TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Maarja-Liis Helü 143048

KASUTAJALIIDES KATEGORISEERITUD PROJEKTIDE VAATAMISEKS: ANALÜÜS JA ARENDUS

Bakalaureusetöö

Juhendaja: Ago Luberg
Magistrikraad

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Maarja-Liis Helü

22.05.2017

Annotatsioon

Kasutajaliides on tähtis igapäevaelus ja samuti infotehnoloogias. Hea kasutajaliidesega on parem töötada ning seetõttu tõuseb kasutajakogemus. Tänapäeval on hea viis kasutajate hoidmiseks kasutajaliidest uuendada ning uusi tehnoloogiaid kasutada.

Antud lõputöö eesmärgiks on luua uus kasutajaliides olemasolevale veebilehele kuldmuna.ee/arhiiv, kasutades kasutajaliidese tavasid. Töös on käsitletud, mis on kasutajaliides, kasutajakogemus, miks need on tähtsad. Välja on toodud, mida inimesed veebilehtedel vaatavad ning antud nippe hea kasutajaliidese loomiseks. Teises peatükis on kogutud nõuded, tehtud mockup, analüüsitud JavaScripti raamistikke ning nendest sobiv välja valitud. Veel on räägitud veebilehe arendusest ja kirjeldatud veebilehe komponentide valikuid ning võrreldud uut veebilehte olemasolevaga.

Lõputöö tulemuseks on uus kasutajaliides, mis sobib veebilehele kuldmuna.ee/arhiiv. Veebilehel on võimalik vaadata projektide nimekirja, projektide kataloogipuud ning projekte. Antud kasutajaliidest on võimalik kasutada ka teiste andmebaasidega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 4 peatükki, 14 joonist.

Abstract

User interface for presenting categorized projects: analysis and development

The user interface is important in everyday life as well as in information technology. A good user interface is better to work with and, therefore, increases the user experience. Upgrading user interface and using new technologies are good ways to keep users to visit your site.

This thesis aims to create a new interface to an existing web page kuldmuna.ee/arhiiv using the theory of a good user interface. The paper covers what is user interface and user experience and why are they important. Then it brings out what users do on websites and provides tips to follow on website development. In the second part the requirements are gathered, mockup is done, different JavaScript frameworks are analysed and the best one is picked out. More is talked about developing a website and described the choice of components. Then the new website is compared to the existing one.

Different views are put in place in mockup for developing a user interface. Functionality is done with Angular, which was the selected JavaScript framework. Interface design was added with CSS. Various choices about components, like treeview and pagination, are analysed.

The result of a thesis is a new and improved user interface which suits for a kuldmuna.ee/arhiiv web page. On the website it is possible to view list of projects, projects directory tree and projects with data about them. This interface can also be used with other databases.

The thesis is in Estonian and contains 31 pages of text, 4 chapters, 14 figures.

Lühendite ja mõistete sõnastik

GUI	*Graphical User Interface*, graafiline kasutajaliides
HTML	*Hypertext Markup Language*, hüperteksti märkekeel
JSON	*JavaScript Object Notation*, andmevahetusvorming
UI	*User Interface*, kasutajaliides
URL	*Uniform Resource Locator*, universaalne ressursilokaator
UX	*User Experience*, kasutajakogemus

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tänapäeval on väga palju veebilehekülgi [1] . Nendest osa on unustusse jäänud või täidavad küll oma eesmärki, kuid nende välja nägemine on aastate jooksul samaks jäänud. Sellised veebileheküljed võivad kasutajad eemale peletada oma vana kujundusega või ebamugava kasutajaliidesega. Selleks võtan ette ühe vana veebilehe, millele teen uue kasutajaliidese.

Projekt on vajalik vananenud kasutajaliidese uuendamiseks. Kasutajaliides on mõeldud kasutajatele, et nende lehekülje külastus oleks mugavam.

Käesoleva bakalaureusetöö eesmärgiks on luua uus kasutajaliides, kus on võimalik vaadata kategoriseeritud projekte ja kaustu ning nende vahel liikuda. Antud lõputöös teen uue ja tänapäevase kasutajaliidese hetkel olemasolevale veebilehele, milleks on http://www.kuldmuna.ee/arhiiv/. Veebilehe funktsionaalsus on piisavalt üldine ning seda on ka võimalik kasutada teiste andmebaasidega, kus on olemas failid ja kaustad. Veebilehe kujundusel toetun hea kasutajaliidese ja kasutusmugavuse tavadele.

Esimeses peatükis võtan kokku teooria, mida on vaja praktilise osa jaoks. Defineerin, mis on kasutajaliides ja kasutajakogemus ning mis on nende tähtsus tänapäeva maailmas. Uurin, kuidas inimesed veebilehti vaatavad ja lõpuks loetlen üles teadmisi, mida järgida paigutusest, värvidest ning elementidest veebilehe tegemisel.

Teises peatükis tegelen kategoriseeritud veebilehe loomisega. Kõigepealt alustan nõuete kogumisest ning teen valmis veebilehe mockupi. Järgmiseks analüüsin erinevaid JavaScripti raamistikke ning valin projekti jaoks sobiva välja. Lõpuks kirjeldan veebilehe loomist, kus teen funktsionaalsuse ja disaini ning igat valikut põhjendan teooriale põhinedes. Lisaks võrdlen uut kasutajaliidest olemasolevaga.

2 Kasutajaliides ja kasutajakogemus

Infotehnoloogias kasutajaliides on tarkvara kasutamiseks ja selle kontrollimiseks. Kasutajaliideseks võib olla näiteks ekraan, klaviatuur või arvutihiir. Kuid see on ka viis kuidas kasutaja suhtleb rakenduse või veebilehega[2]. Hea kasutajaliides pakub kasutajasõbraliku kogemuse, võimaldades kasutajal suhelda tarkvara või riistvaraga loomulikult ja intuitiivselt. [3]

Programmidel võib olla olemas graafiline kasutajaliides ehk GUI (*Graphical User Interface*). See tähendab, et kasutajal on võimalik programmi graafiliselt kontrollida, kus saab valida elemente kasutades hiirt või klaviatuuri. Tüüpilisel graafilisel kasutajaliidesel on menüüriba, tööriistariba, aknad, nupud ja muud kontrollerid. Näiteks *Macintosh*i ja *Windows*i operatsioonisüsteemidel on erinevad kasutajaliidesed, kuid mõlemal on sarnaseid elemente, mis võimaldavad inimesel mõlemat operatsioonisüsteemi kasutada ilma ümber õppimata. [3]

Kasutajakogemus ehk UX (*User Experience*) on ISO ehk Rahvusvahelise Standardiorganisatsiooni poolt defineerituna inimese taju ja reageering, mis tuleneb teatud toote, süsteemi või teenuse kasutamisest[4]. Kui inimene kasutab näiteks veebilehte ning see tekitab talle oma välimuse ja funktsionaalsusega tunde, et see on usaldatav või huvitav, siis kõik need tunded kokku moodustavadki kasutajakogemuse.

Kuid tasuks meelde jätta, et kasutajaliides ja kasutajakogemus pole sama asi. Kasutajakogemusest võib rääkida ilma kasutajaliidest kordagi mainimata. UX on kõik see, mis paneb veebilehe hästi tööle ja UI on kõik see, mis teeb hästi töötava veebilehe ilusaks ja inimesele veetlevaks. Liides hõlmab kõike alates veebilehe liikumisvoolu disainimisest kuni selle väljanägemiseni. Kogemus võtab disaini ja teeb kindlaks, et kasutaja on samal tasemel ning annab edasi vastavat kogemust läbi UX.[5]

2.1 Kasutajakogemuse vajalikkus

Kasutajaliidese disainimisest ainult ei piisa, vaid tuleb arvestada ka kasutajakogemust. Iga tootel, mis on mõeldud inimestele, on olemas kasutaja ja iga kord kui seda toodet kasutatakse, edastab see kogemuse. Võttes näiteks lihtsa toote nagu tool, siis selle kasutamiseks istutakse selle peale. Toode ei pruugi rahuldavalt täita ootusi, kui see ei kannata inimese kaalu. [6]

Keerukamate toodetega on nõuded eduka kasutajakogemuse edastamiseks sõltumatud toote definitsioonist. Telefon on mõeldud edastama ja vastu võtma kõnesid, kuid tänapäeva mobiiltelefonil on palju rohkem funktsioone kui lauatelefonil. Keerukama tootega on raskem kindlaks teha, kuidas edastada head kasutajakogemust. Eduka toote loomise protsess on üsna raske ja seetõttu peab see olema toetatud kasutajakogemuse disaini poolt. [6]

Tänapäevane põhiaspekt kogemustes on jagatud vastastikune vaev ja tasu. Tarbijad harjumuspäraselt ja aktiivselt seovad ennast kaubamärkidega nagu ei kunagi varem. Erakordsete kogemuste loomine kasutaja ümber on see, mis eristab maailma kõige edukamaid kaubamärke. Group XP poolt läbi viidud uuringus leidis kinnitust, et organisatsioonid, kes loovad rikkaliku ja sujuva kogemuse, mis on ehitatud ümber autentse brändi eesmärgi, saavad rahalist kasu ja suure kogemuste kapitali. [7]

2.2 Hea kasutajaliidese tähtsus

Hästi kujundatud liides on väga tähtis kasutajatele, sest läbi selle on võimalik näha süsteemi erinevaid võimalusi. Paljude jaoks on see kogu süsteem, kuna see on üks nähtav osa tootest, mida arendajad loovad. Samuti läbi selle toimub ülesannete täitmine. Nendel ülesannetel on tavaliselt otsene mõju organisatsiooni suhetele klientidega. Ekraani kujundus ja välimus mõjutavad inimest erinevatel viisidel, näiteks segase ja ebaefektiivse puhul on inimesel raskem teha oma tööd ning seeläbi tehakse rohkem vigu. Kehv disain võib mõne inimese veebilehelt jäädavalt eemale peletada. Samuti võib halb disain põhjustada viha, pettumust ja stressi. [8]

2002. aasta uuringus „Kuidas inimesed hindavad veebilehe usaldatavust?” tuli välja, et keskmine tarbija pööras tähelepanu rohkem veebilehe visuaalsele poolele kui selle

sisule. Peaaegu pooled tarbijad, mis oli 46.5%, hindasid lehekülje usaldusväärsust üleüldise visuaalse disaini poolest, milleks olid paigutus, tüpograafia, kirja suurus ja värvilahendus. [9] Selle põhjal võib leida, et kui veebilehe disain ei ole kasutaja jaoks piisavalt hea, siis ei pruugi nad seda väga kasutada ja otsivad järgmise veebilehe, kuigi selle funktsionaalsus võib olla sama.

