helppokäyttöisen verkkopalvelun suunnittelu. Helsinki: Tietosanoma Oy.
UXPin. 2018a. States. https://www.uxpin.com/community/tutorials/states/. 22.03.2019.
UXPin. 2018b. Variables. https://www.uxpin.com/community/tutorials/variables/. 22.04.2019.
UXPin. 2019a. Projects and Documents. https://www.uxpin.com/docs/dashboard/projects-and-documents. 20.05.2019.
UXPin. 2019b. Interactions. https://www.uxpin.com/docs/editor/interactions. 30.05.2019.
UXPin. 2019c. Downloading and Using UXPin. https://www.uxpin.com/docs/getting-started/downloading-and-using-uxpin.14.05.2019.
UXPin. 2019d. API Request. https://www.uxpin.com/studio/updates/api-request/. 24.05.2019.
UXPin 2019e. Expressions. https://www.uxpin.com/docs/editor/expressions. 24.05.2019.
Virtanen, J. 2018. UX-design ja UI-design: Mitä eroa niillä on?. https://contrast.fi/ux-design-ja-ui-design-mita-eroa-niilla-on/. 15.11.2018.
Wesolko, D. 2016. Peter Morville's User Experience Honeycomb. Medium.

https://medium.com/@danewesolko/peter-morvilles-user-experience-honeycomb-904c383b6886. 20.02.2019.

Wikipedia. 2018. Confluence. https://fi.wikipedia.org/wiki/Confluence. 15.05.2019.