TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Elina Antonova 179010IADB

INFOTAHVLI TIMEABLE ARENDUS

Bakalaureusetöö

Juhendaja: Meelis Antoi

Magistrikraad

Tallinn 2020

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: Elina Antonova

15.02.2020

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua töötav Timeable veebirakendus Tallinna Tehnikaülikooli IT Kolledžis paiknevatel televiisoritel tunniplaani kuvamiseks selle kiireks ja mugavaks kättesaamiseks ilma nutiseadet kasutamata. Rakenduse loomise eesmärgiks on automatiseerida tunniplaani kuvamise protsess ning vähendada võimalikke kulusid. Rakendus on avatud lähtekoodiga ning luuakse mõeldes mõeldes võimalikele muudatustele tulevikus.

Arendusprotsessi käigus luuakse veebirakendus, mis automatiseerib tunniplaani allalaadimise protsesse ning võimaldab ekraani vaadete haldusvõimalust: sisseloginud kasutajatel on võimalik luua, muuta ja kustutada ekraane või lisada, muuta ja kustutada üritusi ning loenguid tunniplaanis ja lisada ürituste reklaami. Arendus on jagatud mitmesse ossa: rakenduse loogika ja vaadete arendus ning testimine.

Antud töö rõhuasetus on kasutatavatel tehnoloogiatel: nende kaasaegsus, kasutusmugavus, kasutajabaas ja antud rakenduse skoobi sobivus. Töös on kaetud veebiteenuse ja kliendiliidese arendusprotsess ja testimine. Arendusprotsessi tulemuseks on töötav rakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 73 leheküljel, 5 peatükki, 21 joonist, 1 tabeli.

Abstract

Information Board Timeable Development

The aim of current thesis is to create Timeable web application for the screens located in IT College of Tallinn University of Technology for quick and convenient access to timetable without any need to use a smart device. The purpose of creating the application is to automate the process of obtaining a schedule and reduce potential costs. The application's source code is open and is designed with possible future changes in mind.

During the development process, a web application is created that automates the schedule download process and provides the ability to manage screen views: logged-in users can create/edit/delete screens, add/edit/delete events and lectures in the schedule, add advertisements. Development is divided into several parts: application logic and views development, testing.

The emphasis is on the technologies used: their modernity, ease of use, user base, suitability to the scope of the given application. The thesis covers the development process of the web service and user interface, testing process. The result of the development process is a working application.

The thesis is in Estonian and contains 73 pages of text, 5 chapters, 21 figures, 1 table.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* ehk Rakendusliides
BSON	*Binary JavaScript Object Notation* – binaarne Javascript'il põhinev andmevahetusvorming
CLR	*Common Language Runtime* ehk Microsofti käituskeskkond .NET-platvormi jaoks
CRUD	*Create-Read-Update-Delete* ehk Loo-Loe-Muuda-Kustuta
CSS	*Cascading Style Sheets* ehk kaskaadlaadistik
DOM	***D**omain **O**bject **M**odel* ehk dokumendi objektimudel
DTO	*Data Transfer Object* ehk andmeedastusobjekt
HTML	*HyperText Markup Language* ehk hüperteksti märgistuskeel
JSON	*JavaScript Object Notation* ehk Javascript'il põhinev andmevahetusvorming
MVC	*Model-View-Controller* ehk Mudel-Vaade-Kontroller
ORM	*Object-Relational Mapping* – objekt-orienteeritud koodi kaardistamine teise andmevormi
REST	*Representational State Transfer* - veebiteenusega suhtlemise liidese arhitektuur
SQL	*Structured Query Language* ehk struktureeritud päringukeel
SQL/PSM	*Structured Query Language/Persistent Stored Modules* - struktureeritud päringukeel/püsivad salvestatud moodulid
sudo	*substitute user and do* ehk UNIX-süsteemiga arvutite käsk, mis annab mõned administraatoriõigused mitte-administraatoritele
UOW	*Unit Of Work* ehk tööühik
ÕIS	Õppeinfosüsteem
XML	*eXtensible Markup Language* - laiendatav märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva lõputöö raames luuakse veebirakendus värske tunniplaani kuvamiseks Tallinna Tehnikaülikooli IT Kolledžis. Eesmärgiks on töötav rakendus, mis automaatselt kontrollib tunniplaani muudatusi õppeinfosüsteemis ning värskendab tunniplaani IT kolledži ekraanide peal.

Probleemiks osutus värskete tunniplaani muudatuste kiire ja mugav kättesaamine. Käesoleval hetkel tunniplaan ja selle muudatused on saadaval õppeinfosüsteemi kaudu, kust info leidmiseks läheb vaja interneti ühendusega seadet ning otsimiseks kulub aega. Nii õpilastel, kui ka õppejõududel võivad puududa nii internetiühendusega seade, kui ka aeg, aga IT kolledžis asuvad ekraanid teevad tunniplaani kättesaadavuse mugavamaks.

IT Kolledži ja Tallinna Tehnikaülikooli ühendamisega muutus kasutatav õppeinfosüsteem. Vana IT Kolledži õppeinfosüsteem oli seotud majas asuvate televiisoritega ning muutused õppeinfosüsteemis kajastusid ekraanidel. Peale õppeasutuste ühendamist ning ÕISi (õppeinfosüsteemi) muutmist leiti alternatiivne lahendus.

Teema valimise hetkel oli kasutusel Risevision süsteem, mis võimaldab käsitsi lisada tunniplaani, kasutades Google Sheets ning kuvada seda valitud ekraani peal. Kõik seadistused ning tunniplaani Google Sheets lehele lisamine toimub igapäevaselt käsitsi IT Kolledži multimeediaspetsialisti poolt. Revision lahenduse kasutusele võtmise hetkel oli Google Sheets ühendamine süsteemiga ja ekraanide seadistus tasuta, kuid käesoleval hetkel kõik need võimalused muutusid tasuliseks. Vaatamata sellele, et multimeediaspetsialistil kulub igapäevaselt aega käsitsi tunniplaani ÕIS-ist mahakirjutamisele, toob Revision süsteemi kasutamine ülikoolile lisakulusid.

Ülaltoodud probleemide elimineerimiseks ehitatakse bakalaureusetööna rakendus, mis automaatselt parsib õppeinfosüsteemis kehtiva tunniplaani ning uuendab kasutaja vaadet. Selle lahendusega muudetakse multimeediaspetsialisti tööpäev tõhusamaks, õpilastele ja

õppejõududele vajaliku info kiire kättesaamine mugavamaks ning vabastatakse ülikool lisakuludest.

1.1 Metoodika

Antud lõputöö raames esmalt kirjeldatakse lõputöös lahendatav probleem, selgitatakse, miks eksisteerivad süsteemid on puudulikud ning ei sobi lahenduseks ja pakutakse analüüsi alusel sobiv IT lahendus, mis jääks lõputöö skoopi ning oleks edasiarendatav.

Lõputöö teises osas käsitletakse loodud kasutajagruppe ning nende nõudeid arendatava programmi funktsionaalsuse suhtes. Analüüsitakse ning valitakse sobivat tehnoloogiat rakenduse arendamiseks, andmebaasisüsteemi, arenduskeskkonna ning halduskeskkonna jaoks. Vastavalt kasutajagrupi nõuetele luuakse rakenduse arhitektuur ning kirjeldatakse rakenduse kasutajakogemuse disain.

Rakenduse arenduse protsessi on kirjeldatud nii serveripoolselt, kui ka kasutajaliidese poolt. Samuti on kirjeldatud kihtide kaupa rakendusse valitud tehnoloogiate kasutamine. Hinnatakse lõputöö raames valminud rakenduse kvaliteeti ja funktsionaalsust ning hinnatakse võimalust süsteemi edasiarenguks, mis jääb töö skoobist väljapoole.

2 Probleemi ülevaade

Tallinna Tehnikaülikooli IT Kolledži infotahvleid kasutatakse igapäevaselt tunniplaani ning toimuvate ürituste informatsiooni kuvamiseks. IT Kolledži iga korrus (v.a. esimene korrus) on varustatud ühe ekraaniga, kus kuvatakse tunniplaan, kuupäev ning kellaaeg. Esimesel korrusel asub kaks ekraani ning ühte nendest tavaliselt kasutatakse toimuvate ürituste kohta informatsiooni kuvamiseks. Infotahvleid kasutatakse igapäevaselt üliõpilaste, õppejõudude ja külaliste poolt, kiirelt ja mugavalt tunniplaanist informatsiooni leidmiseks: loengutest (nende toimumise aeg, koht ja läbiviija), millal õppejõud on hõivatud ning kust teda saab leida, millised ruumid on vabad/hõivatud päeva jooksul erinevatel kellaaegadel. Lõputöö rakenduse loomise hetkel on kasutusel Risevision rakendus.

Infotahvli Timeable rakenduse loomise idee tekkis otsesest vajadusest ning oli välja pakutud IT Kolledži multimeediaspetsialisti poolt. Arutelu käigus selgus, et olemasolevad lahendused ei rahulda IT Kolledži nõudeid ning vajavad palju manuaalset tööd. Tunniplaani saamiseks kasutatakse õppeinfosüsteemi, kust informatsiooni kättesaamine on keeruline kasutavate tehnoloogiate pärast. Informatsiooni saamine on võimalik, kas manuaalselt või HTML (*HyperText Markup Language)* parsimisega, mis pole toetatud ekraanidele informatsiooni kuvavates programmides ning tunniplaani saamise protsess pole automatiseeritud.

Lisaks suure hulga manuaalse töö tegemise vajadusega, puudub kasutataval tasuta versiooni rakendusel võimalus programmi konfigureerida: puudub tasuta võimalus lisada uusi ekraane, muuta kasutatavate komponentide paigutust või muuta informatsiooni kättesaamise viisi.

2.1 Eksisteerivad lahendused

Kuna tegemist on sarnaste lahendustega, siis paljud erinevused eksisteerivate süsteemide ning lõputöö raames arendatud lahendusega võib üldisemalt kirjeldada kõikide lahenduste kohta. Järgnevates lõikudes võrreldakse eksisteerivate süsteemide ja Timeable rakenduse

erinevusi: eelised ja puudused. Nende süsteemide töönõuded on kitsapiirilised ning kirjeldatud erinevused katavad ainult tunniplaani ekraanile kuvamise töövoogu.

Eksisteerivate rakenduste arendamise taga on terved meeskonnad ning rakenduse parendamine toimub kiiremini. Timeable ei paku sellist kiirust, aga see oli loodud vastavalt TalTech IT Kolledži nõuetele ja on personaliseeritud vastavalt tellija soovidele. Lisanõuete tekkimisel on Timeable rakenduse edasiarendus lihtne, kuna rakenduse suurus on väike ning kood on avalik.

Kõik alternatiivid Risevision süsteemile töötavad samal põhimõttel. Lahendusega on ühendatud Google Sheets lehekülg, kuhu andmete sissekandmine toimub manuaalselt IT Kolledži multimeediaspetsialisti poolt. Võrreldes sellega, Timeable rakendus võtab informatsiooni automaatselt õppeinfosüsteemist, koos kõikide viimaste muudatustega kasutades HTML parsimist.

Eksisteerivad rakendused on võrgurakendused ning Internetiga ühenduse kaotamisel nende süsteemide kättesaamine on võimatu. Timeable töötab lokaalselt ning selle tõrgeteta tööaeg sõltub otseselt lokaalse võrguketta tööajast. Tunniplaani uuendust tehakse kord päevas ja selle jaoks on vaja Interneti ühendust. Kõik teised teenused on seotud ainult lokaalse võrguketta ülalhoidmisega.

Ülaltoodud erinevused on ühised kõikide alternatiivsete lahenduste puhul. Suuremaks erinevuseks on nende lahenduste hind, mis on toodud iga süsteemi puhul eraldi.

2.1.1 Risevision

Risevision on digitaalsete siltide halduslahendus, mille arendusega aastast 1992 tegeleb Ameerika Ühendriikide ettevõte Rise Holdings Inc. Risevision rakendus on peamiselt loodud õppeasutuste jaoks: kuulutuste, informatsiooni ja tunniplaani kuvamiseks. 28. veebruari 2020 seisuga kasutas seda rohkem kui 3000 õppeasutust [1]. Selle peatüki järgmistes lõikudes on kirjeldatud Rivevision rakenduse ja lõputöö raames arendatud Timeable rakenduse erinevused.

Risevision kasutab kasumi teenimiseks tellimussüsteemi. Kasutaja maksab iga ekraani ühiku lisamise eest ning keskmiselt maksab igakuiselt iga ekraani kasutamine õppeasutusele 10$.

Kuna IT Kolledžis asub 6 ekraani, siis igakuiselt Tallinna Tehnikaülikool peab maksma 49,5$ kuni 54$ [2]. Timeable rakenduse kasutamise kulud on kaudsed: elektrikulu, kõvaketta kulu, halduskulu, jne.

2.1.2 CloudShow

CloudShow on digitaalsete siltide halduslahendus, mis loodi 2007. aastal asutatud Kanada ettevõte Binary Fortress Software poolt [3] [4]. Selle peatüki järgmises lõikudes on kirjeldatud CloudShow, Risevision ja Timeable rakenduste erinevused.