Veetlevad asjad töötavad paremini, väitsid kaks jaapanlasest uurijat. Nende poolt oli tehtud uuring, kus oli üles pandud kaks pangaautomaati sama funktsionaalsusega, aga erineva paigutusega – atraktiivne ja mitteatraktiivne. Jaapanlased leidsid, et ilusamat oli kergem kasutada. Iisraeli teadlane kordas katset Iisraelis, arvates et katsel oli puudusi või tegemist oli kultuuriliste erinevustega. Teine katse aga kinnitas tugevamini, et ilusa väljanägemisega pangaautomaat on kergemini kasutatav. [10]

Donald Normani häirisid nende uurimuste tulemused, sest ta ei suutnud seletada, miks atraktiivsed asjad töötavad paremini. Kuid siis ta leidis, et varasemalt inimese mõttemaailm püüdis emotsioone peita ratsionaalse ja loogilise mõtlemisega, kuid tänapäeval ollakse vastupidisel seisukohal. Inimene on kõige emotsionaalsem olend ning tähtsad on nii negatiivsed kui positiivsed tunded, kuid viimased soodustavad loovat mõtlemist. Seega näitab, et kui toode on ilus, teeb see inimese õnnelikumaks, mis omakorda soodustab loovat mõtlemist ning teeb lihtsamaks tekkinud probleemide lahendamise. [10]

2.3 Kasutajad veebilehel

Veebilehe tegemisel tuleks kõigepealt teada, millised on inimesed, kes seda hakkavad vaatama. Tuleb mõelda, kellele on veebileht mõeldud ja milline on publik. Mõista, mis on kasutajate motivatsioonid, hirmud, püüdlused ning mis on nende eesmärk veebilehe kasutamisel. Disainimisel võib inimesed välja mõelda ja kujutada ette üksikasjalikult, millised on nende isiksused, et saaks ennustada, kuidas nad esimesel korral saiti külastavad. [11]

Pärast veebilehe jaoks väljamõeldud tegelaskujusid, tuleb neile anda tegevused, mida nimetatakse kasutusjuhtudeks. Kasutusjuhtudel kaardistatakse tegevused iga kasutaja jaoks esimesest kokkupuutest veebilehega kuni eesmärgi täitmiseni. Kui isikul ei ole

võimalik eesmärki täita või see on liiga keeruline, siis on olemas visuaalne probleem ning seda on kergem lahendada. Kasutusjuhud on hea viis testimaks veebilehe struktuuri. [11]

Veebidisainis kasutatakse Guthenbergi diagrammi, mis kasutab lugemisgravitatsiooni – vasakult paremale, ülevalt alla [12]. Nielsen Norman Groupi silmajälgimissüsteemi uuringus tuli välja, et inimesed vaatavad veebilehte hoopis F-kujulise mustrina. Lisaks paljud kasutajad ei loe teksti põhjalikult ning seda teevad ainult mõned. Esimesed kaks lõiku peaksid sisaldama kõige olulisemat informatsiooni, kuid enamasti loetakse ainult esimest. Alapealkirjad ja lõigud peaksid sisaldama alguses kõige tähtsamaid sõnu, mis võiksid tekitada inimestes huvi, kui nad sirvivad veebilehte. [13]

Inimesed kalduvad rohkem vaatama veebilehe vasakut poolt. Samuti on välja toodud Nielsen Norman Groupi uuringus, kus selgus, et kasutaja vaatab 69% ajast vasakut poolt lehest. Sellest võib välja tuua, et tähtsam sisu tuleb hoida vasakul pool. Seetõttu navigatsioonimenüü hoida kõige vasakul ja selle kõrval sisu. Vertikaalselt tähtsamat informatsiooni näidata ühest kolmandikust kuni poole leheni, kuna see on koht, kuhu kasutajad suunavad oma tähelepanu. Teisejärguline sisu tuleks panna paremale, sest see ei jää nii väga ette. [14]

2.4 Mida järgida veebilehe tegemisel

Veebilehte ehitama hakates on raske välja tulla millegi uuega. Läbi aegade on inimesed harjunud teatud veebilehtede kujundusega. Kui aga mingid elemendid panna uute kohtade peale, võivad inimesed segadusse sattuda ning vajaliku otsimine on raskendatud. Kasutaja tegeliku eesmärgi täitmine veebilehele tulles võtab palju rohkem aega ja seeläbi tekib frustratsioon, mille tagajärjeks võib olla veebilehelt lahkumine.

Veebilehe loomisel veenduda, et kasutajad saavad oma peamist eesmärki täita kiiresti ja kergelt. Oluline on, et leht tunduks kiire, isegi kui see ainult tundub nii. Esmalt tuleks näidata veebilehe elementide kondikava samal ajal kui ülejäänud veebileht veel laeb. Teksti tuleks näidata ennem kui pilte, et inimene saaks juba lugema hakata. Mitmesekundilised viivitused panevad inimesed saidilt lahkuma. [15]

2.4.1 Paigutus ja liikumine

Enamik tekstile orienteeritud informatsiooni veebilehti koonduvad suhteliselt järjepidevaks paigutuses – päis, jalus, kohalik navigatsioon ja sisu elemendid. Need koos annavad tuttava lähtepunkti veebiliidese kujundamiseks. Üldiselt inimesed leiavad, et tuttavat on kergem kasutada ja meelde jätta. Kui veebileht järgib neid mustreid, siis kasutajad kohanevad kiiremini ja saavad keskenduda unikaalsele sisule ning funktsioonidele. Samuti meeles pidada, et veebiliides ei tohiks kunagi konkureerida lehe sisuga kasutaja tähelepanu pärast. [16]

Kasutajad kerivad veebilehel allapoole, kui neil on selge, et asjakohast teavet on veel allpool tulemas. Veebilehel peaks hästi aru saama, kuhu poole peab kerima ja kas rohkem sisu on veel saadaval. Üks lehekülg ei tohiks olla väga pikk, sest tavaliselt inimene väsib enne ära, kui ta jõuab lõppu. Kuigi kerida on alati mugavam ja seda on kiirem teha kui vajutada, siiski peaks veebilehe pikkus jääma mõistlikkuse piiridesse. [15]

Kasutajale tuleb anda informatsiooni, kus ta veebilehel asub. Selleks võib kasutada jäljerida(*breadcrumb*). Menüü ripploendid peaksid olema vertikaalsed, mitte horisontaalsed, sest viimast on raskem kerida. Suured menüüd peaksid olema kitsamad kui lehekülg, et neist oleks võimalik välja minna. Sellise menüü kasutamisel organiseerida lingid gruppidesse ning eristada vajutatavaid ja mitte-vajutatavaid elemente. Sisse logimine ja otsing peaksid olema alati nähtavad, mitte menüüsse peidetud. [15]

Veebilehel võiks alati olla otsinguväli, väljaarvatud juhul kui lehel on väga vähe sisu. Otsing võiks alati välja näha kui tekstiväli. Otsinguväli peaks olema piisavalt pikk, et näha tervet otsingu päringut. [15] Inimesele kasuks tulevad elemendid, näiteks otsing, sisse logimine, kasutajakonto registreerimine ja seaded, tuleks panna lehekülje ülemisse paremasse nurka [17] .

Sageli jäetakse tähelepanuta, kuid veebilehe juures on tähtis ka valge ruum erinevate komponentide vahel [18] . Soovitatav on kasutada ühe veeru paigutust. Osade elementidega võiks ülekattuda teistele, et rõhutada järjepidevust. Pealkirjad paigutada lähemale vastavatele peatükkidele ning mitte minna pealkirjaga üle servade teistesse

sektsioonidesse. Sarnased funktsioonid või menüü elemendid on parem grupeerida ühte. [17]

Pidevad ikoonid, graafilised skeemid, pealkirjad, graafilised- või tekstipõhised ülevaated ja kokkuvõtted annavad kasutajale kindluse, et nad leiavad, mida nad otsivad ilma aega raiskamata. Kasutajatel peaks alati olema võimalus naasta koduleheküljele ja teistele peamistele navigatsiooni punktidele. Vältida tupik lehekülgi, sest sageli inimestel on võimalik linkidega minna järjest sügavamale hierarhiasse, nii et nad ei leia enam teed koduleheküljele. Kui alamlehed ei sisalda linke kodulehele, blokeeritakse kasutaja ülejäänud saidist. [16]

2.4.2 Värvid

Värv võib olla üks viise brändi identiteedi näitamisel. Mõni bränd on nii tugevalt seotud värvidega, et seda on raske ilma ette kujutada. Heaks näiteks on Coca-Cola, kes on kasutanud punast värvi järjekindlalt aastate jooksul, et luua seeläbi oma identiteeti. See ei tähenda teiste värvide välistamist, vaid pigem on põhivärv osa laiemast värvipaletist. [6]

Värvide ja kontrastsuse disainimisel tuleks arvesse võtta inimesed, kes on värvipimedad. Selleks on võimalik oma disain viia hallidesse toonidesse, mis tagaks, et kõigil inimestele on võimalik veebilehel lugemine. Teksti jaoks mitte sinist värvi kasutada, vaid see jätta ainult linkide jaoks. Soojad ja erksad värvid tulevad rohkem esile, kuid tumedad ja külmad värvid jäävad taustaks. [15]

Kui värve õigesti kasutada, siis see on hea võimalus saada kasutajaid keskenduma funktsioonidele ja olulistele osadele veebilehel. Näiteks punane on üks julgemaid toone, mis nõuab ja tõmbab tähelepanu. See näeb isegi veel erksam kui kasutada seda koos külmade toonidega nagu roheline või sinine. Kollane kuju sinisel taustal hakkab kohe silma. [18]

Kasutajaliidese värviteooria võib jagada kolme osasse: kontrastsus, täiendav ja elavdav. Igal värvitoonil on olemas vastand, mille kontrastsus on kõige suurem kui mistahes muul värvil. Värvid ei ole alati üksteisega vastuolus, täiendavad värvid rõhutavad üksteist ja toovad välja parima vastupidiselt kontrastsusele. Värvikettal on need

tavaliselt lähestikku, näiteks lillat täiendab roosa või sinine. Iga värv tekitab mingi meeleolu - heledamad ja soojad toonid kipuvad kasutajat erksamaks tegema, samas tumedamad ja külmemad toonid teevad rohkem rahulikumaks. [19]

Tõhusa värviskeemi valimiseks on kolm võimalust – kolmik, ühend ja analoog. Kolmik koosneb kolmest värvispektri erinevatest otsast. Kolmiku värvilahendust on lihtne teha, valides värvirattal ühe baasvärvi ning seejärel joonistades võrdkülgne kolmnurk antud punktist. Kolmnurga tipud moodustavad kolme värvi skeemi nagu on näidatud Joonis 1. Kasutades seda viisi, võib kindel olla, et värvid elavdavad ja rõhutavad üksteist. [20]

Joonis 1: Kolmik värviskeem.[20]

Ühend värvilahendus põhineb värvidel, mis täiendavad üksteist. Värvispektrilt on valitud kaks vastandvärvi nagu Joonis 2. Selline värvivalik annab disainerile rohkem vabadust disainimisel. [20]

Joonis 2: Ühend värviskeem.[20]

Analoogne värviskeem põhineb hoolikalt valitud värvidest värvispektri samast piirkonnast. Tavaliselt värvid erinevad nende elavdatuse või kontrastsuse poolest, kui neid üksteisega võrrelda. Analoogse värviskeemi näideteks on kollase ja oranži varjundid või baasvärvitooni varjundid (Joonis 3). [20]

Joonis 3: Analoog värviskeem.[20]

2.4.3 Elemendid

Vormide kasutamisel panna tähele, et sildid ja väljad oleksid üksteise all nende paremaks läbi vaatamiseks. Silti on parem kasutada väljaspool tekstivälja, et kõigel oleks võimalik silma peal hoida. Pikad vormid tuleks jaotada osadeks koos eraldajatega, et see oleks kasutajasõbralikum. Viga peaks olema välja, mis tekitab viga, kõrval. Veateated peaksid olema abiks kasutajale vea mõistmiseks, sisutihedad ja kergesti arusaadavad. [15]

Nuppude kasutamisel jälgida, et need näeksid välja vajutatavad ning nende ümber on piisavalt ruumi. Sagedased tegevused peaksid olema suurtel nuppudel ja kergesti kättesaadavates piirkondades. Taustavärvid, ääred ning tegevusele suunatud sildid annavad kasutajale aimu, et tegemist on vajutatava elemendiga. Veebileht peaks andma visuaalselt märku, et nupuvajutus oli edukas lühikese aja jooksul. [15]

Lingid peaksid veebilehtedel silma paistma, kas sinise värviga või allajoonituna. Kasutaja ei pea linki vajutades teada saama kuhu see viib, vaid seda peaks ütlema lingi tekst. Viide täielikule URLile saidil, peaks alati viima vastavale lehele. Teatud elemendid, näiteks eeldatakse, et toote pildid või ülevaade on alati vajutatavad.

Külastatud lehtede lingid võiksid olla teist värvi, et vähendada kasutaja mälukoormust tuletades meelde, et leheküljel on juba käidud. [15]

3 Veebilehe loomine

Selles jaotuses räägin, kuidas ma ehitasin veebilehe üles. Millest alustasin, mida kasutasin loomisel, miks midagi tegin ja mis probleemid kaasnesid.

3.1 Nõuded

Veebilehe jaoks panin paika funktsionaalsed ja mittefunktsionaalsed nõuded, mida järgisin lehe tegemisel. Funktsionaalsed nõuded on need, mis kirjeldavad seda, mida süsteem peaks tegema. Mittefunktsionaalsed nõuded seavad funktsionaalsetele nõuetele piirangud, kuidas midagi peab tegema. [21]

Funktsionaalsed nõuded:

- Veebilehele sisenedes peab näitama pealehel kõiki projekte
- Veebilehel on kataloogipuu, mis näitab projektide struktuuri
- Projektide listi saab vaadata nimekirja ja ruudustikuna
- Projekti vaates on näha kõiki pilte ja videoid
- Projekti vaates on näha kõiki projektiga seotud andmeid
- Otsingu tulemusena näidatakse projektide listis kõiki vasteid
- Otsingu tulemusena näidatakse kataloogipuus neid katalooge, kus leidus vaste

Mittefunktsionaalsed nõuded:

- Kasutajaliides peab olema eesti keeles, kuid seda peaks olema võimalik tõlkida
- Kasutajaliides peab olema veebipõhine

- Otsing ei tohi võtta mitte rohkem kui 3 sekundit aega
- Veebileht peab vastama kiiremini kui 3 sekundit

3.2 Mockup

Enne veebilehe ehitama hakkamist tegin *mockup*i. *Mockup* on visand, mis tähistab seda, kuidas veebileht hakkab välja nägema. Minu arust tuleb mockupi tegemise peale kulutada aega minimaalselt, sest reaalse veebilehe tegemine on palju mahukam. *Mockup* on tavaliselt staatiline lehekülje väljanägemine, millel on veel lisaks kujunduslikud detailid kuid puudub interaktsioon.

Veebilehe visualiseerimisel on seda kergem alustada ning on eesmärk kuhu poole liikuda. Teisalt ei jätnud ma *mockup*i kindlaks raamiks ning uute ideede tulekul ei piiranud neid töösse lisamistel *Mockup*i tegemiseks kasutasin programmi Axure RP 8, kus oli piisavalt elemente olemas ning neid sai mugavalt tõsta ja tirida vajaliku koha peale.

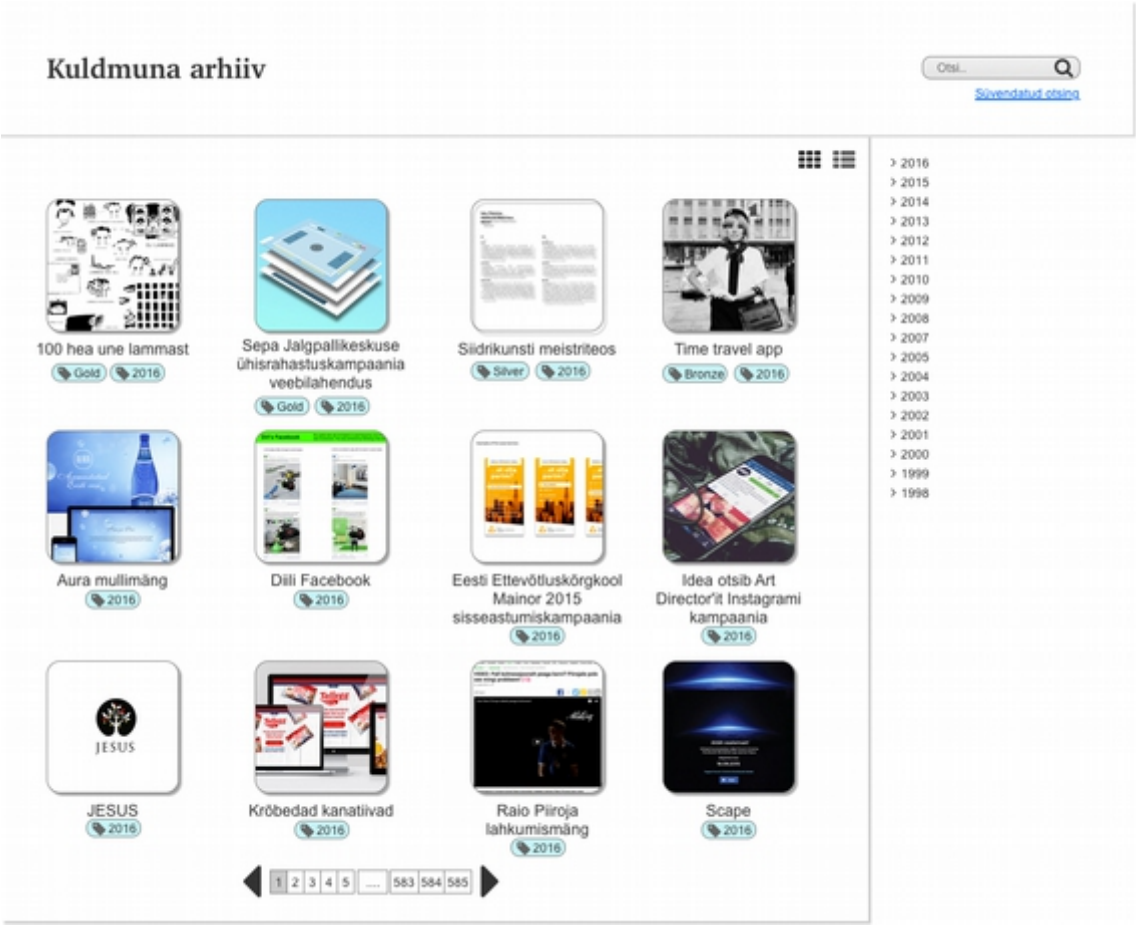
Otsustasin veebilehe jagada kolmeks :

1) Pealkirja paneel – lehekülge tutvustav osa ning otsing.
2) Sisu paneel – nimekiri projektidest ning pildid.
3) Parempoolne küljepaneel – kataloogipuu ja projekti andmed.