CloudShow rakendus võimaldab samuti ühendada Google Sheets süsteemi tabeleid ning võrreldes teiste võimalustega (pildid, Power Point presentatsioon, PDF-failid) oleks optimaalsem. Kuid nii nagu Risevision süsteem, vajab selline lahendus palju manuaalset tööd info sisestamiseks, mida Timeable rakenduse puhul automatiseeriti.

Võrreldes Risevision süsteemiga, vajab CloudShow vähem rahalist kulu. Igakuiselt kuue ekraani eest tuleb tasuda 36$ kuni 40$ olenevalt valitud maksestrateegiast. Timeable puhul ei ole kulu nii suur.

2.1.3 Yodeck

Yodeck on samuti digitaalsete siltide halduslahendus. Yodeck on loodud Ameerika Ühendriikides Flipnode LLC poolt ning selle arendus alustati 2012. aastal [5]. Yodeck lahenduses kasutatakse Raspberry Pi arvutit ning pilveteenust ennast, kuid teiste funktsioonidega süsteem on sarnane varem kirjeldatud süsteemidega.

Samuti nagu ülaltoodud süsteemid pakub Yodeck erinevaid tellimustariife. Arvestades IT Kolledži nõudeid oleks piisav kuuele ekraanile tunniplaani kuvamine, mis maksaks 47,94$ kuus ning oleks soodsaim lahendus. Lisaks sellele on Yodeck süsteemi kasutamiseks vaja hankida 6 Raspberry Pi arvutit, aga Flipnode LLC pakkub neid tasuta tingimusel, et kliendiga vormistatakse aastane tellimus. Sellised kulud oleksid Risevision süsteemiga sarnased, kuid ei parandaks multimeediaspetsialisti töövoogu võrreldes Timeable süsteemiga.

2.2 Timeable lahenduse skoop

Eksisteerivate lahenduste suuremaks puuduseks on tunniplaani manuaalse mahakirjutamise vajadus. Esiteks, selline töövoog vajab suurt ajaresurssi multimeediaspetsialisti igapäevasest tööajast - umbes 30 minutit igal hommikul. Töövoo optimiseerimiseks on loodud lisa Google Sheets tabel, kuhu on sissekantud ainete nimetused ja läbiviivad õppejõud. Vajaliku aine kohta informatsiooni saamine toimub aine koodi järgi. Kuid selleks, et igapäevaselt lisada tunniplaani infotahvlile tuleb teha järgmised tegevused: avada ÕIS, avada tunniplaani vaade, avada otsing ning sisestada IT Kolledži kood, kuvada tunniplaan iga õppevormi jaoks ning sisestada informatsioon (alguse/lõpu kellaaeg, toimumise koht, ainekoodi järgi leida ühendatud failist loeng) ja nii iga loengu kohta eraldi.

Üks lahendus manuaalse töö asendamiseks oleks lisaprogramm, mis võtaks automaatselt informatsiooni ÕIS-ist ning sisestaks seda Google Sheets faili. Selline programm saab automaatselt HTML-i parsimisega iga õppevormi jaoks tunniplaani ja salvestab lokaalselt andmebaasi. Sellise automatiseerimisega lahendatakse veel üks probleem: vea tekkimine aine nimetuses/koodis/õppejõu nimes, kuna kõik info saamise ja kuvamise töö on tehtud masina poolt. Selline programm lahendaks protsessi automatiseerimise probleemi ja vähendaks inimfaktoriga seotud vigade hulka, kuid ei tooks kaasa kulude vähendamist.

Ülaltoodud probleemide tõttu kulude vähendamiseks ja kliendi soovide arvestamiseks pakub antud töö autor luua Timeable digitaalsete siltide halduslahendus. Üheks programmi osaks oleks ülalkirjeldatud informatsiooni saamise alamprogramm. Tunniplaani otsimisel kasutatakse IT Kolledži eesliidet, seal toimuvate loengute leidmiseks kõikide loengute nimekirjast. Selline võimalus annab kasutada Timeable süsteemi ka teistes Tallinna Tehnikaülikooli majades eesliite vahetamisel. HTML-i parsimise koodi väljavahetamisel või välis API (*Application Programming Interface)* töötlemist kasutamisele võtmisel ka teistes õppeasutustes, kuid see jääb antud lõputöö skoobist väljaspoole aga jätab ruumi edasiarenduseks.

Võrreldes Risevisioni vaatega muutub Timeable kliendi vaade paremini loetavamaks ja dünaamilisemaks ning administraatorile antakse rohkem võimalusi ekraanide konfigureerimiseks. Lisandub ka funktsionaalsus täiendada tunniplaani toimuvate üritustega

ning kuvada ekraanile teateid. Kuna selle süsteemiga soovitakse saavutada olemasoleva rakenduse edasiarendust piiritletult, siis arvestatakse tellija ehk multimeediaspetsialisti ja töö autori nõudeid ja soove. Timeable rakendus ei käsitle üldisemat siltide halduslahendust ning on piiratud ainult tunniplaani ja toimuvate ürituste informatsiooni edastamisega ning rakenduse administraatori poolt selle konfigureerimisega.

Lõputöö kontekstis piirdutakse vaid brauseri põhise lahendusega. Eelkõige selle tõttu, et lokaalses võrgus paigutatud rakendus on saadaval igast ekraanide külge ühendatud arvutist vaatamata operatsioonisüsteemile ning rakendus kuvab ühesugust informatsiooni igal ekraanil. Selline lahendus annab võimaluse seadistada ühesugused ekraani sätted sisevõrgus igast arvutist või kasutades virtuaalse privaatvõrgu lahendust ekraani seadistamiseks väljaspool sisevõrku.

Antud lõputöö raames arendatava lahenduse eesmärgiks pole raha teenimine, vaid õpilaste, õppejõudude ja IT Kolledži külastajate igapäevaselt vajaliku informatsiooni kättesaamise lihtsustamine. Arvestades seda, et projekti kood jääb avalikuks on võimalus tulevikus lisada funktsionaalsust või muuta programmi koodi (näiteks teistes õppeasutustes kasutamiseks).

3 Timeable rakenduse analüüs

3.1 Rakenduse nõuded

Kirjeldatud rakenduse nõuded katavad antud lõputöös määratud skoopi. Nõuete määramisel olid arvesse võetud eelkõige IT Kolledži multimeediaspetsialisti soovid ja töö autori täiendused.

Nõuded on kirjeldatud kasutaja lugudega ning grupeeritud omakorda epikutesse. Kasutaja lugude rollideks on rakenduse administraator, peaadministraator ning rakenduse kasutaja. Administraatori rollis lõputöö kirjutamise hetkel on IT Kolledži multimeediaspetsialist. Kasutajateks on õppejõud, õpilased ja külalised, kes vaatavad tunniplaani IT Kolledži korrustel asuvate televiisorite pealt. Kasutajalood kirjeldavad nii rakenduse funktsionaalsust, kui ka visuaalset aspekti.

Nõuetega on võimalik tutvuda Lisas 1.

3.2 Tehnoloogia valik

Kaasaegne tehnoloogiate valik veebirakenduse ehitamiseks on suur [6]. Sisuhaldustarkvara tehnoloogiad polnud käsitletud antud lõputöö raames, kuna nende abil ei saa lahendada lõputöös püstitatud probleeme ning nad ei sobi lõputöö skoopi. Timeable rakenduse ehitamiseks sobivate tehnoloogiate valimisel on vaadeldud kaasaegseid, suure kasutajabaasiga tehnoloogiaid, mille arendamise protsess pole lõppenud.

3.2.1 Serveri-poolne programmeerimiskeele valik

Raamistiku valik sõltub esiteks programmeerimiskeele valikust, kuna raamistikud põhinevad teatud keeltel. Allpool on toodud tuntud veebirakenduste programmeerimiskeeled, mis olid käsitletud ülikooli õpingute ajal [7]:

- PHP – üldotstarbeline objektorienteeritud skriptimiskeel, mida kasutatakse serveripoolsetes lahendustes dünaamiliste veebilehtede loomiseks. Keel on populaarne algajate seas ning selle võib lihtsalt integreerida HTML-i [7] [8].
- Java – üldotstarbeline objektorienteeritud programmeerimiskeel, arendatud Sun Microsystems poolt. Java on avatud lähtekoodiga keel, mis on platvormist sõltumatu. Java abil on võimalik arendada suuri veebi projekte [7].
- JavaScript – objektorienteeritud skriptimiskeel, mida võib kasutada objektorinteeritud, protseduuraalseks või funktsionaalseks programmeerimiseks. JavaScript on populaarseim programmeerimiskeel aastal 2019 ning Node.js käitussüsteem on enimkasutatud tehnoloogia veebi arenduses [9]. JavaScript sobib nii serveripoolseks, kui ka klientrakenduse arenduseks.
- C# - üldotstarbeline objektorienteeritud programmeerimiskeel, arendatud Microsoft-i poolt. C# sobib nii Windows programmide, mängude, kuid ka veebi arenduseks. C# on kõrge koormustaluvusega keel, sobilik ka keerukamate veebirakenduste jaoks, mida jooksutakse platvormidel .NET keskkonna toetusega. [7] [10].
- Python – aspektipõhine programmeerimiskeel. Veebi arendamisel kirjutatakse mooduleid, mida kontrollitakse ja seostatakse sõltuvalt lõppkasutaja otsustest kasutajaliidese poolel [11].

Serveripoolse programmeerimiskeele võrdlemisel on välja toodud töö autori kogemus ning selle keele õppimise keerukus arvestades varasemat kogemust. Iga keel omab põhjalikku dokumentatsiooni ning suurt kasutajabaasi. Järgnevalt on välja toodud keelte võrdlus tabelina (Tabel 1).

Tabel 1. Teenusepoolsete programmeerimiskeelte võrdlus.

Keel	**Kogemus**	**Õppimise keerukus**
PHP	Nõrk	Keskmine [12]
Java	Hea	Keskmine [13]
JavaScript	Keskmine	Madal [14]
C#	Väga hea	Madal
Python	Nõrk	Madal [15]

Arvestades, et uue keele õppimine on aeganõudev, aga lõputöö kirjutamiseks ja rakenduse arenduseks on piiratud aeg, siis mõistlikum lahendus oleks valida keeled, kus lõputöö autor omab rohkem kogemust. Nendeks keelteks antud hetkel on C# ja Java.

C# tundub nendest kõige sobilikum, kuna töö autori jaoks on see olnud lõputöö kirjutamise hetkel, juba aasta töökohal arenduseks kasutatavaks keeleks ning selle keelega seotud teadmiste täiendamine pole keeruline. Sarnaselt Java-ga omab C# paketihaldurit – NuGet [16], mille eeliseks võib pidada graafilise liidese kasutamise võimalust ning pidevat pakettide arendamist. Java analoogiks on Gradle või Maven, mis omakorda tegelevad ka projekti ehitamisega. C# keeles projektide ehitamiseks kasutatakse MSBuild, mis on .NET raamistiku osaks [17]. Lisaks sellele arenduskeskkonnad on tudengite jaoks tasuta ja C# keel oleks mõistlikum valik, arvestades ülaltoodud punkte.

3.2.2 Arvutitarkvara raamistiku valik

Kuna serveripoolse keelena on valitud C#, siis raamistiku valik on järgmine: .NET Framework, .NET Core ja Xamarin. Xamarin on avatud lähtekoodiga raamistik Android ja iOS rakenduste programmeerimiseks. Kuna lõputöö skoobiks on just veebi rakenduse programmeerimine, siis valikut tuleb teha .NET Core ja .NET Framework-i vahel.

.NET Framework on arvutitarkvara raamistik, mis loodi 2002. aastal Microsoft-i poolt. .NET Framework on mõeldud erinevate rakenduste ehitamiseks, mille jooksutamine on mõeldud Windows operatsioonsüsteemidega arvutites. Viimaseks raamistiku versiooniks antud hetkel (22.03.2020) on 4.8 [18].

.NET Core on arvutitarkvara raamistik, mis loodi 2016. aastal .NET Foundation-i poolt [19]. .NET Core on edasiarendus .NET Framework raamistikule ning selle peamisteks erinevuseks on platvormist sõltumatus ning raamistiku uuenduste toimumine NuGet paketihalduri abil, mitte Windows uuendustega. Viimane raamistiku versioon antud hetkel (22.03.2020) on 3.1 [20].

Mõlemad raamistikud pakuvad võimalust ehitada veebi rakendust ASP.NET Core raamistiku abil. .NET Framework ASP.NET rakendus peab olema paigaldatud Windows operatsioonsüsteemiga serverisse, kuid .NET Core ASP.NET võib olla paigaldatud

Windows, Linux ja macOS süsteemides [21]. Valiku tegemisel meie eeliseks on võimalus panna rakendus tööle serveril sõltumata selle operatsioonsüsteemist.