Kuna tegemist on projektide vaatamise veebilehega, siis kõige tähtsamad on pildid ning kohe pealehel (Joonis 4) asuvad projektid. Seega peaksid pildid olema piisavalt suured ja andma edasi juba pisipildina seda, mida projekt endast kujutab. Vaikimisi on esilehel ruudustik vaade, kus on näha projekti pilt ja selle pealkiri, lisaks oli mõte panna märgetena kaasa aasta ja auhind. Sisu paneeli üleval nurgas on valik, kas vaadata projekte ruudustikuna või nimekirjana.

Paremale paigutasin küljeriba ning sinna panin kataloogipuu, et oleks kogu aeg näha, kus kohas parasjagu asutakse. Kataloogipuud on võimalik avada ja erinevate lehekülgede vahel liikuda. Üleval paremal nurgas on otsing, sest kui inimene on

harjunud midagi leidma, siis vaadatakse lehekülje ülemisse ossa ja valdavalt paremale nurka [15] . Otsing on üleüldine ning lisavõimalusena saab teha süvendatud otsingut, kus on võimalik rohkem parameetreid sisestada.

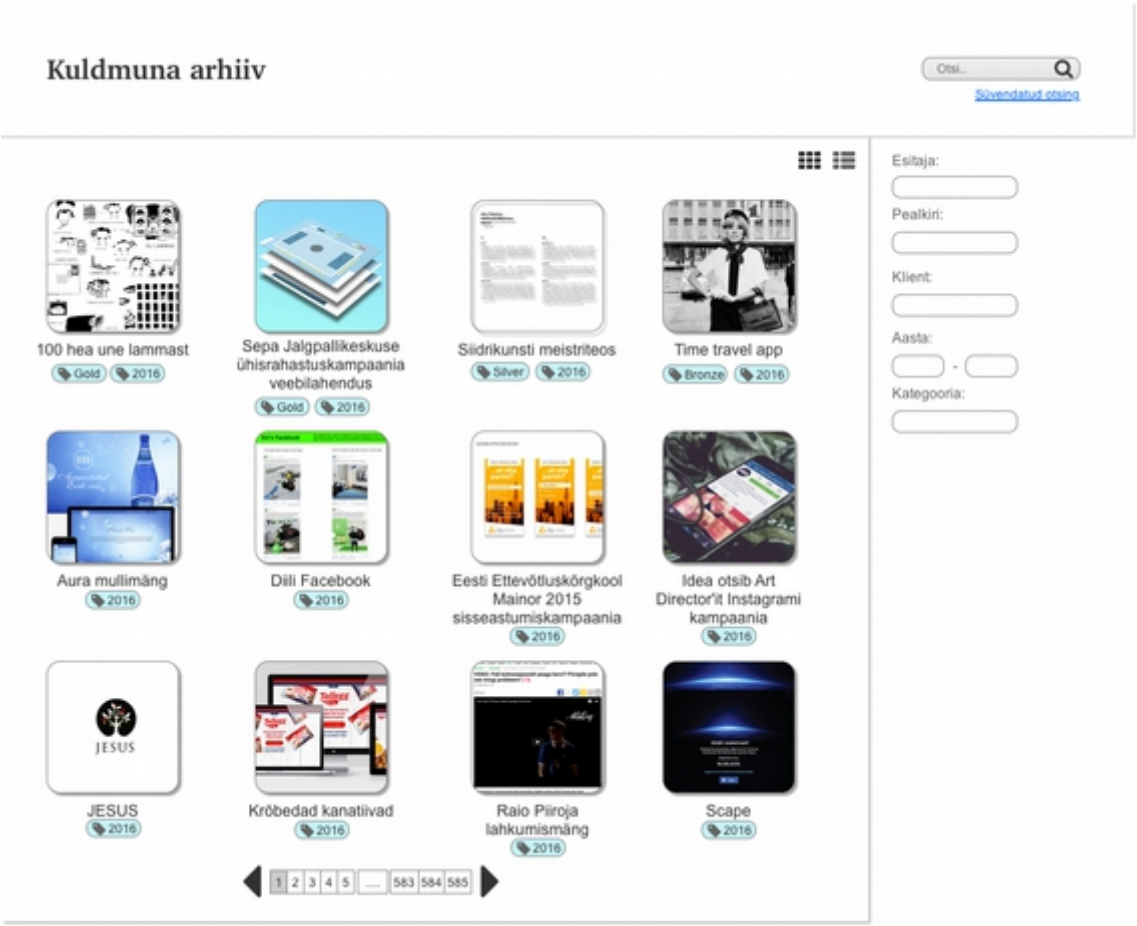


Joonis 4: Pealehe vaate mockup.

Kasutajal on võimalik valida ka nimekirja vaade (Joonis 5), kus on projekti pilt vasakul pool ja paremal pool projekti andmed. Kuna selles vaates on rohkem ruumi, siis on peale projekti pealkirja välja toodud esitaja, aasta ning märgetena auhinna koht ja kategooriad. Selles vaates ülejäänud informatsioon on kõik sama nagu ruudustikvaatel.

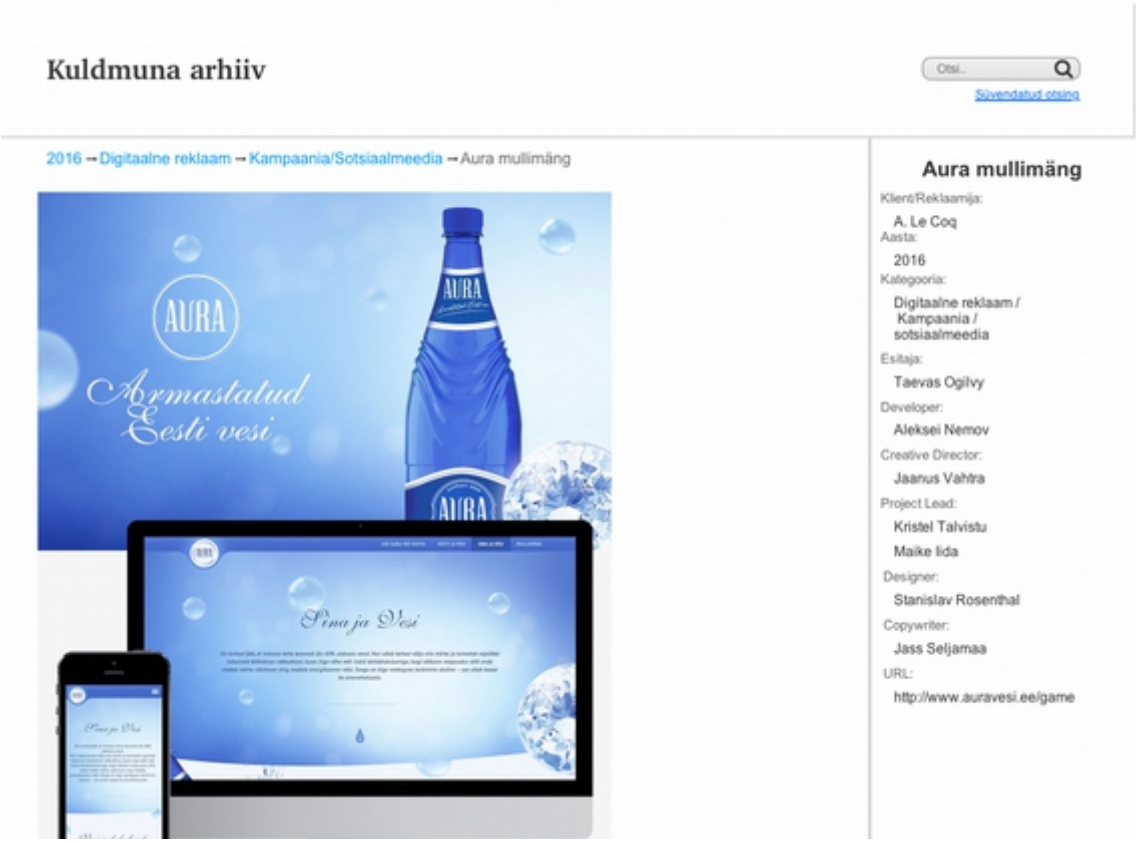
Joonis 5: Nimekirja vaate mockup.

Lisaks tavalisele otsingule on ka süvendatud otsing, mis avaneb kui vajutada otsingu all olevale lingile „süvendatud otsing” (Joonis 6). Otsing tuleb samasse kohta paremale poole küljepaneelile, kus asub kataloogipuu. Süvendatult on võimalik otsida pealkirja, autori, kliendi ja kategooria järgi, lisaks saab määrata ajavahemiku, millal projekt on esitatud.