2019. aastal mais Microsoft teatas, et .NET Core järgmine versioon on .NET 5, kuhu ühendatakse .NET Framework (WPF, Windows Forms. ASP.NET Core), Xamarin ja .NET Core (ASP.NET Core, UWP) [22]. .NET Framework 4.8 on viimane suurem pikkaajalise toega uuendus ning selle raamistiku uuendused tulevikus on väiksemad ning tulevad välja 4.8.x numbritega [23]. .NET Core versioon 3.1 on samuti pikkaajalise toega, kuid selle uuendus .NET 5 peale peab toimuma lihtsamini Microsoft arendajate sõnul [24]. Arvestades raamistike paindlikust tulevastele uuendustele .NET Core raamistiku kasutamine tundub olema parem otsus.

Arvestades .NET Core ja .NET Framework ASP.NET Core erinevust ning .NET uuenduste plaane võib langetada otsust, et Timeable rakendust ehitatakse kasutades .NET Core raamistiku. Versiooniks valitakse 3.1, kuna ta on pikkaajalise toega ning selle uuendamine .NET 5 peale on lihtsam.

3.2.3 Raamistiku valik

Kuna arvutitarkvara raamistikuks oli valitud .NET Core, siis serveripoolse raamistikuks võetakse ASP.NET Core. ASP.NET Core on ümbertehtud ASP.NET 4.x, mida võib kasutada platvormist sõltumata veebi rakenduste ehitamiseks. Ta on suurema jõudlusega, kasutab .NET Core CLR (*Common Language Runtime*), kliendivaadete arendamiseks Razor Pages [25], mis on leheküljele orienteeritud raamistik dünaamiliste, andmepõhiste veebisaitide loomiseks [26].

.NET Core on valitud arvutitarkvara raamistikuks rakendusmudeli jaoks, kuid baasklassi projektide raamistikuks oli valitud .NET Standard. .NET Standard on ametlik .NET API-de spetsifikatsioon, mille valik teeb võimalikuks muuta rakendusmudeli raamistiku .NET Framework või Xamarin peale ilma rakenduse põhiloogika muutmata [27]. Selline lahendus muudab rakenduse paindlikuks ning lihtsustab kasutajaliidese väljavahetamist. Kuna rakendusmudeli jaoks oli valitud .NET Core 3.1 versioon, siis vastavalt sellele .NET StandarD versiooniks valitakse 2.1 [28].

3.2.4 MVC või Web API

ASP.NET Core raamistik võimaldab ehitada rakendusi, mis kasutavad MVC (*Model-View-Controller*) mustrit, veebi API-t või mõlemad.

Timeable rakendus on esialgu mõeldud veebibrauseris kasutamiseks ning pole käesoleval hetkel mõeldud muul viisil kasutamiseks. Rakenduse funktsionaalsus on kasutajaliidese-keskne, ehk HTML-i laadimisele keskendunud. Antud juhul kliendi ja serveri vaheliseks suhtlemiseks REST (*Representational State Transfer*) API kasutamine võib tõsta rakenduse loomise raskust ja suurendada kulutatud aega, lisaks rakendada antud hetkel üleliigset funktsionaalsust [29]. Kuna MVC mustri kasutamine võimaldab luua rakendust, mis vastab täielikult lõputöö nõuetele, siis valiti see lähenemine Timeable rakenduse arendamiseks.

ASP.NET Core on paindlik raamistik, mis võimaldab mugavat MVC ja Web Api kasutamist ühes rakenduses, luues MVC ja API kontrollereid ning kasutades sama rakenduse äriloogikat. Selline pandlikkus jätab võimaluse lisada Web API poolse kliendi ning luua rakendusele täiendava rakenduse nutiseadmele või arvuti töölauale.

3.2.5 Kasutajaliidese keele valik

Kuna rakenduse tehnoloogiaks oli valitud MVC muster, siis rakenduse kliendipoolseks arenduskeelteks on HTML ja C# keelte segu. Failid laiendiga *.cshtml* kasutavad Razor märgistuse süntaksi [30], mis muudavad veebilehekülge dünaamilisemaks ja kasutajale mugavamaks. Razor süntaksi vaikimisi kasutavaks keeleks on HTML, kuhu on juurde lisatud C# koodi read.

Lisaks ülaltoodud keeltele, kasutatakse rakenduse kliendivaate loomiseks ka JavaScript keelt [31], millega animeeritakse kasutajaliidese elemente (näiteks kalender, kell).

3.3 Integreeritud arenduskeskkonna valik

.NET rakenduste arendamiseks kasutatakse kõige tihedamini kolme integreeritud arenduskeskkonda: JetBrains Rider, Visual Studio ja Visual Studio Code [32].

Rider loodi JetBrains-i poolt aastal 2016 ning seda on siiani arendatud. Rider toetab Linux, Windows ja MacOS operatsioonsüsteeme ning selle versioonid on samad, sõltumata operatsioonisüsteemist. Rider arenduskeskkonna kasutamine on tasuline (83-138 Ameerika dollarit aastas) [33], kuid tudengina sisselogimisel on arenduskeskkonna kasutamine ja allalaadimine tasuta. Lisaks sellele on Rider-is sisseehitatud koodi staatiline analüsaator – ReSharper, mis analüüsib koodi kompilleerimise vigade olemasolu ning pakub täiendavaid võimalusi koodis: otsimiseks, navigeerimiseks, komplekteerimiseks, optimeerimiseks ning koodi genereerimiseks ja palju muud [34].

Visual Studio ja Visual Studio Code (VS Code) on loodud Microsoft-i poolt. Visual Studio kasutamine on võimalik Windows ja macOS operatsioonsüsteemidel, Visual Studio Code on kasutatav ka Linux-il. Visual Studiost on võimalik paigaldada tasuta *Community* versioon, mis on mõeldud isiklikuks kasutamiseks ning mille funktsionaalsus on väiksem, kuid kõik Timeable rakenduse loomiseks vajalikud funktsioonid on toetatud [35]. Lisades ReSharper laiendi on võimalik muuta programmeerimist tõhusamaks. VS Code on tasuta koodiredaktor, mida on kasutatud nii C# rakenduste loomiseks, kui ka paljudes muudes keeltes programmeerimiseks. VS Code on muudetav laiendustega, kuid selle kasutamine C# arendamiseks on ebamugav, kuna projektide genereerimiseks ja pakettide haldamiseks on vaja kasutada käsurida, mis on aeganõudev õige käsu otsimisel ja selle kasutamiseks. Arvestades eelnevat, on mõistlik VS Code jätta kõrvale, kuna alternatiivide olemasolul on selle arenduskeskkonna kasutamine ebapraktiline.

Visual Studio kasutab võrreldes Rider-iga rohkem graafilise kasutajaliidese tüüpi lähenemist, mitte käsurea kasutamise lähenemist. Näiteks MVC ja API kontrollerite genereerimine toimub optsioonide all vastava käsu valimisel, ressursside failid on graafiliselt muudetavad ilma faili skripti muutmiseta. Selline lähenemine säästab aega vastava käsu ja selle parameetrite otsimiseks ja uurimiseks ning vähendab käsitsi tehtavat tööd, automatiseerides protsesse. Kuna lõputöö autori enimkasutatud arvuti on Windows operatsioonsüsteemiga, siis Visual Studio valimine arenduskeskkonnaks tundub kõige õigema otsusena. Lisaks sellele on kasutusel macOS operatsioonsüsteemiga arvuti, mille kasutamisel arendusetööks valitakse JetBrains Rider, mille kasutamine võrreldes Visual Studio macOS-iiga oli kiidetud macOS operatsioonsüsteemi kasutajate poolt [36].

3.4 Koodihalduskeskkonnad

Koodihalduskeskkondadena vaadeldakse kolme kõige populaarsemat süsteemi [37], kus on olemas tasuta kasutamise võimalus: GitLab, GitHub, Bitbucket. Versioonihaldustarkvarana kasutatakse Git-i [38], mis on toetatud kõikide eelmainitud keskkondade poolt.

GitHub on koodihaldus keskkond, millel on 100 miljonit repositooriumi seisuga august 2019 [39]. Koodihaldus keskkonda saab kasutada tasuta, omades kuni 3 privaatset repositooriumi ning piiramatu arvu avalikustatud repositooriume (mälu maht maksimaalselt 500 MB) [40]. GitHub on peamine keskkond, kus avalikult hoitakse NuGet pakettide lähtekoode [41].

Bitbucket on koodihalduskeskkond, mida kasutab üle kümne miljoni arendaja. Selle tasuta versioon lubab luua limiteerimata arvu privaatseid repositooriume ning jagada tööd viie liikme vahel [42]. Bitbucket-i omanik on Atlassian, kelle toodete hulka kuulub üks populaarsematest vea leidmise tööriistadest Jira [43], mille integreerimine Bitbucket-isse on kõige tõhusam võrreldes teiste keskkondadega.

GitLab on koodihaldus keskkond, mis on kasutusel 100 000 ettevõttes ning kus on rohkem kui 2200 kaasarendajat. GitLab keskkond põhineb DevOps arenduskultuuril ning toetab DevOps etappe: integratsioon, testimine, avaldamine, kasutusele võtmine, taristu haldamine [44]. Tasuta versioon võimaldab hoida mitut repositooriumi ning võimaldab kasutada mõnda DevOps funktsionaalsust, mis pole seotud korporatiivse töövooga [45].

Antud rakenduse jaoks võetakse kasutusele Tallinna Tehnikaülikooli GitLab, kus hoitakse arenduse protsessis muudetavat koodi ning GitHub, kus avalikustakse reliisiks valmis saanud kood.

3.5 Andmebaasi valik

Andmebaasid jagunevad mitmesse kategooriasse: hierarhiline (puustruktuuriga), relatsiooniline, objekt-orienteeritud, võrgu mudeliga, dokument-orienteeritud, graafi tüüpi [46]. Kõige kasutatavamad andmebaasid veebiarenduses on relatsioonilised andmebaasid, dokument-orienteeritud andmebaasid ning võti-väärtus paaride andmebaasid [47]. Antud lõputöö raames on vaatluse alla võetud kõige populaarsemad, suure kasutajabaasiga ja

ülikoolis käsitletud andmebaasid. Valik toimub eelkõige dokumenteerituse, hinna ja nõuetele vastavuse järgi.

Relatsioonilise andmebaasi kasutamisel luuakse esialgu skeem, millele andmebaas hakkab vastama. Skeemis määratakse, kuidas andmed jagunevad tabelitesse, millised veerud on tabelis ning veergude andmetüübid ja millised suhted on tabelite vahel. Dokument-orienteeritud andmebaasi kasutamisel andmed salvestatakse dokumentidesse JSON (*JavaScript Object Notation*), XML (*eXtensible Markup Language*), BSON (*Binary JavaScript Object Notation*) kujul ning andmebaasi välimus pole ette teada [48].

Oracle andmebaasi on arendatud Oracle Corporation poolt 1977-st aastast [49]. Andmebaasid on nii sisseehitatud, kui ka pilvepõhise teenustena [50]. Oracle pakub nii relatsioonilist andmebaasi, mis kasutab keele laiendina PL/SQL-i (*Procedural Language/Structured Query Language*), kui ka NoSQL andmebaasi, kus info hoidmine on võimalik dokumendi, veeru või võti-väärtus paaride formaadis [51]. Oracle andmebaas on kõige sobilikum valik ettevõtetele, kelle andmekogused on suured ning kiirused on tähtsad [52]. Väikese projekti raames selline andmebaasi installeerimine ja haldamine pole mõistlik ning andmebaasi hinnad on liiga kõrged [53].

MySQL on avatud lähtekoodiga relatsiooniline andmebaas, mis hetkel on Oracle toode [54]. MySQL andmebaas kasutab SQL/PSM (*Structured Query Language/Persistent Stored Modules*) keele laiendit [55]. Andmebaas on kiire ja tasuta. Kahjuks aga funktsionaalsus ja tugi on piiratud tasuta versioonis ning MySQL kasutamine vajab litsensi, kui seda kasutab äriline rakendus [52].

MariaDB on relatsiooniline avatud lähtekoodiga andmebaas, mis oli tehtud MySQL arendajate poolt, peale MySQL ostmist Oracle poolt. See on üles ehitatud jõudluse, stabiilsuse ja avatuse väärtustele. MariaDB süntaks põhineb MySQL süntaksil [56].

SQL Server on Microsoft-i poolt loodud relatsiooniline andmebaas, mida saab kasutada nii Windows kui ka Linux masinatel. SQL Server-il on loodud ka versioon, mis on mõeldud väiksematele rakendustele, mille andmete kogus ei ületa 10 GB. Andmebaas kasutab Transact-SQL-i keele laiendina [57]. SQL Server andmebaas on kiire ja stabiilne ning selle

tavaline versioon on palju kasutatud organisatsioonide poolt, kellel on tegemist Microsoft toodetega [52].

SQLite on relatsiooniline andmebaasi haldussüsteem, mis võrreldes teistega pole klient-server andmebaasi mootor, vaid on sisseehitatud lõpprakendusse. Keelena on SQL standard, mille süntaks on sarnane PostgreSQL süntaksiga. SQLite vajab vähe konfigureerimist ning terve andmebaas on salvestatud ühte faili. Võrreldes teistega ta on kiire, kuid piiratud funktsionaalsusega [58]. SQLite on tasuta ning on sobilik manussüsteemideks, asjade Internet seadmetes, rakenduste failivormingus, väikestes veebirakendustes [59].