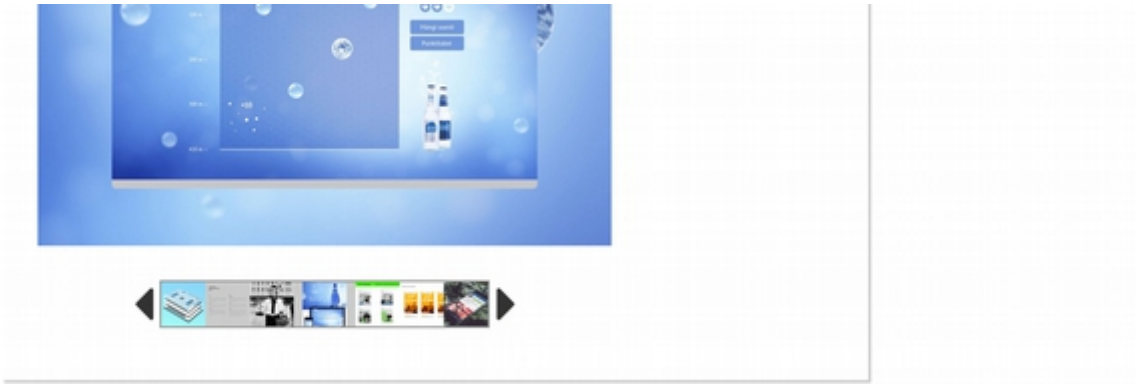
Joonis 6: Otsingu vaate mockup.

Projekti vaates (Joonis 7) on esikohal kõikide piltide ja videote suures mõõdus vaatamine. Sellisel viisil ei pea kasutaja iga pildi pärast klõpsu tegema, vaid saab kerida kõikide valikute vahel. Projekti vaates on näha üleval osas jäljerida, mis näitab hierarhiliselt, kus inimene hetkel asub ning läbi selle on võimalik tagasi minna. Piltide kõrval paremal, on näha projekti andmed. Suuremalt on kirjas pealkiri ning väiksemalt on esitaja, aasta reklaamija ja muud andmed. Kõikidel projektil ei pea olema ühepalju andmeid, vaid see võib varieeruda.



Joonis 7: Projekti vaate ülemise osa mockup.

Lisaks projektivaate lõpus on võimalik minna eelnevale või järgnevale projektile. Noolte asemel on pandud eelvaate pildid, et oleks parem arusaam, mis projektile minnakse.



Joonis 8: Projekti vaate alumise osa mockup.

3.3 JavaScripti raamistiku valimine

Raamistik on üldine struktuur, mis on mõeldud toetama veebilehe ehitamisel [22] . Lisaks saab raamistikku kasutades vähem koodi kirjutada ja selle võrra rohkem ära teha. Raamistik on kirjutatud inimeste poolt, kes valdavad oskuslikult JavaScripti ja seetõttu ei pea endal olema süvendatud teadmisi antud valdkonnas. [23]

Peale arendaja töö lihtsustamist aitab raamistik saavutada kiirema reaktsiooniga kasutajakogemust. Näiteks teatud raamistikel on võimalus, et nupuvajutusel laaditakse terve lehe asemel ainult teatud osad, mida kasutajal on vaja. Selline käitumisviis kiirendab kasutajaliidese reageerimisvõimet, millega traditsioonilised serveripoolsed veebirakendused võivad hätta jääda. Väledat kasutajaliidest saab teha ka tavalise rakendusega, kuid tihti võivad tulla erinevad probleemid, mis viivad keerulise ja mahuka koodini. Kuid modernsed JavaScripti raamistikud pakuvad paremat koodi haldamist kindlaks määratud rakenduse arhitektuuridega, mis kergendavad suurel määral arendamist. [24]

JavaScripti raamistikke on mitmeid ja neist paljudest jäid peale Angular, React, Ember ja BackBone. Võrdlesin raamistikke selle järgi, kui populaarne see hetkel on, kui suur on hetkeline programmeerijate kogukond, kui palju on selle kohta informatsiooni, kui arusaadav on kood ning kas minu projektile vajaminevaid osi on olemas.

AngularJS üheks peamiseks omaduseks on kahesuunaline andmete sidumine ning mudel ja vaade saavad mõlemad andmeid värskendada, mis omakorda tähendab vähem koodi kirjutamist dünaamiliste vaadete jaoks. Teine omadus on direktiivid, näiteks ng-repeat, mis võimaldab arendajatel elemente korrata, mis teeb mugavamaks leheküljele massiivi elemente näidata. Sõltuvuste süstimine on funktsioon, mis võimaldab arendajatel lisada teenuseid oma moodulitesse. Lisaks, kuna Angular on Google hõlma all, siis annab see juurde stabiilsust. Angular võib olla aeglane mahukate rakenduste korral, eriti kui on tegemist keeruka kasutajaliidesega. [24]

Angular töötab paljudel juhtudel väikestest projektidest kuni ettevõtete rakendusteni [24] . Lisaks kasutades Google trendi(Joonis 9) leidsin, et nendest neljast on varasemalt olnud kõige populaarsem AngularJS. Lugedes erinevaid artikleid ja soovitusi, millist raamistikku valida, oli AngularJS seal alati olemas ning enamasti esimeste seas.

Programmeerijate kogukond on üsna suur, kuna googlest ja stack overflowst otsides tuli palju tulemusi. Vaadates erinevaid koodinäiteid ja neid läbi proovides, oli seda hästi aru saada. Lisaks leidus palju materjali minu veebilehe jaoks olevate elementide jaoks nagu kataloogipuu, piltide ja informatsiooni esitamine.

React on Facebooki poolt arendatud raamistik, mis keskendub mudel-vaate-kontrolleri arhitektuurimustris pigem vaatele. Eelnevalt JavaScripti kogemusega arendajale on kergem Reacti õppida. React on komponentidel põhinev, kus iga üks neist esitab ühte kasutajaliidese osa. Neid komponente saab omavahel sobitada, mis lubab maksimaalselt koodi taaskasutada. Kõige vastuolulisem aspekt Reacti juures on kasutajaliidese jaoks komponentides mallide mitte kasutamine. [24]

React on praegu tõusuteel ning igas suuruses projektidel on võimalik edukalt seda kasutada [24] . Google trendi(Joonis 9) alusel oli React varasemalt teisel kohal, kuid hiljutised tulemused näitavad, et see on tõusnud esikohale. Samuti oli enamikes artiklites Reacti väga palju mainitud. Programmeerijate kogukond Reacti ümber tundus palju väiksem, kuid materjali on siiski piisavas koguses. Lisaks on olemas kataloogipuu kohta näiteid. Kood tundus väga tavalise JavaScripti sarnane olevat.

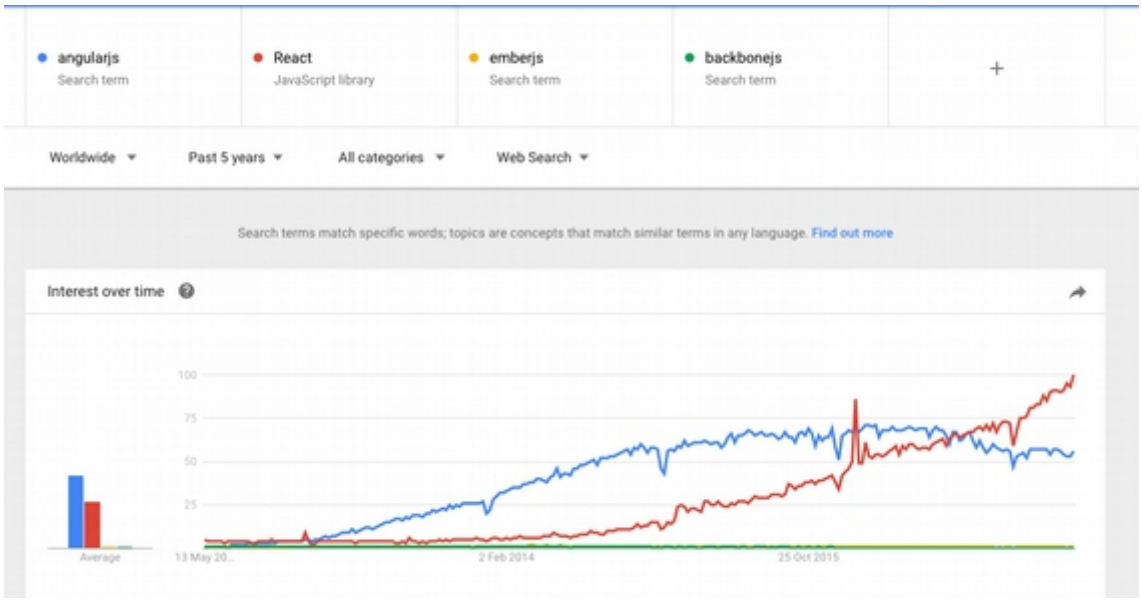
Ember kirjeldab ennast kui „Ambitsioonikas veebirakenduse ehitamise raamistik”. Raamistik on loodud arendajate kogukonna poolt nende enda soovide rahuldamiseks. Rakendusse on sisse ehitatud näiteks mallide teek ja marsruutimine, mis on mõeldud arendajate rutiinist vabastamiseks ja annab seeläbi võimaluse neil kontsentreeruda suurematele probleemidele, mis on nende projektiga seotud. Suur raamistik paljude sisseehitatud rakendustega muudavad selle õppimise raskemaks. [24]

Ember peaks kõige paremini sobima keskmistele või suurtele projektidele [24] . Google Trendi(Joonis 9) arvestades, jääb Ember juba vähem populaarsete hulka. Kogukond ei ole väga suur ning näiteid ja õpetusi nappis, seega algajal oleks võimalike probleemide korral lahendust otsida keeruline.