MongoDB on dokument-orienteeritud andmebaas, mis salvestab andmeid JSON ja teistes NoSQL kujudes. Paindlik ja kergelt skaleeritav andmebaas, mis võimaldab paindlikke päringuid [60]. Puuduseks on pikk seadistamise protsess, ebaturvaline esmane konfiguratsioon [52].

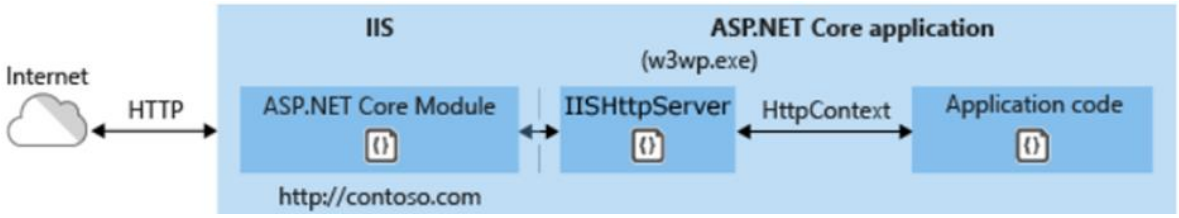
Timeable rakenduse jaoks oli valitud SQL Server Express andmebaas, mis on hästi sobilik väikse koguse andmete hoidmiseks.

3.6 Veebirakenduse haldus

Timeable veebirakenduse kasutamine pole mõeldud laiemale avalikkusele ning selle kasutamine on skoobi järgi piiratud Tallinna Tehnikaülikooli IT Kolledžis asuvatel arvutitel ehk lokaalvõrgus olevatel *host*-seadmetel. Kuna IT Kolledž omab enda serveriruume, siis rakenduse jooksutamiseks pole vaja rentida serverit, vaid saab kasutada olemasolevaid.

ASP.NET Core rakenduste *hostimiseks* võib kasutada Linux ja Windows operatsioonsüsteemiga servereid. Protsessijuhtimissüsteem vastutab rakenduse käivitamise eest serveri laadimisel ja selle taaskäivitamise eest juhul, kui server jookseb kokku. Enim kasutatud protsessijuhtimissüsteemid on IIS, Apache, Nginx [61].

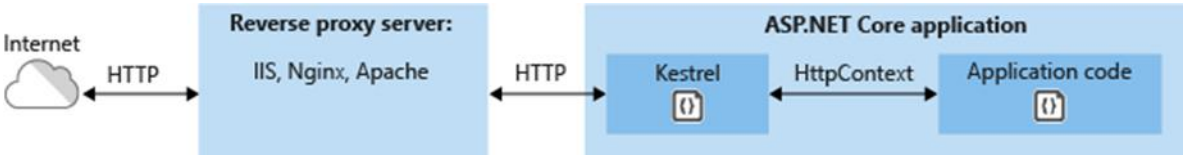
IIS[1] on veebiserver, loodud Microsofti poolt kasutamiseks Windows 7 ja uuemate operatsioonsüsteemidega serveritel. Kuna rakendus on väike ning selle jooksutamine on mõeldud ühes serveris, siis protsessisisese *hostimise* mudel on kõige sobilik. Sellise mudeli kasutamisel IIS laeb .NET Core CLR-i, kutsub välja Program.Main klassi ning käsitleb päringuid [62].



Joonis 1. IIS protsessisisese *hostimise* mudel.

Apache veebiserver ASP.NET Core rakenduste puhul tuleb paigaldada CentOS 7 serverile, et suunata HTTP-liiklus Kestreli serveris töötavale ASP.NET Core-i veebirakendusele. Apache veebiserveri käivitamiseks peab olema serveril *sudo*-privileegidega (*substitute user and do*) kasutaja, .NET Core käituskogu ning rakendus ise [63].

Nginx veebiserver tuleb paigaldada samuti Linux-ile, kuid kasutatavaks distributsiooniks on Ubuntu server. Samuti nagu Apache, vajab Nginx kasutajat *sudo*-privileegidega, .NET Core käituskogu ning rakendust. Nii Apache, kui ka Nginx on kasutatud pöördproksi serveritena [64].



Joonis 2. Pöördproksi server.

Rakenduse halduskesskkonnaks valiti Nginx veebiserver Ubunti masinal selle kasutamismugavuse pärast.

[1] Internet Information Services

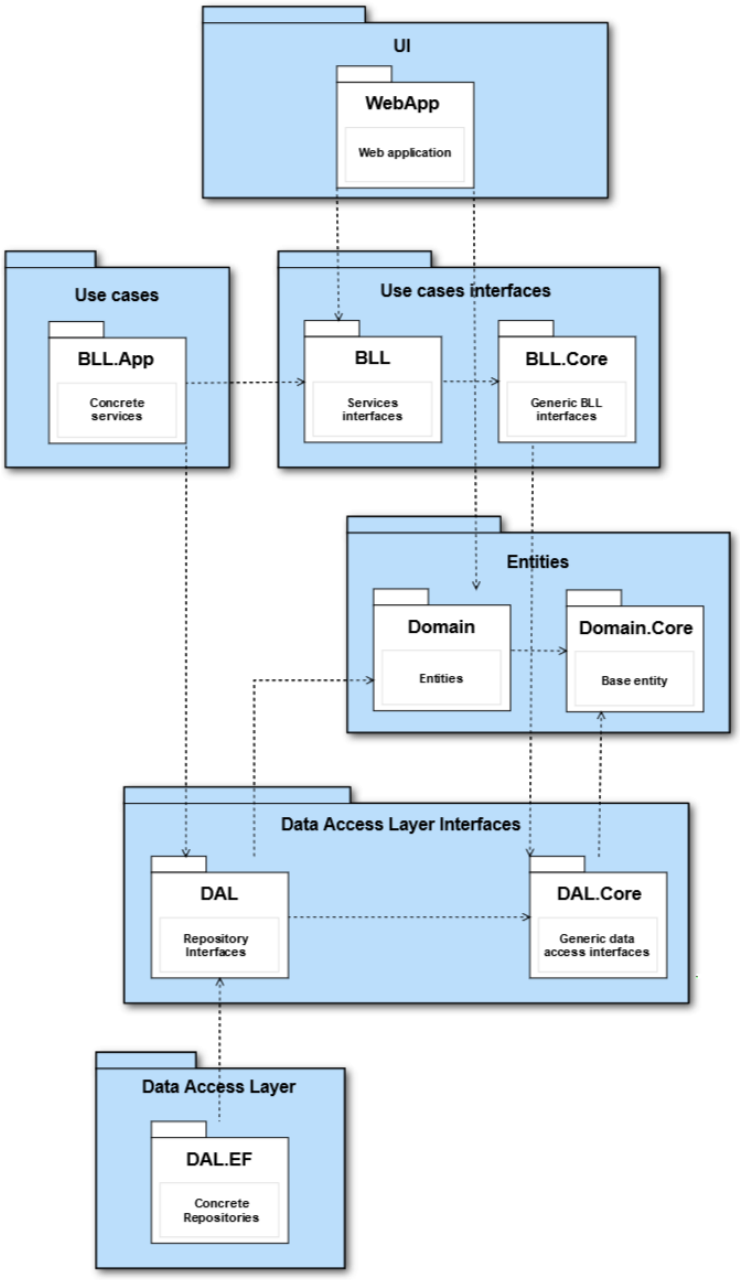
3.7 Veebirakenduse arhitektuur

Veebirakenduse arhitektuuri loomine viidi läbi lähtudes puhta arhitektuuri printsiipidest, eelkõige *SOLID* printsiipidest [65]:

- ***S**ingle Responsibility Principle* ehk üheainsa vastutuse põhimõte.
- ***O**pen-Closed Principle* ehk avatud-suletud vastutuse põhimõte.
- ***L**iskov Substitution Principle* ehs Liskovi asendatavuse põhimõte.
- ***I**nterface Segregation Principle* ehk liideste eraldatuse printsiip.
- ***D**ependency Inversion Principle* ehk pööratud sõltuvuse printsiip.

Veebirakenduse arhitektuur on välja toodud joonisena (Joonis 3) ning koosneb järgnevatest kihtidest:

- Andmemudeli kiht (*Entities*): baas-olemid ja olemid.
- Andmetele juurdepääsu kiht (*Data Access Layer*): üldine andmetele juurdepääsu liides, repositooriumite liidesed ning repositooriumid.
- Kasutamise juhtumise kiht (*Use Cases Layer* või *Business Logic Layer*): üldine äriloogika liides, teenuste liidesed ja teenused.
- Veebiliidese ehk kasutajaliidese kiht (*User Interface*): veebirakendus.



Joonis 3. Veebirakenduse arhitektuur.

3.8 Veebirakenduse disain

Antud veebirakenduse põhieesmärkideks on teha mugavamaks rakenduse administraatori töövoog ning luua mugav tarkvara televiisorites näidatava informatsiooni haldamiseks. Kasutajakogemuse kirjeldamine ning kasutaja disaini loomine annab võimaluse paremini hinnata, millist funktsionaalsust peab toetama veebirakendus.

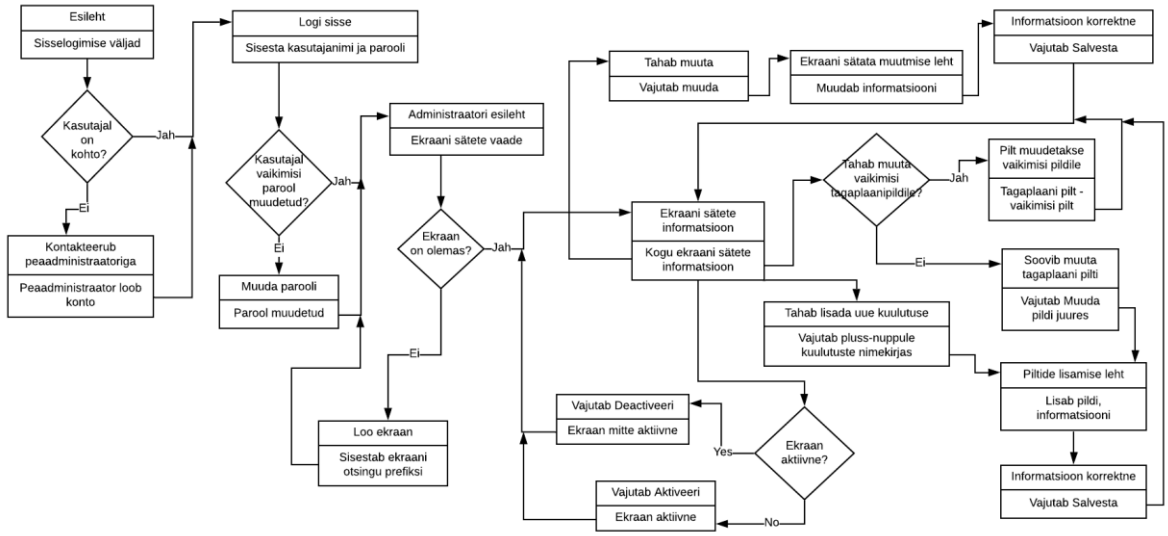
Selleks läbiti kolm sammu:

- Kasutajakogemuse disain.
- Kasutajakogemuse visualiseerimine Adobe XD tarkvaraga.
- Andmebaasi skeemi loomine.

3.8.1 Kasutajakogemuse disain

Lähtuvalt kasutajalugudest jaguneb administraatorite veebilehel navigeerimine 2 osaks: ekraani sätted ning tunniplaani/ürituste vaade. Peaadministraatori vaates lisandub ka rakenduse kasutajate (administraatorite) haldus. Sisselogitud kasutaja saab muuta ka enda kasutaja informatsiooni (nimi, e-mail, parool), kuid nende lehtede genereerimine on automaatne .NET koodi generaatori poolt [66] ning enamuses kasutatakse genereeritud disaini ning funktsionaalsust.