Backbone tugevus on mitmekülgsus. Kuna see raamistik on kerge, siis sobib kasutamiseks väiksemates projektides, kus on tähtis kiirus. Backbone sobib rohkem arenenud JavaScripti arendajatele. Raamistikul on palju teeke ja pistikprogramme, mida saab omavahel sobitada. [24]

Backbone on raamistik, mis sobib kogenud arendajatele [24] . Olles veel vähe erinevaid projekte teinud ning ka mitte suure kogemusega JavaScripti arendaja, tundub mõistlik see raamistik välja jätta.

Peamiselt jäid pinnale kaks tugevamat React ja Angular (Joonis 10). Mõlemal on piisavalt suur kogukond ja head raamistikud. Valituks osutus siiski Angular, sest selle populaarsus oli kõige suurem, materjali oli piisavas koguses ning programmeerijate kogukond oli arvukas. Lisaks meeldis endale Angulari kirjaviis ning oli olemas hulga erinevaid õpetusi ja näiteid minu projekti jaoks vaja minevatele komponentidele.



Joonis 9: Google Trendi väljavõte 7. mai 2017.

	Angular	React	Ember	Backbone
Kirjanduses mainitud	Esimesel ja teisel kohal	Esimesel ja teisel kohal	Esimese viie hulgas	Esimese viie hulgas
Google's otsingutulemused	16 miljonit	2,7 miljonit	1,4 miljonit	2,1 miljonit
Koodi arusaadavus	+	+/-	-	-
Stack Overflow küsimusi	230 tuhat	42 tuhat	21 tuhat	20 tuhat
GitHubi toetajad	1600	1000	700	300
Lõputööks vajaminevate osade olemasolu	+	+	+	+

Joonis 10: Raamistike võrdlus

3.4 Veebilehe arendamine ja analüüs

Kui olin sobiva raamistiku välja valinud, milleks oli Angular, siis hakkasin veebilehte arendama. Angulari kasutamiseks, tuli mul lisada projektile scripti *tag*i vastav fail. Direktiiv `ng-app` määrab ära, et tegemist on Angulari rakendusega. Hea tava on `ng-app` defineerida HTML sildi sees.

3.4.1 Paigutus

Esialgselt, ilma midagi disainimata, määrasin ära lehekülje struktuuri nagu olin seda teinud algses mockupis – pealkirja paneel, sisu paneel ja parempoolne küljepaneel. Järgmiseks oli kõige tähtsam sisu paneel, kuhu lähevad kõik pildid ja nende andmed, kuid selleks oli mul vaja andmeid. Kuna alguses ei ole midagi serveriga seotud, siis tuli teha *dummy* JSON fail projektide nimekirjast.

Lehekülje ülevale paremas nurgas on otsing. See on üleüldine otsing ning sinna sõna sisestades, otsitakse kõikidelt väljadelt. Lisaks on olemas võimalus süvendatult otsida, milleks tuleb vajutada lingile „Täpsem otsing”. Link avab lehel uue kasti, kuhu on võimalik erinevaid parameetreid sisestada, näiteks pealkiri, autor ja aasta. Kasti saab sulgeda samast lingist või kastis olevast x-st. Sellise otsingu mugavus on see, et süvendatud otsingut ei pea kogu aeg nägema, kuid seda saab vajadusel kasutada. Lahendus sai valitud, kuna paljudel tuntud veebilehtedel, näiteks Youtube, on sarnane otsingu näitamine.

Kogu loodud kasutajaliides töötab Single Page Applicationina, kus iga vaate jaoks ei ole eraldi leht, vaid HTML koodi vahetatakse komponentidena ning seetõttu ei pea tervet lehte uuesti laadima. Vaadete vahetamiseks kasutatakse Angulari funktsiooni ng-route. Index faili sees on element nimega ng-view, kus vahetatakse HTMLi ning ülejäänud jääb samaks terve saidi piires. URLi muutumisel vahetatakse ng-view sees olev HTML, mis on ära määratud ng-route seadistamisel. Näiteks kui on url, mis lõppeb „/project/1”, siis on tegemist projekti vaatega, mille taga olev number on projekti identifikaator. Ng-view sisse määratakse fail gallery-single-project, kus on paigutus projekti piltide ja andmete jaoks. Lisaks määratakse antud HTMLile veel kontroller, mis tegeleb selle faili sees oleva sisuga. URList on võimalik välja lugeda projekti identifikaator, mille abil leitakse andmebaasist vastava projekti andmed. Kui kasutaja kirjutab ise URLi midagi

juurde ehk url ei lange kokku ühegi ära määratuga, siis `otherwise` meetod viib tagasi pealehele. Hilisemas versioonis vahetasin AngularUI Routeri[25] vastu, mis töötab samamoodi, kuid millel on rohkem võimalusi.

3.4.2 Projektide nimekirja vaade

Nimekirja vaate JSON koosneb hulgast projektidest (Joonis 11). Igal projektil on määratud pealkiri, aasta, kategooria, esitaja, auhind ning tee pisipildini. Andmete kätte saamiseks tuli teha mul kontroller, mis tuli siduda vastava elemendiga HTMLis. Kontrolleri nimeks sai `GetProjectsCtrl`. Kasutades Angulari $http teenust, oli võimalik saada kätte nimekirja vaate JSON. Määrates kätte saadud andmed kindla nimega skoopi, siis on võimalik neid kontrolleri all olevas HTML elemendis välja kutsuda.

Kuna mul oli tegemist andmetega, kus igale projektile tuleb sama element teha, siis selleks on Angularil olemas direktiiv `ng-repeat`. Saanud andmed JSON failist HTML'i kuvatud, tegin piltidele koos infoga ruudustik paigutuse. Algselt proovisin tabelina, kuid hiljem jäin *div*ide kasutamise juurde, kuna nendega on rohkem võimalusi. Igas ruudus on pilt keskel ning selle all projektile viitav tekst. Sama JSONit kasutades, tegin ka nimekirja vaate.

```
{
      "projects": [{
                  "title": "Sepa Jalgpallikeskuse
ühisrahastuskampaania veebilahendus",
                  "year": "2016",
                  "category": "Digitaalne reklaam /
Kampaania/sotsiaalmeedia",
                  "applicant": "Neway",
                  "award": "Gold",
                  "thumbnail": "thumbnails/2434.png",
                  "id": "1"
            },
            {
                  "title": "Siidrikunsti meistriteos",
                  "year": "2016",
                  "category": "Digitaalne reklaam /
Kampaania/sotsiaalmeedia",
                  "applicant": "Imagine",
                  "award": "Silver",
                  "thumbnail": "thumbnails/2437.png",
                  "id": "2"
            }
      ]
}
```

Joonis 11: Projektide nimekirja JSON

Veebileht keskendub projektide vaatele ja kõige keskmes on pildid koos infoga, siis mõnikord ei piisa ainult pildist ja selle pealkirjast. Tegin juurde nupud, mis vahetavad ruudustik ja nimekirja asetuse vahel, et inimesel on võimalik ise valida, kummal viisil meeldib talle projektide nimekirja vaadata. Kasutajal on näha, millist varianti ta hetkel kasutab, näidates seda nuppu tumedamana. Ühendasin nupud sellise direktiiviga nagu `ng-include`, mis vahetab teatud elemendis HTML koodi, mis on võetud teisest HTML failist. Sellise viisi hea omadus on see, et ei koorma ühte faili tohutute ridadega.

3.4.3 Numeratsioon

Projekte on palju ja nende ühel lehel näitamine on kasutajale väga koormav. Nimekirjade lehekülgedeks nummerdamine on hea viis andmeid osadeks jaotada. Alustasin kohe tavalise JavaScripti kirjutamisega, kuid paljud Angulari arendajad

kirjutavad laiendusi, mida saab oma töös kasutada. Selleks tuleb alla laadida seotud failid ning Angulari moodulisse sõltuvusena kirja panna. Seejärel on võimalik oma koodis laiendust kasutada ja erinevaid parameetreid seadistada.

Nummerdamisel kasutasin Michael Bromley laiendust Pagination Directive [26]. Lisades koodi HTMLi, saab seal erinevaid atribuute määrata. Üheks atribuudiks on, mitu projekti näidatakse ühel lehel korraga. Ruudustik vaates valisin selleks 16 projekti lehe kohta, sest vähem näidates, peab kasutaja liiga tihti järgmisele lehele liikuma. Nimekirja vaates valisin kümme projektide lehe kohta, sest katsetades erinevate arvudega, oli see kõige mõistlikum lahendus. Olles nimekirjavaates, võtab üks projekt enda alla terve rea. Selletõttu saan näidata vähem projekte lehekülje kohta.

Kui palju numbreid näidatakse lehekülje numeratsioonil, määrasin kümme, sest vähesemaga on kasutajal liiga väike võimalus edasi või tagasi liikumiseks ning suurema puhul on liiga palju numbreid koormaks. Numbri määramisel vaatasin üldkasutatavaid lehekülgi, näiteks Google, millel samuti näidati numeratsioonis kümmet lehekülge. Lisaks käivitus lehekülje vahetamisel meetod, mis viis kasutaja sisupaneeli ülemisse ossa, sest jäädes samasse kohta, peab inimene üles kerima ja lehekülje vahetamiseks jälle alla tagasi tulema. Seetõttu, peab kasutaja ainult alla kerima saab edasi valida järgmise lehekülje. Selline lahendus teeb kasutaja elu kergemaks, kuna ta ei pea lisa liigutust tegema.