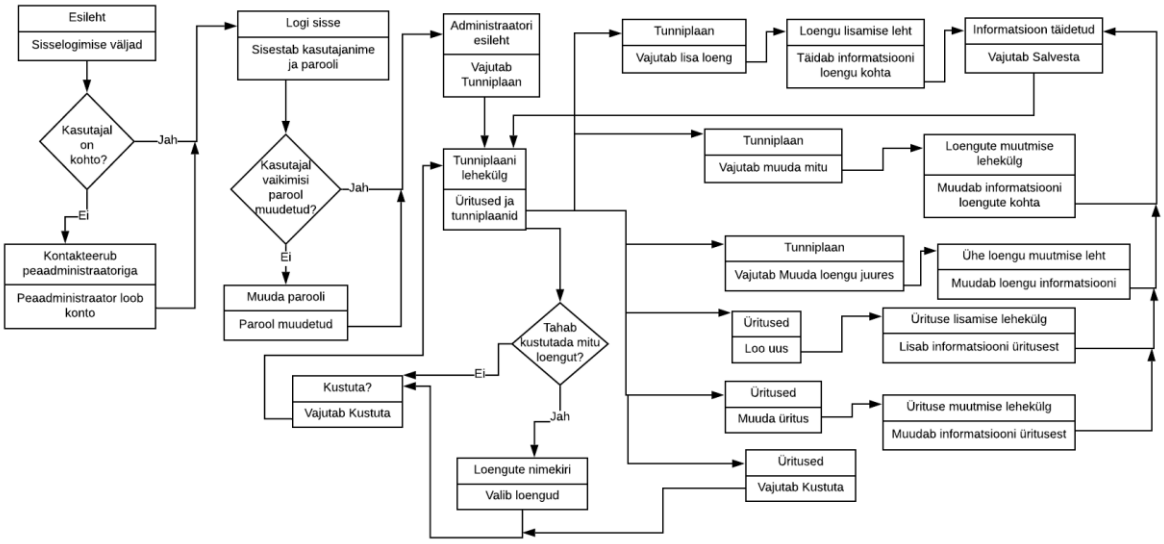
Joonisel 4 on toodud välja kasutajakogemus, mis keskendub ekraani sätete vaatamisele ja muutmisele. Ekraani sätted on sisselogitud kasutaja esilehel, kuna seal asub kõige tähtsam Timeable rakenduse funktsionaalsus: ekraani lisamine ja selle haldus.



Joonis 4. "Ekraani sätted" alamlehe kasutajakogemuse diagramm.

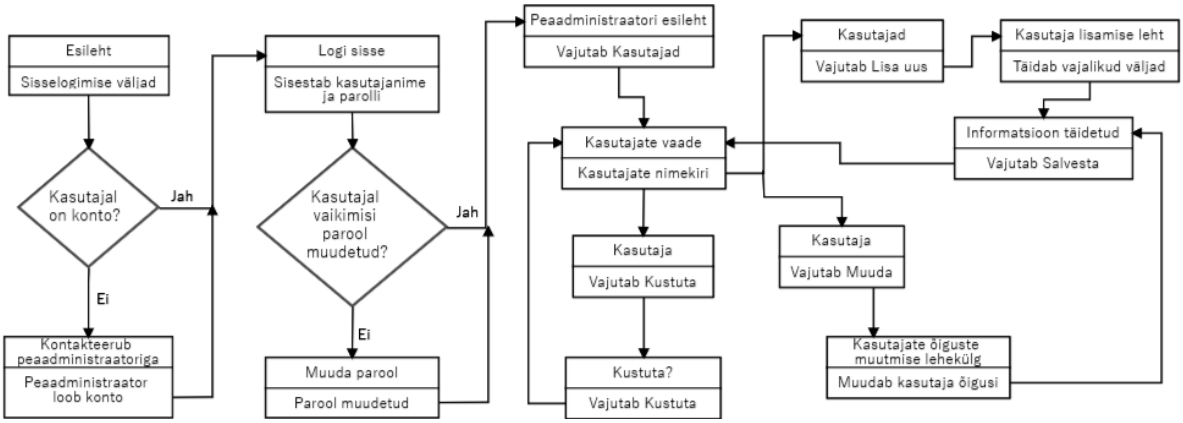
Joonisel 5 on välja toodud „Tunniplaan“ alamlehe kasutajakogemus. Sellel alamlehel on kuvatud nii tunniplaan, kui ka tunniplaani sees kuvatatavad üritused ning

põhifunktsionaalsuseks on nende lisamine, kustutamine, muutmine ning vaatamine ehk CRUD (*Create-Read-Update-Delete*) funktsionaalsus.



Joonis 5. "Tunniplaan" alamlehe kasutajakogemuse diagramm.

Joonis 6 kirjeldab alamlehe „Kasutajad“ kasutajakogemust. Kasutajatega alamleht kuvab peaadministraatorile kõike Timeable rakenduse kasutajaid ning annab õigust neid hallata.



Joonis 6. "Kasutajad" alamlehekülge kasutajakogemuse diagramm.

Kuna tunniplaani kuvamiseks on vaja vajutada „Näita ekraani“ nuppu, siis selle kasutajakogemuse kirjeldus jäeti välja selle lühiduse tõttu.

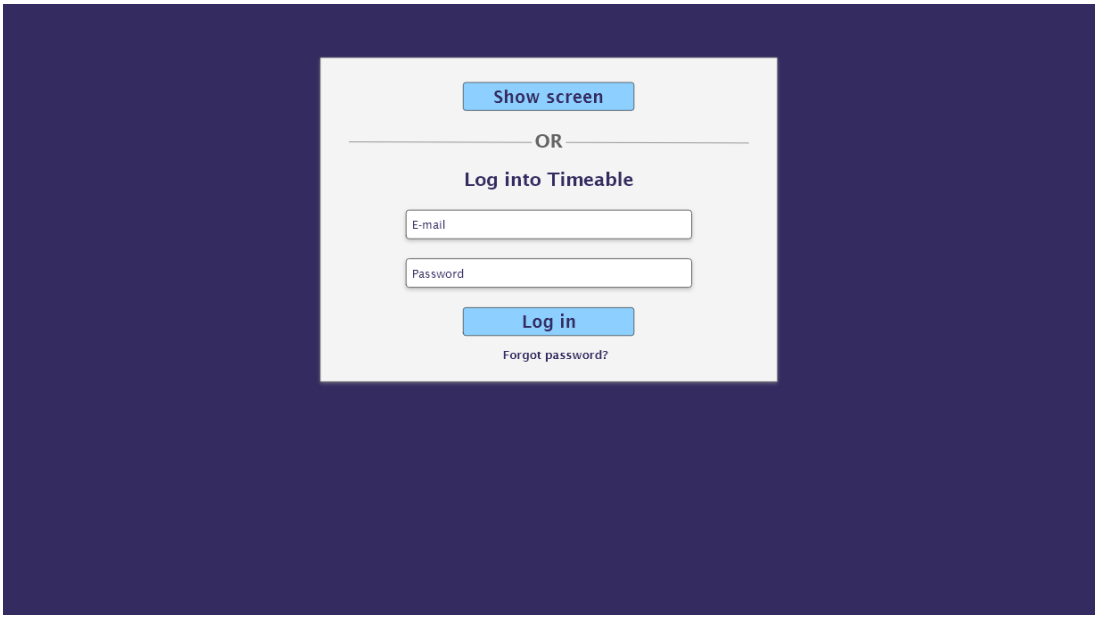
3.8.2 Veebilehe eeldatav disain Adobe XD tarkvaraga

Adobe XD tarkvara on laialdaselt kasutatud veebilehtede disainerite poolt prototüüpide loomiseks. Selle kasutamise keerukus on madal ning tudengite kontoga on selle kasutamine tasuta.

Prototüübi loomine on üks parimatest viisidest panna paika tulevase tarkvara visuaalne pool, arvestades kliendi soove, mis on kirjeldatud kasutajalugudes. Prototüüp ei vasta üks ühele lõplikule disainile, kuid annab ettekujutuse sellest, kuidas veebilehekülg hakkab välja nägema ning kuidas elemente on mugavam paigutada. Arenduse jooksul disaini loomine ja selle muutmine kuulutab rohkem ressurse, kui prototüübi loomine ning sellele tuginemine kliendivaadete arenduse protsessis.

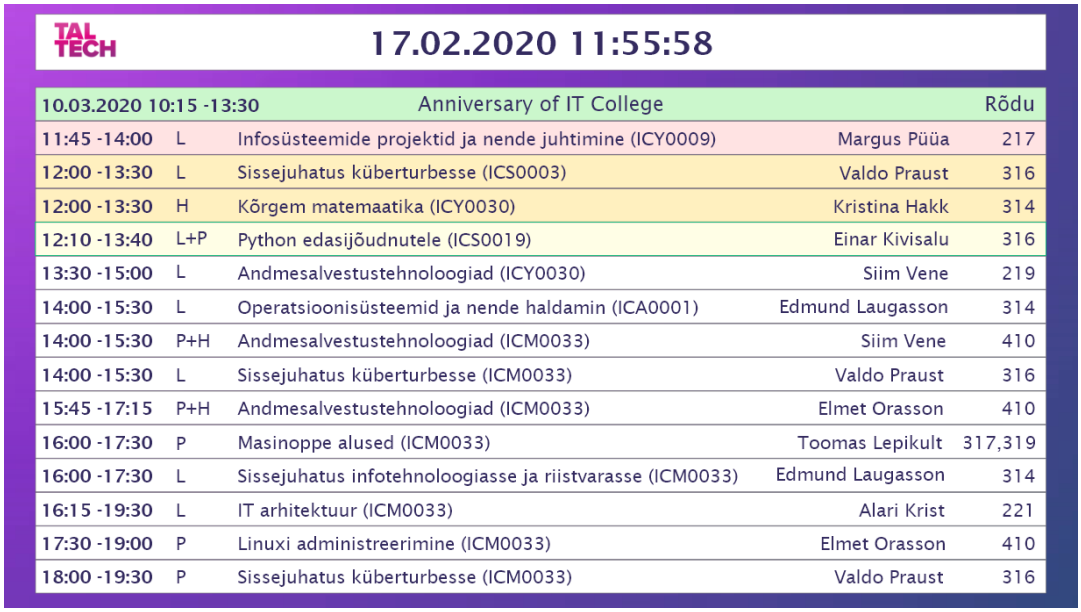
Veebilehe disaini loomisel arvestati, et kliendivaadete loomisel kasutatakse enamasti Bootstrap 4 tööriista ning elementide välimus prooviti teha võimalikult sarnane Bootstrap elementidele [67]. Kuna arendus toimub enamasti inglise keeles ning lõpus lisatakse eestikeelseid tõlked, siis prototüüpimisel kasutati inglise keelt.

Joonisel 7 on näha sisselogimata kasutaja esileht, kust saab navigeerida kas tunniplaani vaatesse või logida sisse administraatorina.



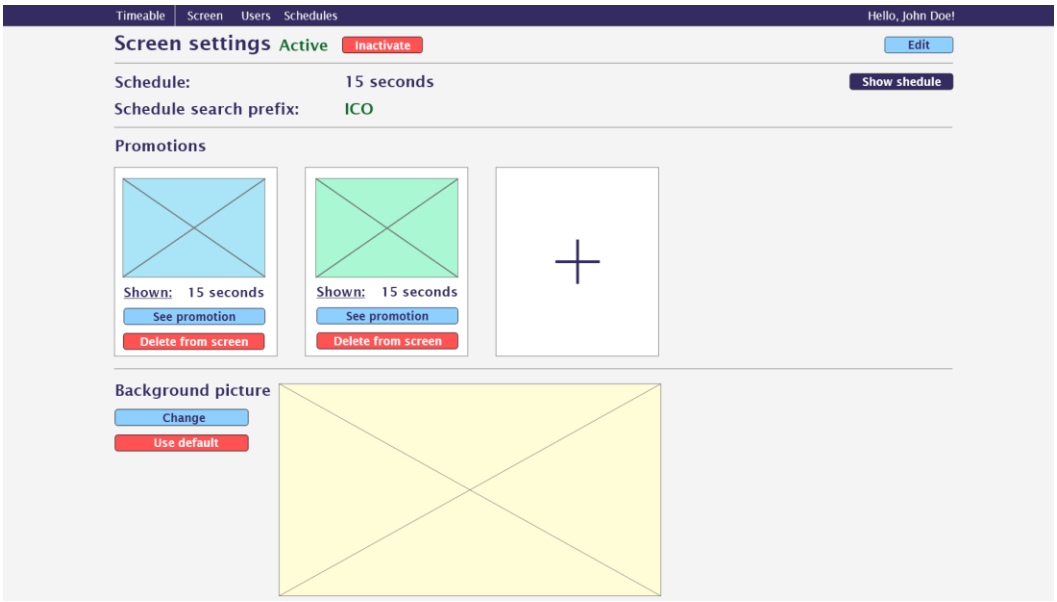
Joonis 7. Timeable rakenduse esilehe disain.

Joonisel 8 on näidatud, kuidas tunniplaani kuvamise lehekülg hakkab välja nägema. Näiteks olid välja toodud üritused, alanud ja varsti algavad loengud ja tulevikus algavad loengud.



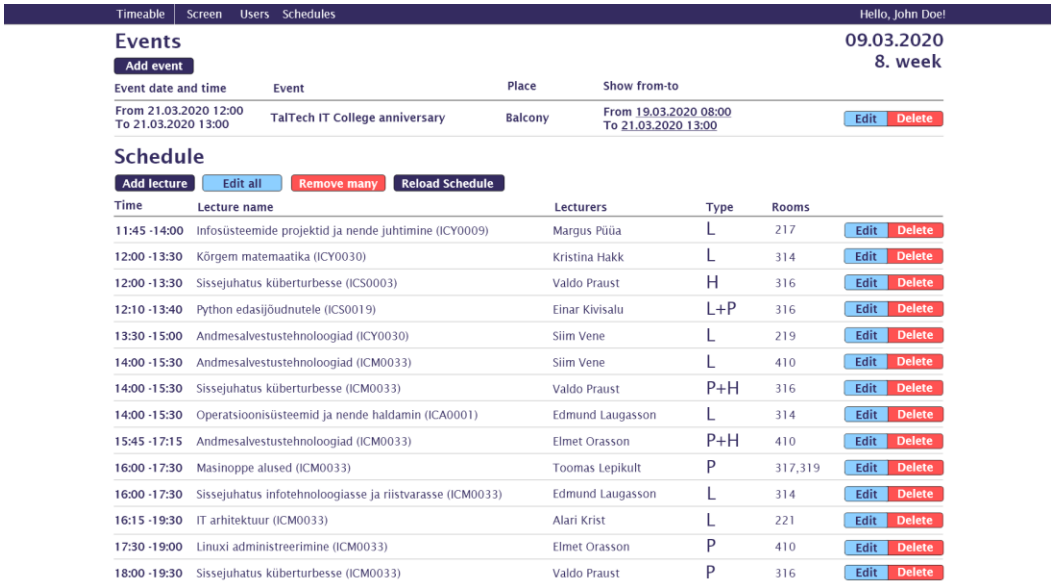
Joonis 8. Timeable rakenduse tunniplaani disain.

Joonisel 9 on näidatud administraatori põhileht, millel on tema ekraani sätete vaade. Siin on näha, kas ekraan on aktiivne või mitte, ekraanile kuvatud tunniplaani eesliide otsimiseks ning kui kaua see on kuvatud. Lisaks on piltidena ürituste reklaamid ja kasutatav tausta pilt.



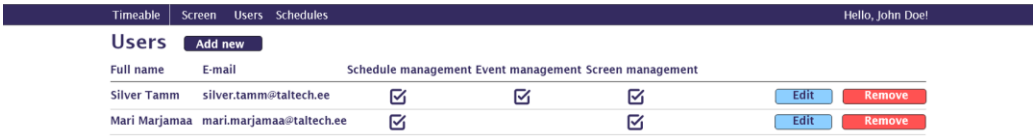
Joonis 9. Administraatori põhilehe ekraani sätete disain.

Joonisel 10 on näha ürituste ja tunniplaani leht, kus on kuvatud tänase päeva tunniplaan ning kõik üritused, mis toimuvad tulevikus.



Joonis 10. Ürituste ja tunniplaani veebilehe disain.

Joonisel 11 on näha funktsionaalsus, mis on saadaval ainult peaadministraatorile – kasutajate lisamine, muutmine ning kasutajatele õiguste jagamine.

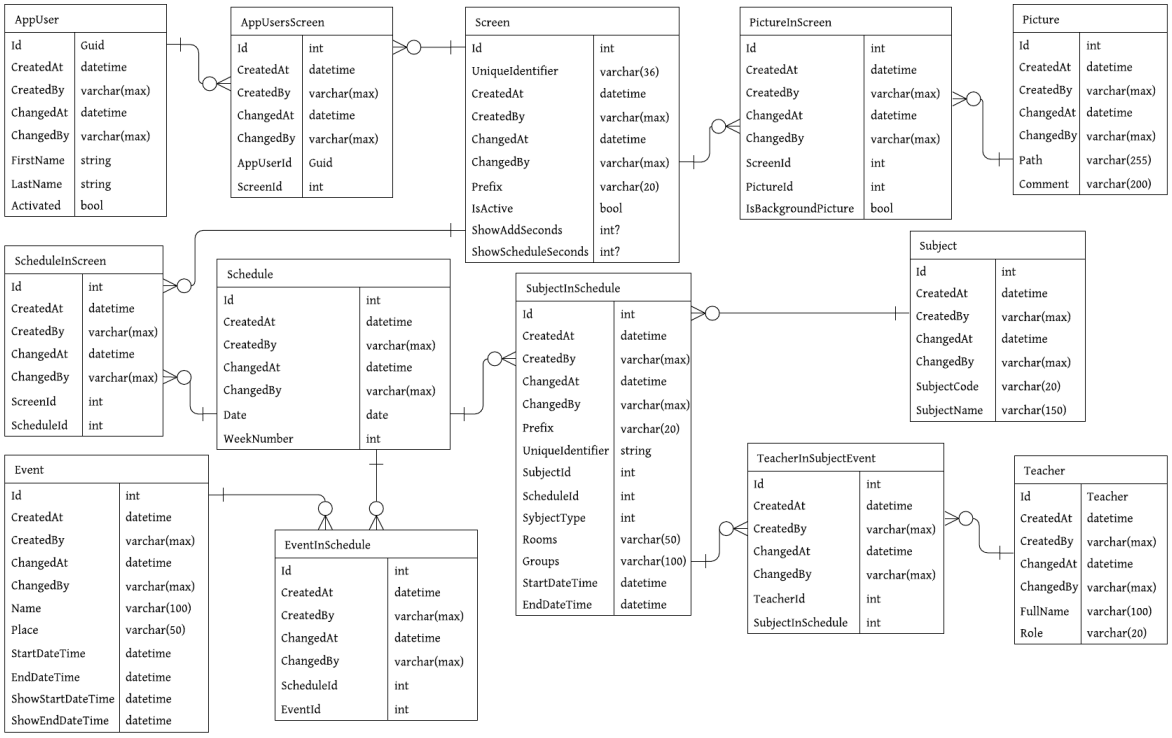


Joonis 11. Kasutajate halduse veebilehe disain.

3.8.3 Andmebaasi mudel

Lähtuvalt kasutajalugudest ja kasutajakogemuse disainist kujunes arenduse käigus hea ettekujutus andmebaasi struktuurist ning vajalikest olemitest. Olemisuhete diagramm loodi veebilehe Lucidchart tööriistaga.

Joonisel 12 on näidatud andmebaasi skeem. Skeemil on puudu mõned olemid, mis olid genereeritud automaatselt Entity Framework Core[1] tehnoloogiaga ning kirjeldavad identiteedi olemeid. Andmebaasi skeemi loomisel võeti aluseks ainult üks identiteedi olem – *AppUser* ehk rakenduse kasutaja, et kirjeldada kasutaja-ekraani olemite vahelisi suhteid. *AppUser* olemis on kirjeldatud vaid mõned veerud.



Joonis 12. Olemi-suhete diagramm.

3.9 Analüüsi kokkuvõte

Analüüsis sai käsitletud kasutajanõuded ning selle alusel ehitatud kasutajakogemuse diagramme, veebirakenduse põhilehtede disain ning andmebaasi skeemi olemi-suhete diagramm.

Tehnoloogia poole pealt ilmnes, et arvestades rakenduse suurust ning selle äriloogikat, oli mõistlik valida MVC tehnoloogia Timeable rakenduse ehitamiseks. Kuna ASP.NET on

[1] EF Core - .NET andmetele juurdepääsu tehnoloogia.

paindlik .NET tehnoloogia, mis võimaldab lisada vajadusel ka REST API, siis selline valik on tehtud paindlikkuse ning ajaressurside kokkuhoiu eesmärgil.

Serveripoolseks keeleks oli valitud C#, millega lõputöö autoril eelnevalt juba palju kogemust ning selle teadmiste täiendamine oli lihtsam. C#-l on kaks põhjalikku arenduskeskkonda, mida võib kasutada macOS, Windows ning Linux masinatel. Kliendirakenduse loomine põhineb MVC mustril ning kliendi vaadete arendamisel kasutatakse peamiselt Razor süntaksi ning komponentide animeerimiseks JavaScript-i.

Kuna rakendus on mõeldud kasutamiseks sisevõrgus, siis valiti lokaalne serverilahendus. Serverid võivad olla nii Windows, kui ka Linux operatsioonsüsteemidega, kuid antud lahenduseks valiti Ubuntu server, millele on paigaldatud Nginx proksiserver. Kood kirjutatakse Visual Studio ja JetBrains Rider programmis ning andmebaasi lahenduseks on valitud SQL Server Express. Koodihaldus toimub GitLab-is ning koodibaas on avalikustatud reliisina GitHub-is tulevastele arendustele ning tasuta kasutamiseks ka väljaspool IT Kolledži.

4 Veebirakenduse arendus

Käesoleva veebirakenduse arendus on jaotatud kolme suuremasse peatükki: veebiteenuse, kliendivaadete arendus, testimine. Kasutatud tehnoloogiad ja arenduse teed on kirjeldatud rakenduse osade arenduse peatükkides.

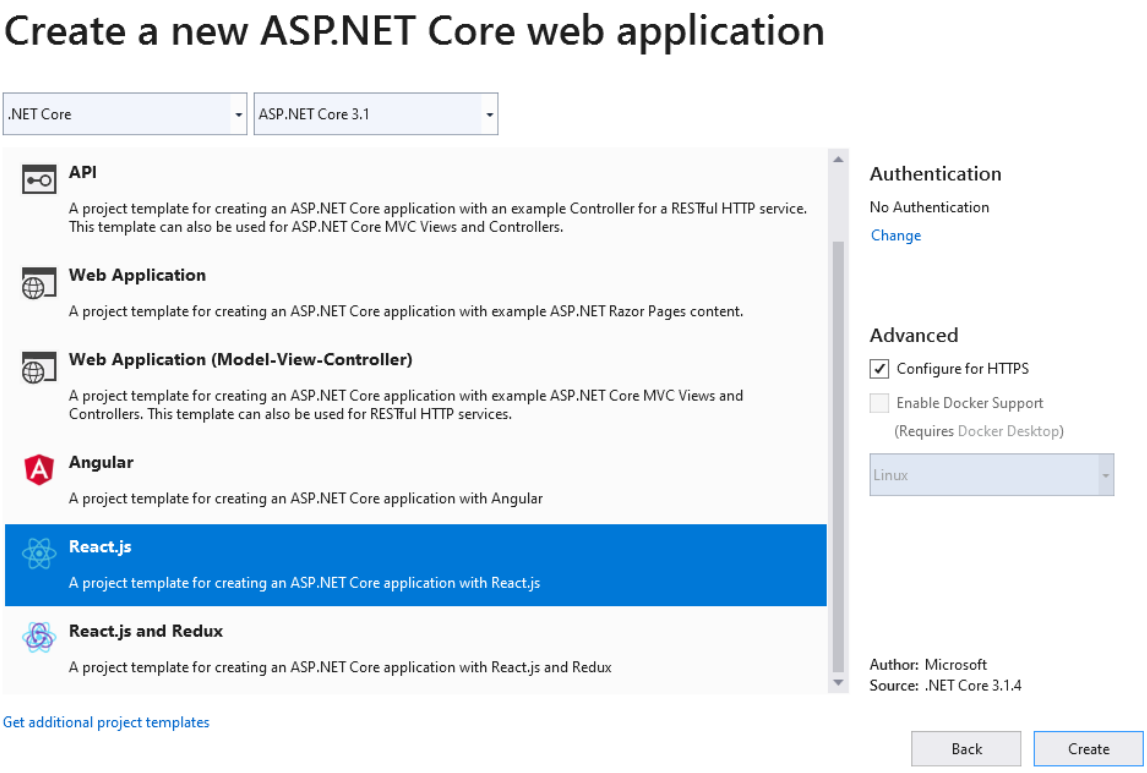
4.1 Veebiteenuse-poolne arendus

Veebirakenduse veebiteenus ja kliendivaated olid ehitatud kasutades .NET põhist ASP.NET Core tehnoloogiat. Veebiteenuse kiht vastutab nii rakendusele vajalike andmete pärimise ja salvestamise, kui ka rakenduse äriloogika eest. Järgnevates peatükkides on lähemalt kirjeldatud teenusepoolse arenduse erinevaid aspekte ja tehnoloogiaid.

4.1.1 ASP.NET Core rakenduse loomine

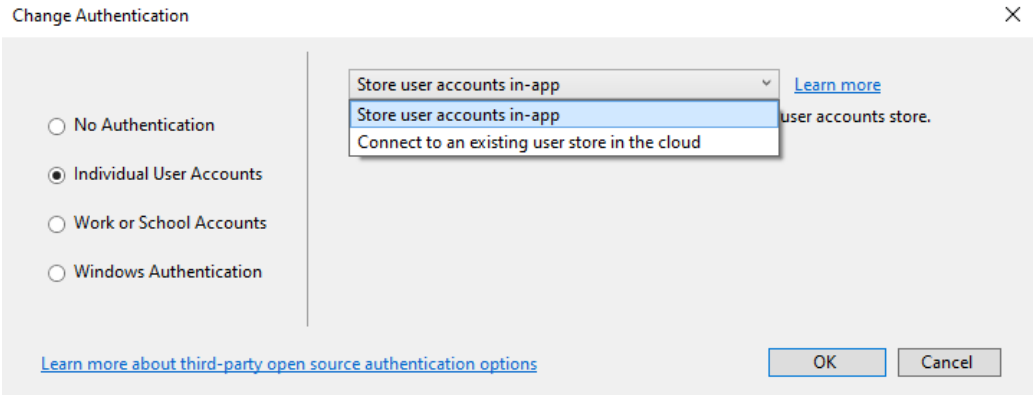
ASP.NET Core on ASP.NET MVC/Web API/Web Pages edasiarendus, mis ühildas endas kõik need tehnoloogiad [68]. ASP.NET Core rakenduse saab ehitada ja käivitada .NET keskkondades macOS, Linux ja Windows operatsioonsüsteemidega arvutitel. Nii Visual Studio-ga koos pakutavas IIS Express serveris, kui ka .ASP.NET Core-is sisseehitatud Kestrel serveris saab käivitada rakendust arendustöö käigus. Kuna rakendus on loodud kasutades enamikus Visual Studio 2019 arenduskeskkonda, siis on allpool välja toodud keskkonnapõhised arendusetapid.