Leheküljel olles kasutaja võib tahta minna tagasi eelmise lehekülje juurde kasutades veebilehitseja tagasi nuppu. Alguses ei olnud seda võimalust liikudes läbi nummerdatud lehekülgede. Selleks, et oleks selline võimalus, tuleb probleem lahendada läbi URLi. Kui inimene valib näiteks järjekorras teise lehekülje, ise samal ajal olles esimesel, siis lehekülge vahetades saadetakse uus järjekorranumber URLi. URList saadakse number, millega määratakse pildid, mis on teisel leheküljel. Veebilehitseja jätab URLi muutused meelde ning seeläbi on võimalik tagasi nuppu vajutades minna eelmisele lehele.

Nummerdamisel on võimalik saada andmeid asünkroonselt. Hetkelisel näitel loetakse sisse terve JSON fail, kus on kõikide projektide andmed. Sellisel juhul võib veebileht muutuda aeglaseks kuna liiga palju andmeid on lugeda. Nummerdamise laienduses on olemas võimalus saada just selle lehekülje andmeid serverist, millele vajutatakse. Selline viis tagab vähemate andmete lugemise ja kiirema vastamise aja.

3.4.4 Projekti vaade

Projekti vaates näidatakse suurtena kõiki pilte ja videosid, mis antud projektiga on seotud. Piltide kõrval on eraldi sektsioon andmete jaoks, kus suurelt on projekti pealkiri ning selle all asuvad projekti täiendavad andmed. Projektiga seotud andmed võetakse kõik JSONist (Joonis 12). Andmed on eraldatud katkendliku joonega, et nendel vahet teha. Projekti andmete all on lingid, mis viivad eelmisele ja järgmisele projektile. Kui eelmist või järgmist projekti ei ole, siis linki ei kuvata. Kuna igal projektil võib andmete hulk erineda, siis võetakse need eraldi `details` hulgast, kus võti ja väärtus näidatakse mõlemad HTMLis välja.

```
{
      "project": [{
            "title": "Siidrikunsti meistriteos",
            "photos": ["photos/238.png"],
            "details": [{
                  "year": "2016",
                  "category": "Digitaalne reklaam /
Kampaania/sotsiaalmeedia",
                  "applicant": "Imagine",
                  "award": "Silver"
            }],
            "lastProjectId": "1",
            "nextProjectId": "3"
      }]
}
```

Joonis 12: Projekti andmete JSON

3.4.5 Kataloogipuu

Esimesena sai kasutusele võetud treeview direktiiv. Kataloogipuu paigutasin parempoolsele küljeribale, arvates selle pidev nägemine on hea silma peal hoidmiseks, kus parasjagu hierarhias asub. Treeview laiendusel oli viimane uuendus tehtud neli aastat tagasi ning lihtsate funktsioonide jaoks tuli palju lisa koodi kirjutada. Otsisin uue kataloogipuu süsteemi laienduse, milleks oli Angular Tree Control[27]. Uuega oli võimalik määrata, et kataloogi nimele vajutades see laieneks ehk tuleks nähtavale alamkataloogid või kitseneks ehk alamkataloogid läheksid peitu. Tekstile vajutades, mis samal ajal laieneb, oleks võimalus neid projekte vaadata, mis on kõikides

alamkataloogides. Kausta ees on ikoon, millele klõpsimine samuti vahetab laia ja kitsa vahel, kuid kasutajad ei pruugi kohe aru saada, et kaustad käivad lahti, mistõttu saab seda teha nime peale vajutades.

Kataloogipuu põhineb JSONil (Joonis 13). JSONis on olemas iga kausta `label` ehk pealkiri, mis kuvatakse kataloogipuus välja. Teiseks on igal kaustal olemas `id`, mida kasutatakse URLi määramisel ja andmebaasist projektide pärimisel. Kõikidel objektidel on olemas järglased, kuhu kuuluvad samuti kausta pealkiri ja id, millel võivad olla omakorda järglased. Kui `children` väli on tühi, siis tähendab see järglaste puudumist ning rohkem kaustu ei järgne. Kataloogipuus otsustasin projekte mitte näidata, sest see teeks puu liiga pikaks ja kohmakaks.

Veebilehe peamiseks eesmärgiks on edastada pilte ning kataloogipuu kõrval näitamine vähendab piltide vaatamise ruumi. Suuremat pilti on kasutajal parem vaadata, kuid vähese ruumi tõttu näidatakse pilte väiksemana. Selleks otsustasin parema küljepaneeli ära võtta ning asendada nupuga, mis kutsub külje pealt välja paneeli. See andis võimaluse pilte suuremalt näidata, kuid samas siiski kataloogipuud sirvida ja vastavat kausta valida.

Lisades algselt nupu „Vaata kataloogipuud” otsingu alla tähendas, et iga kord kui kasutaja tahab kataloogipuud vaadata, peab ta kerima täiesti lehe algusesse. Teiseks jääb nupp otsingu varju ning inimesed ei pruugi seda tähele panna. Määrates nupu fikseeritud koha peale lehekülje vasakul üleval nurgas, tõmbab see tähelepanu sellega, et see käib leheküljega kaasas. Kuna nupu tekst on jällegi liiga pikk ning jääb piltidele ette, siis otsustasin teksti asemele panna noole. Ainult nool ei ütle midagi, millega on tegemist, seega hiirega nupu peale minnes, tuleb nähtavale tekst, et on võimalik vaadata kataloogipuud.

```
{
	"label": "Home",
	"id": "home",
	"children": [{
		"label": "2016",
		"id": "k1",
		"children": [{
			"label": "PR",
			"id": "k11",
			"children": []
		}]
	}]
}
```

Joonis 13: Kataloogipuu JSON

Vajutades noolele, tuleb külje pealtvälja paneel, kus on kataloogipuu. Saades teada uurimusest, et inimesed vaatavad rohkem lehekülje vasakut poolt, sai paneel paigutatud ümber vasakule poole. Ülejäänud taust läheb tumedaks, et paneelile tähelepanu tõmmata. Väljuda on võimalik kasutades kõige üleval olevat noolt või vajutades ükskõik kuhu tumedale alale lehel. Tume ala tekitab inimeses tunde, et see on hetkel mitte aktiivne ja vajutades sinna peale, teeb selle osa aktiivseks.

3.4.6 Jäljerida

Igal leheküljel on sisupaneeli üleval osas jäljerida, mis näitab hierarhiliselt, kus kasutaja hetkel asub. Jäljerida on vajalik veebilehel, kui lehekülgi on palju ning need on hierarhiliselt üksteise sees. Reas olevad lingid viivad hetkel olevast kohast kõrgemale kausta kataloogipuus. Jäljerida on saadud kataloogipuu JSONist, kus iga lehekülje teekond on salvestatud uude JSONisse (Joonis 14). Igal leheküljel otsitakse JSONist lehekülje id abil üles objekt ning võetakse path muutujast objektid, mis lisatakse jäljeribale.

```
[{
      "id": "home",
      "label": "Home",
      "path": [{
            "id": "home",
            "label": "Home"
      }]
}, {
      "id": "k1",
      "label": "2016",
      "path": [{
            "id": "home",
            "label": "Home"
      }, {
            "id": "k1",
            "label": "2016"
      }]
}, {
      "id": "k11",
      "label": "PR",
      "path": [{
            "id": "home",
            "label": "Home"
      }, {
            "id": "k1",
            "label": "2016"
      }, {
            "id": "k11",
            "label": "PR"
      }]
}]
```

Joonis 14: Tee leheküljeni JSON

3.4.7 Disain

Peale funktsionaalsuse tegemist hakkasin tegelema disainiga. Aluseks võtsin Piccolo[28] malli, milles oli näiteid erinevate vaadete ja galerii struktuuri kohta, kuid minule vajaminevat täpselt ei olnud. Valisin parempoolse küljeribaga lehekülje, kuhu sisu asemele panin galerii ja kustutasin üleliigsed elemendid. Seejärel lisasin antud templatele olemasoleva funktsionaalsuse. Ka sai ära võetud taustapilt, mis tegi ekraani ruuduliseks ja vaatamise ebamugavaks.

Kuigi Piccolo mallil oli eelnevalt värvid olemas, siis muutsin värve vastavalt teooriale. Kasutasin eelnevalt mainitud analoog värviskeemi, kus on valitud värvid lähestikku. Värvideks sai lilla ja roosa, mida täiendasid veel valge ja hall. Roosat värvi sai kasutatud nuppude ja linkide välja toomisel ning lilla jäi rohkem teksti ja elementide taustavärviks. Valget ja halli sai kasutatud taustaks ning erinevate lehekülje osade eristamiseks.

Mallis oli algselt täisekraani vaatel suurtel piltidel kõik tekst näha, mis võis tekitada olukorra, kus pikkade pealkirjadega võis ruudustik vaade paigast nihkuda. Mulle väga meeldis, et kui pildid läksid väiksemaks, siis pikemad pealkirjad läksid peitu. See tagas paigutuse hea väljanägemise ja kõik kastid olid ühe suurused. Muutsin selle suurtel piltidel ära ning panin juurde võimaluse, et kui hiirega pealkirja peale minna, siis hüppab lahti kastike, kus on täies pikkuses näha pealkirja.

Iga projekti pealkirja kasti all on joon, mis aitab eristada ja piltide struktuurist aru saada. Nii ruudustik kui nimekirja vaates tuleb projekti kastile juurde vari, kui selle peale hiirega minna. Selline lahendus näitab inimesele, mis projekti peal ta hetkel on ning vajutusega minnakse õigesse kohta.