Joonisel 13 on näidatud lahenduse loomise algfaas, mille käigus valitakse ASP.NET Core tehnoloogia, mille järgi ehitada rakendus. Valik oli RESTful teenuse, MVC ja Web Pages (Razor pages) vahel, lisaks pakub Visual Studio võimaluse luua RESTful teenust koos React või Angular klientrakendustega. Kuna varem oli valitud MVC mustri järgi rakenduse loomine, siis nüüd valitakse „*Web Application (Model-View-Controller)*“ rakenduse mudel.



Joonis 13. ASP.NET Core rakenduse mudelite valik.

Lisaks mudeli valimisele tuleb muuta autentimise viisi, kuna ekraani sätete muutmise õigused on ainult sisseloginud kasutajatel vastavalt nende õigustele. Vaatamise õigused on igal sisseloginud kasutajal. Joonisel 14 on näitatud autentimise viiside valikud. Timeable rakenduse jaoks valitakse *Individual User Accounts*, ehk kasutaja sisselogimine kasutajatunnuse ja parooliga, kuna see muudab kasutajate lisamise paindlikumaks ning ei sõltu organisatsiooni domeenist.



Joonis 14. Kasutajate autentimise viisi valik.

Rakenduse baas kujutab endast projekti, milles on rakenduse käivitamiseks mõeldud klassid, kasutajahalduse jaoks mõeldud funktsionaalsus, andmebaasi kontekst ja migratsioonid, näidis kontrollerid ja vaated (*Home*), jagatud vaated, rakenduse konfiguratsiooni JSON-fail ning ressurside jaoks mõeldud kataloog. Antud projekti raamistikuks on .NET Core 3.1.

4.1.2 Veebirakenduse teenuse-poolsed osad

Veebiraksenduse kõik *Class Library* projektid kasutavad .NET Standard raamistiku 2.1 versiooni. Veebirakenduse teenuse-poolne kiht on jaotatud mitmeks osaks: domeen, andmetele juurdepääsu kiht, äriloogika kiht, kontrollerid, HTML-i parsimise teenus tunniplaani saamiseks, tunniplaani ja andmebaasi uuenduste väljakutsumise teenus:

- Domain – domeenimudelit kirjeldav projekt. Projekti klassidest andmebaasi migratsiooni tegemisel Entity Framework Core moodustab andmebaasi olemeid ja vastavalt mudelile paneb paika nendevahelisi suhteid.
- Andmetele juurdepääsu kiht (*DAL – Data Access Layer*):
    - Contracts.DAL.Base ja Cotracts.DAL.App – projektid liideste hoidmiseks, mis kirjeldavad DAL baasklasse ning rakenduse põhiseid klasse.
    - DAL.DTO – *Data Transfer Objects* ehk andmeedastusobjektid. Projektis on klassid, mis kirjeldavad objekte DAL ja BLL kihtide vahel.
    - DAL.Base – ehk andmetele juurdepääsu kihi baasfunktsionaalsus, mis pole seotud rakenduse domeenimudeliga. Klasside funktsionaalsus on läbi kõikide domeeniobjektide kasutatav: baasrepositoorium, baastööühiku (*Unit Of Work - UOW*) mustri klass, baasobjektide kaardistaja, kasutades AutoMapper teeki [69].
    - DAL.App – ehk andmetele juurdepääsu kihi domeenimudelipõhine funktsionaalsus. Projektis on rakenduse andmebaasi konteksti klass ja rakenduse põhine tööühiku mustri klass, repositooriumid info pärimiseks ja objektide kaardistajad, mis ei kasuta väliseid teeke.
- Äriloogika kiht (*BLL – Business Logic Layer*):
    - Contracts.BLL.Base ja Cotracts. BLL.App –projektid liideste hoidmiseks, mis kirjeldavad BLL baasklasse ning rakenduse põhiseid klasse.

    - BLL.DTO – äriloogika andmeedastusobjektid. Projektis on klassid, mis kirjeldavad domeeniobjekte äriloogika perspektiivist.
    - BLL.Base – äriloogika kihi baasfunktsionaalsus. Projektis asub domeenimudelist sõltumata äriloogika teenus, andmeedastus objektide kaardistaja, mis samuti kasutab AutoMapper teeki, ja baasklass, mis kutsub välja UOW funktsioone.
    - BLL.App – projektis on äriloogika kihi domeenimudeli põhine funktsionaalsus: äriloogika teenused, DTO objektide kaardistajad ja baasklass töö haldamiseks.
- HTMLParser – ÕIS-is tunniplaani saamise teenus, mis parsib HTML objekte ning tagastab Schedule domeeniobjekti.
- TimedScheduleUpdateHostedService - teenus, mille funktsiooniks on tunniplaani saamise teenuse käivitamine ja tulemuse andmebaasi salvestamine. On TimeableAppWeb projekti osaks.
- Controllers – eraldi kaust MVC kontrolleritele, mis on TimeableAppWeb projekti osaks.

4.1.3 Andmebaasi ühendamine

Analüüsi osas valiti välja SQL Server Express andmebaas, mille kasutuselevõtmine rakenduses ei vaja palju seadistamist. Samuti ei tekita andmebaasi paigaldus suuri väljakutseid, sest arenduse teostamiseks on valitud Windows operatsioonisüsteem ja serverile Linux-põhine lahendus.

SQL Server Express on loodud Microsoft-i poolt ning selle kasutamine .NET keskkonnas on mugav. Rakenduse loomise hetkel lisatakse NuGet paketihalduris *Microsoft.EntityFrameworkCore.SqlServer* teeki automaatselt ning seda kasutatakse andmebaasiga ühenduse loomiseks.

SQL Server Express on relatsiooniline andmebaas ning Entity Framework Core pakub ORM (*Object-Relational Map*) funktsionaalsust, mis annab võimaluse arendajatel töötada .NET objektidega, välistades suuremat osa andmepöörduskoodi kirjutamise vajadusest [70].

Arenduse alguses luuakse domeeniobjekte varem koostatud olemi-suhete diagrammi alusel. Kõik domeeni objektid pärinevad *DomainEntity* objektist, mille omadusteks on unikaalne täisarvuline ID ja *meta*-andmed. Timeable rakenduse loomisel kasutati ka annotatsioone, millega piirati objekti omaduste sõnede pikkust. Lisaks sellele kasutati andmebaasi konteksti klassis LINQ-väljendie *SubjectInSchedules* tabeli *UniqueIdentifier* veeru indekseerimiseks [71]. EF Core ise vastutab suhete loomise eest, kuid vajadusel võib sätestada annotatsioonidega.

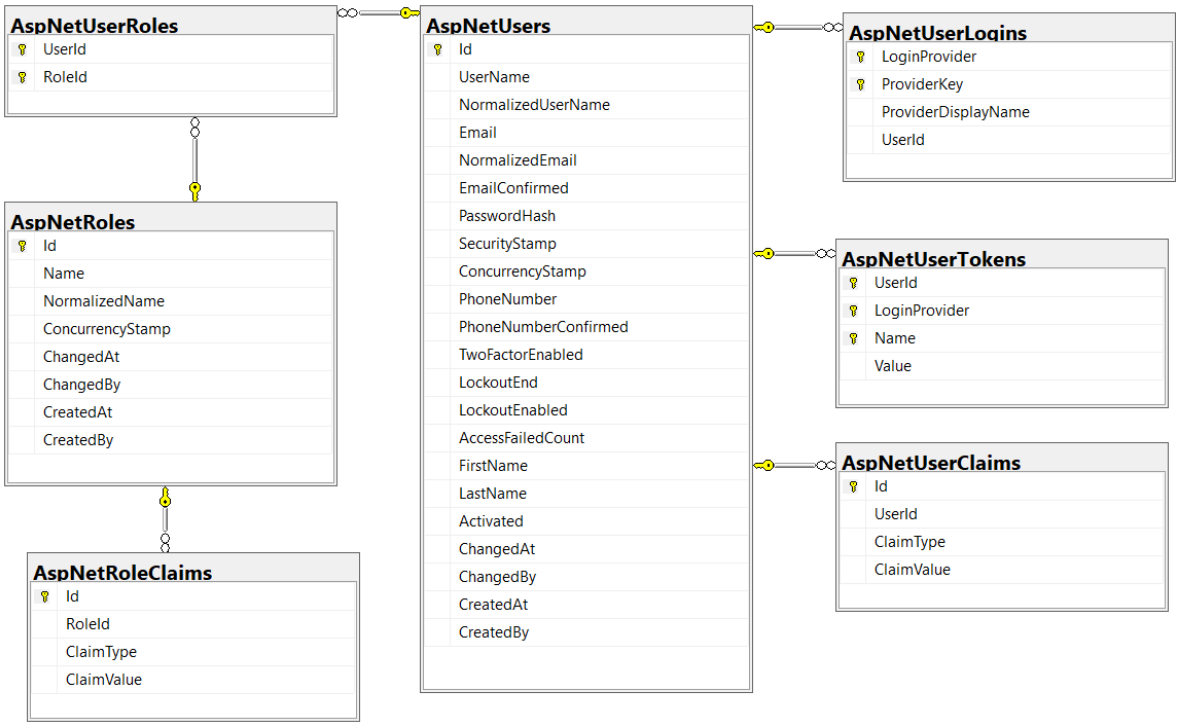
Andmebaasi loomiseks tuleb genereerida migratsioon, mida Visual Studio keskkonnas tehakse paketihalduri konsoolis käsuga *Add-Migration <migratsiooni nimi>*. Selle käsuga genereeritakse .cs laiendiga migratsioonifaile ning kutsudes seejärel *Update-Database* käsku, migratsioonifaile kasutatakse andmebaasi loomiseks ja muutmiseks [72].

Lisaks sellele lisati rakenduse Startup.cs klassi meetod, mis rakenduse käivitamisel kontrollib andmebaasi olemasolu ning vajadusel migreerib andmebaasi varem genereeritud failide alusel.

4.1.4 Kasutajate ja kasutajagruppide domeeniobjektid

Individual User Accounts autentimisviisi valimisel, genereerib EF Core vaikimisi identiteediga seotud tabeleid esimese migratsiooni loomisel ning seob neid *UserManager* ja *RoleManager* klasside funktsionaalsusega. *UserManager*-i ja *RoleManager*-i on sisseehitatud ASP.NET Core rakenduse identiteedi baasklassid, mis tulevad *Microsoft.AspNetCore.Identity* nimeruumist.

Timeable rakenduse mõistes kasutajate klassi, mis pärineb ASP.NET Core *IdentityUser* klassist, lisati rakenduse spetsiifilised omadused: *LastName*, *FirstName*, *Activated* ja *meta*-andmed. Rolle kirjeldavat klassi täiendati *meta*-andmetega. Joonisel 15 on näidatud andmebaasi lisatud identiteediga seotud tabelid. *AspNetUsers* ja *AspNetRoles* on tabelid, mis vastavad *AppUser* ja *AppRole* Domain projektis asuvatele domeeniobjekti klassidele.



Joonis 15. Identiteeti kirjeldavad tabelid.

Vaikimisi loodud kasutaja/rolli klasside asendamise konfiguratsioon on välja toodud peatükis **4.1.7 Konfigureerimine**.

4.1.5 Valideerimine

Kasutajaliidesest saadud informatsiooni valideerimisega tegeletakse äriloogika tasemel ning sarnaselt domeeniobjektidele kasutatakse annotatsioone äriloogika DTO objektidel. Selleks kasutatakse *System.ComponentModel.DataAnnotations* teeki, mis on üks .NET Standard raamistiku teekidest.

Joonisel 15 on toodud välja annotatsioonide näited. Kuna terves rakenduses oli koodis *Nullable Reference Types* sisse lülitatud, siis *[Required]* annotatsioonide kasutamine koodis on mõeldud ainult tõlgitud veateate kuvamiseks. *Nullable Reference Types* kohta kirjutatakse **4.1.7 Konfigureerimine** peatükis.

```
[Required(ErrorMessageResourceName = "ErrorMessage_Required",
ErrorMessageResourceType = typeof(Resources.Views.Common))]

[MaxLength(30, ErrorMessageResourceName = "ErrorMessage_MaxLength",
ErrorMessageResourceType = typeof(Resources.Views.Common))]

[Display(Name = nameof(SubjectCode), Prompt = nameof(SubjectCode),
ResourceType = typeof(Resources.Domain.SubjectInSchedule.SubjectInSchedule))]

public string SubjectCode { get; set; } = default!;


[Required(ErrorMessageResourceName = "ErrorMessage_Required",
ErrorMessageResourceType = typeof(Resources.Views.Common))]

[MaxLength(200, ErrorMessageResourceName = "ErrorMessage_MaxLength",
ErrorMessageResourceType = typeof(Resources.Views.Common))]

[Display(Name = nameof(SubjectName), Prompt = nameof(SubjectName),
ResourceType = typeof(Resources.Domain.SubjectInSchedule.SubjectInSchedule))]

public string SubjectName { get; set; } = default!;
```

Joonis 16. Valideerimine annotatsioonidega.