Kuldmuna arhiivis on olemas projekte, mis on saanud auhinnalise koha. Lisaks kullale, hõbedale ja pronksile, on lisaks eriauhinnad. Kirjutades ruudustik vaates pildi juurde tekstina auhinnaline koht, tekitab see juurde lisalugemist. Selle asemel leidsin visuaalse lahenduse ning panin teksti juurde tähe ikooni, mis vahetab värvi vastavalt, mis koht on saadud. Tähe märk on vaid nendel projektidel, mis on auhinnalise koha saanud, teistel täht puudub. Täht saab olla, kas kuldne, hõbedane, pronks või eriauhinnal sinine. Kui inimene ei ole kindel, mida see täht tähendab, hüppab hiirega peale minnes lahti kastike, mis ütleb, mis auhinnaga on tegemist.

3.5 Tehniline

Veebilehel on ainult index lehekülg, kuhu sisestatakse kõik kujunduse CSS ja JavaScripti failid sõltuvustena. Osad failid on lokaalsed ja osad võetakse interneti aadressidelt. Index lehel on raam terve veebilehe ulatuses sama, ainult vahetatakse komponente `ui-view` sees.

Veebileht on jagatud komponentideks, kus näiteks kataloogipuu on üks komponent kõigist olemasolevates. Igal komponendil on oma kontroller, mis on määratud `ng-controller`iga ning mida juhitakse script.js failist. Kontrolleris toimub andmete töötlus ja funktsionaalsus. Näiteks nupule vajutades kutsutakse välja funktsioon kontrolleris. Funktsioonis on realiseeritud, mis veebilehel juhtub.

Osades kontrollerites on vaja saada andmeid mujalt, näiteks serverist, selleks kasutatakse http päringut. Päringuga saadud tulemust on võimalik välja näidata vastava kontrolleri HTML komponendis. Erinevatel kontrolleritel võib vaja minna samu andmeid, kuid mõlemas kontrolleris eraldi päringut teha ei ole mõtet. Andmete jagamiseks kontrolleri vahel tuli teha teenus, kuhu on võimalik andmeid salvestada. Esimeses kontrolleris salvestatakse andmed teenusele ja teises kontrolleris küsitakse andmeid teenuselt.

Kuna tegemist ei ole väga suure rakendusega, siis ei ole sellel ka väga palju faile. Kuid failidel on siiski struktuur. Kõige alguses on index fail, millega läheb terve rakendus tööle. Samal kõrgusel on ka kõik HTMLi vahetatavad komponendid. Kaustas css asuvad kõik failid, mis on seotud veebilehe kujundamisega. Kaustas js asuvad kõik failid, mis tagavad veebilehe funktsionaalsuse ning img kaustas on pildid, mida kasutatakse veebilehel ikoonidena.

3.6 Uue kasutajaliidese võrdlemine olemasolevaga

Pealehele sisenedes on erinevus juba selles, et uuel veebilehel on pildid palju suuremad ning täidavad lehe. Olemasoleval jääb paremalt poolt ruumi kasutamata ning pildid on väikesed. Kataloogipuu vaatamiseks peab alati ülesse kerima, kuid uuel on alati võimalus ühe nupuvajutusega seda näha.

Olemasoleval veebilehel on palju vaba ruumi üleval osas ning süvendatud otsing on kogu aeg nähtaval, mis võtab projektidelt ruumi ära. Uuel kasutajaliidesel on võimalus täpsemat otsingut avada ja sulgeda, mis tagab põhirõhu piltide vaatamisele. Samuti on olemas üldine otsing, mis tähendab, et kasutaja ei pea teadma kindlalt, mis lahtrit täita.

Ruudustik ja nimekirja vaate vahel valides, ei jäeta vanal kasutajaliidesel kasutaja valikut meelde. Valides kõrvalt kataloogipuust uue kataloogi, näidatakse alati nimekirja vaadet. Jäljerealt ei ole võimalik eelmistele vaadetele tagasi minna. Kuid uuel on võimalik linkidele vajutades vaadata ülemkatalooge.

4 Kokkuvõte

Hea kasutajaliides ja kasutajakogemus annavad inimestele paremaid kogemusi. Need on tähtsad, et inimesed saaksid kiirelt ja mugavalt oma toimetused ära teha. Seetõttu tasub teha veebilehed sarnase struktuuriga, et inimesed ei peaks ümber õppima ja saaksid seda kasutada kiirelt, ilma aega raiskamata.

Lõputöö käigus sai uuritud, mis on kasutajaliides ja kasutajakogemus ning miks need on tähtsad. Sain teada, et kasutajad üldiselt vaatavad veebilehti F-kujulise mustrina ning ilus kasutajaliides töötab paremini. Lisaks uurisin, kuidas teha ilusat disaini, mida peab jälgima paigutusel, milliseid värve kasutada ja mida erinevate elementide juures meeles pidada.

Teises osas rääkisin veebilehe arendamisest. Alustuseks koostasin nõuded, tegin mockupi. Valisin välja raamistiku, millega veebilehte hakkasin arendama. Valmistasin Angulariga kasutajaliidese funktsionaalsuse ja disaini määrasin CSSiga. Erinevate komponentide, näiteks kataloogipuu ja nummerdamise laienduste, juures analüüsisin erinevate valikute üle.

Lõputöö eesmärgiks oli luua kasutajaliides Kuldmuna arhiivile, mis näeks tänapäevane välja. Kasutajaliidese tegemisel toetusin teooriale ja vastava lehekülje omapäradele. Töö tulemusena valmis veebilehe kasutajaliides, kus on projektide nimekirja vaade, projekti vaade, kataloogipuu ja otsing. Antud töös realiseeriti mugav kasutajatuge toetav baasfunktsionaalsus kui ka disain.

Kasutatud kirjandus

[1] Total number of websites [WWW] http://www.internetlivestats.com/watch/websites/ (15.04.2017)

[2] User interface (UI) [WWW] http://searchmicroservices.techtarget.com/definition/user-interface-UI (27.04.2017)

[3] User interface [WWW] https://techterms.com/definition/user_interface (27.04.2017)

[4] International Organization for Standardization [WWW] https://www.iso.org/obp/ui (29.04.2017)

[5] Winter, J. What is User Experience? [WWW] https://www.usertesting.com/blog/2015/08/13/what-is-user-experience/ (29.04.2017)

[6] Garret, J. J. The Elements of User Experience. 2nd ed. Berkeley : Peachpit, 2011.

[7] Group XP. Experience index 2016 [WWW] http://cdnwww.group-xp.com/pdfs/Group_XP_Experience_Index_2016_WEB6.pdf PDF (11.05.2017)

[8] Galitz, W. O. The Essential Guide to User Interface Design. 2nd ed. New York : John Wiley & Sons, Inc., 2002.

[9] Stanford, J., Tauber, E. R. How Do People Evaluate a Web Site's Credibility? [WWW] http://www.jsuccba.com/feather/375/HowDoPeopleEvaluateWeb%20SiteCredibility.docx (12.05.2017)

[10] Norman, D., Emotional Design. Why we love or hate everyday things, New York : Basic Books, 2004 .

[11] Web UI design best practices [WWW] http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/UXPIN_PL/U141030B.pdf (12.05.2017)

[12] Andrade, M. R., The Gutenberg Diagram in Web Design [WWW] https://medium.com/user-experience-3/the-gutenberg-diagram-in-web-design-e5347c172627 (14.05.2017)

[13] Nielsen, J. F-Shaped Pattern For Reading Web Content [WWW] https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/ (14.05.2017)

[14] Nielsen, J. Horizontal Attention Leans Left [WWW] https://www.nngroup.com/articles/horizontal-attention-leans-left/ (14.05.2017)

[15] Kucheriavy, A. [WWW] https://www.intechnic.com/blog/100-ux-design-pro-tips-from-user-experience-master/ (05.04.2017)

[16] Interface Design [WWW] http://www.webstyleguide.com/wsg3/4-interface-design/3-interface-design.html (11.05.2017)

[17] UX/UI Best Practices: 125 Easy Tweaks to Optimize Your Website's Usability [WWW] https://www.nickkolenda.com/user-experience/ (26.04.2017)

[18] Visual Hierarchy, UI Design Tips To Create A Great Web User Interface [WWW] https://visualhierarchy.co/blog/ui-design-tips-to-create-a-great-website/ (12.05.2017)

[19] Web UI Design for the Human Eye [WWW] https://academy.anetwork.ir/content/uploads/2016/07/Web-UI-Design-for-the-Human-Eye-%E2%80%93-Part-1.pdf (16.05.2017)

[20] Cannon, T. An Introduction to Color Theory for Web Designers [WWW] https://webdesign.tutsplus.com/articles/an-introduction-to-color-theory-for-web-designers--webdesign-1437 (16.05.2017)

[21] What is functional and non functional requirement? [WWW] http://stackoverflow.com/questions/16475979/what-is-functional-and-non-functional-requirement (01.05.2017)

[22] Framework [WWW] http://whatis.techtarget.com/definition/framework (07.05.2017)

[23] Andras, S. JavaScript Frameworks, why and when to use them [WWW] https://blog.hellojs.org/javascript-frameworks-why-and-when-to-use-them-43af33d0608d (07.05.2017)

[24] Hannah, J. Choosing the Right JavaScript Framework for the Job [WWW] https://www.lullabot.com/articles/choosing-the-right-javascript-framework-for-the-job (07.05.2017)

[25] AngularUI Router [WWW] https://github.com/angular-ui/ui-router (04.05.2017)

[26] Pagination Directive, Bromley, M. [WWW] https://github.com/michaelbromley/angularUtils/tree/master/src/directives/pagination (23.04.2017)

[27] Angular Tree Control [WWW] https://github.com/wix/angular-tree-control (06.05.2017)

[28] Piccolo – A Free Bootstrap HTML Theme [WWW] https://wegraphics.net/downloads/piccolo-a-free-bootstrap-html-theme/ (27.04.2017)