4.1.6 Internalisatsioon ja küpsisfailid

Timeable rakendus oli loodud Tallinna Tehnikaülikooli IT Kolledži jaoks, kus töötavad nii eesti kui ka inglise keelt igapäevaelus kasutavad inimesed. Rakenduse tellijaks on eesti keelt emakeelena kasutav multimeediaspetsialist ning rakendusse eesti keele lisamine võiks olla prioriteediks. Võttes arvesse, et tulevikus kasutajad võivad muutuda, lisati rakendusse ka inglise keel.

Lokalisatsiooni lisamine mõjutab kõige rohkem staatilist teksti. Selle tõlkeid hoitakse *.resx* failides ning faili nimetustes kasutatakse keele lühendit: eesti keeles – **et-EE**, inglise keelseid ressursse hoitakse vaikimisi kasutatavas failis ilma keele lühendita. Internalisatsioon mõjutab ka kuupäevade näitamist vastavalt valitud kultuuri formaadile.

Keelteks olid lisatud eesti ja inglese keel, mille vahetamine toimub *dropdown*-nimekirja abil. Keele vahetamisel on vaja jätta kasutaja valik meelde ning selleks võetakse kasutusele küpsisfailid [73]. Euroopa liidus on range küpsisfaili poliitika, mis nõuab kasutaja teavitamist küpsisfailide kasutamise viisist [74]. Selleks oli loodud *alert*-aken, kus on lühidalt kirjeldatud

küpsiste kasutamine antud veebilehel ning millega nõustamine lubab kasutada antud leheküljel küpsisfaile.

4.1.7 Konfigureerimine

Rakenduse ehitamise seadistamine toimub kasutades Directory.Build,props faili - kasutaja määratletud fail, mis pakub kohandeid lahenduses asuvatele projektidele [75]. Timeable rakenduse raames konfigureeriti keele versioon 8.0 ning sisse oli lülitatud *Nullable Reference Types*, mis hoiatab, et viitetüübid võivad osutada nullile ja iga viitetüübiga töötav kood peab juba teadma, et objekt pole null, või tegema selgesõnalise kontrolli [76].

Rakenduse seadistamine toimub nii läbi *appsettings.json* faili, kui ka *Startup.cs* klassi.

JSON-konfiguratsiooni fail on loodud rakenduse sätestamiseks juba kompilleeritud koodiga. JSON-failis on kirjutatud andmebaasiga ühenduse loomiseks sõne, rakenduse käivitamise hetkel kutsutavad funktsioonid, TalTech korpuse eesliide mida vaikimisi kasutatakse tunniplaani saamiseks. Joonisel 17 on välja toodud *appsettings.json* faili sisu.

```
{
   "AppDataInitialization": {
      "DropDatabase": false,
      "MigrateDatabase": true,
      "SeedIdentity": true,
      "SeedData": true
   },
   "ConnectionStrings": {
      "DefaultConnection": "Data Source=.\\SQLExpress;Initial
      Catalog=TimeableDatabase;Integrated
      Security=SSPI;MultipleActiveResultSets=True;"
   },
   "ScreenDataPrefix": "ICO",
   "AllowedHosts": "*"
}
```

Joonis 17. appsettings.json faili sisu.

Rakendusesisene konfigureerimine toimub *Startup.cs* klassis, kus lisatakse rakenduses kasutatavaid sõltuvusi, kasutades 2 peamist mehhanismi selle jaoks: *AddScoped* (iga päringu jaoks luuakse oma teenuse objekt) ja *AddSingleton* (teenuse objekt luuakse üks kord esimesel pöördumisel) [77]. Nende mehhanismide abil registreeritakse teenuseid ja abiteenuseid DAL ja BLL kihtidest.

Lisaks sellele, vahetatakse välja vaikimisi kasutatud anbmebaasi konteksti klass ja identiteedi klassid arendaja poolt loodud klassidega. Joonisel 18 on toodud välja uue konteksti registreerimine ning Domain projektist identiteedi klasside kasutusele võtmine.

```
services.AddDbContext<AppDbContext>(options =>
    options.UseSqlServer(
        Configuration.GetConnectionString("DefaultConnection")));

services.AddIdentity<AppUser, AppRole>()
    .AddEntityFrameworkStores<AppDbContext>();
```

Joonis 18. Andmebaasi konteksti ja identiteedi klasside lisamine.

Rakenduse kesksete konfiguratsioonidena lisati kasutaja paroolide piirangud ja sätted ning marsruutimine kasutaja vaadete ja administraatori ala vahel.

Internalisatsiooni kasutamiseks lisati rakendusse ka küpsisfailide kasutamine, mida tuleb konfigureerida rakenduse *Startup.cs* failis. Joonisel 19 on näidatud küpsiste ja lokalisatsiooni teenuste seadistused rakenduse konfiguratsiooni osas.

```
services.Configure<CookiePolicyOptions>(options =>
{
    options.CheckConsentNeeded = context => true;
    options.MinimumSameSitePolicy = SameSiteMode.None;
});

services.Configure<RequestLocalizationOptions>(options => {
    var supportedCultures = new[]{
       new CultureInfo(name: "en-GB"),
       new CultureInfo(name: "et-EE")
    };
    options.DefaultRequestCulture = new RequestCulture(culture: "en-GB",
    uiCulture: "en-GB");
    options.SupportedCultures = supportedCultures;
    options.SupportedUICultures = supportedCultures;
});
```

Joonis 19. Internalisatsiooni ja küpsiste seadistus.

Lisaks ülaltoodud lokalisatsiooni teenuse konfiguratsioonile *ConfigureServices* meetodis, on vaja lisada lokalisatsiooni vahevara kasutamist veebilehel andmete põhjal kultuuriväärtuse saamiseks, mis on näidatud Joonisel 20.

```
app.UseRequestLocalization(options:
    app.ApplicationServices
        .GetService<IOptions<RequestLocalizationOptions>>().Value);
```

Joonis 20. Internalisatsiooni vahevara seadistus.

4.1.8 ÕIS-ist tunniplaani saamise automatiseerimine

Peamiseks Timeable rakenduse eeliseks on tunniplaani salvestamise automatiseerimine. Võrreldes teiste rakendustega, mis olid kirjeldatud peatükis **2.1 Eksisteerivad lahendused**, ei vaja Timeable rakendus igapäevast manuaalset tunniplaani kopeerimist.

Tunniplaani õppeinfosüsteemist kättesaamiseks kasutatakse ÕIS-ist HTML sisu saamist ning sellest vajaliku informatsiooni otsimist. HTML sisu saamiseks kasutatakse HttpClient klassi meetodeid *GetStringAsync()* veebilehe sisu saamiseks sõne kujul ning *Dispose()* mälu vabastamiseks peale info saamist. Kuna veebilehe DOM (*Domain Object Model*) struktuur võib ajaga muutuda ning HTML-i parsimist muudatuste ilmumisel peab muutma, siis valiti teine meetod vajaliku informatsiooni saamiseks, kasutades *regex*[1] mustrit. Regex mustri kasutamisel saadakse loengute informatsioon gruppidena. Kuna iga õpperühma jaoks võib õppeinfosüsteemist alla laadida või avada *iCalendar*-faili, siis kasutatakse seda tunniplaani saamiseks selle hea struktuuri pärast, kust info otsimiseks võib kasutada *iCalendar* standardi võtmesõnu. Selline lahendus vajab tulevikus väiksema tõenäosusega muutmist, kuna faili struktuur on fikseeritud standardiga.

Grupeeritud andmetest moodustatakse DTO objektid, mida kaardistatakse ümber domeeniobjektiks andmebaasi salvestamisel. Andmebaasi salvestamisel valitakse andmed, mis vastavad valitud päevale ning mille toimumise koht vastab valitud eesliitele.

4.1.9 Turvalisus

Turvalisus on tähtis aspekt tänapäeva rakendustes. Timeable rakenduses saavutati see kasutades ASP.NET Core tehnoloogias sisseehitatud turvalisustehnoloogiad ning kasutades rolle, mille järgi saab anda kasutajatele õigust seadistada ekraani vaadet.

---

[1] ***Reg****ular* ***ex****pression* ehk regulaaravaldis

Tunniplaani vaade on Timeable rakenduses saadaval ka sisselogimata kasutajale ning veebilehe avamisel on valik, kas vaadata tunniplaani või logida sisse sätete muutmiseks. Kõik ekraani, tunniplaani, sündmuste haldus on saadaval ainult autoriseeritud kasutajatele vastavate õigustega. Peaadministraatoril on ka kasutajate halduse võimalused. Kõik kasutajad lisab rakendusse peaadministrator ning annab õigusi muuta seadistusi: ekraani haldus (sealhulgas ka ürituste reklaam), tunniplaanis ainete haldus, tunniplaanis näidatavate sündmuste haldus. Kõikidel sisselogitud kasutajatel on vaatamise õigus, kuid puudub muutmise õigus, kui seda pole antud peaadministraatori poolt. Autorisatsioon on seadistatud koodis vastavate annotatsioonidega.

ASP.NET Core suur eelis on andmetekaitse tehnoloogiad, mis enamuses on sisseehitatud raamistikku. ASP.NET Core pakub juba sisseehitatud identiteedipakkujaid, mis võimaldavad luua kasutajaid nii lokaalselt, kui ka kasutades kolmandate osapoolte sisselogimist [78]. Timeable rakendus kasutab *Microsoft.AspNetCore.Identity* teeki kolmandat versiooni. Turvalisuse poolest teegi kolmas versioon pakub järgnevat kasutaja parooli räsimist: kasutatakse PBKDF2 standardit koos HMAC-SHA256 kontroll-mehhanismiga, 128-bitise soolaga, 256-bitise alamvõtmega. Iteratsioonide arv räsi arvutamisel on 10 000 [79]. Arvestades rakenduse sihtrühma, selle funktsionaalsust ning selle kasutamise kohta, võib väita, et ASP.NET Core turvalisuse tehnoloogia on piisav antud rakenduse raames ja antud ajal. Lisaturvalisuse mõttes võib lisada kohandatud parooli räsistamist ning see võiks kuuluda Timeable lahenduse tulevasse arendusse.

ASP.NET Core raamistik pakub ka tehnoloogiaid, mis kaitsevad tuntumate rünnakute eest:

- **SQL süstimise rünnak** – Timeable rakenduses ei kasutata puhta SQL-i päringuid, kasutatakse LINQ struktuuri ja sisendi valideerimist SQL süstimise vältimiseks [80].
- **Saidideülene skriptimise rünnak** – selle vältimiseks valideeritakse kasutajate sisendi [81].
- **Saitideülene päringu võltsimine** – kasutades <*form*> HTML märgendit kasutajate vormide jaoks ASP.NET Core hoolitseb selliste rünnakute ennetamise eest. Sellise kaitse kasutamise eelduseks on ühe nendest API-dest kasutamine: *AddMvc*

(kasutatakse Timeable rakenduses), *MapRazorPages*, *MapControllerRoute*, *MapBlazorHub* [82].

- **Ümbersuunamisrünnakud** – Timeable rakenduses kasutatakse ainult rakendusesisest navigeerimist. Vastavalt kasutaja tegevustele navigeeritakse koodi sees täpsustatud lehekülgedele ning ei edastata POST-päringutes sihtkohta parameetrina [83].

4.1.10 Piltide üleslaadimine

Rakenduses on vaja peale tavaliste andmete numbrilises ja sõnalises formaadis, salvestada ka pilte. Kuna on kasutusel MVC muster, siis pildifailide salvestamine on lihtsam, võrreldes server-klient lahendusega.

Pilte salvestatakse ühekaupa ning need pildid tähistavad kas ürituse reklaami või taustapilti. Piltide salvestamine failisüsteemi on hea tava ning andmebaasi salvestamine on mõeldud piltide *meta*-andmete jaoks [84]. Faile lisatakse *wwwroot* kataloogi rakenduse siseselt ning salvestatakse andmebaasi relatiivne teekond pildini. Piltide salvestamiseks ja kustutamiseks kasutatakse *Path.Combine()* meetodit, mis annab võimaluse kombineerida kogu teekond vastavalt kasutatavale süsteemile. Piltide kuvamisel otsitakse pilte andmebaasi salvestatud teekonna järgi.

4.1.11 Kasutatud teegid

Lisaks teekidele, mis olid loodud koos projektiga ning mis olid kirjeldatud ülalpool on lahendusse lisatud järgmised teegid:

- *Microsoft.VisualStudio.Web.CodeGeneration.Design* – kontrollerite ja vaadete genereerimiseks [85].
- *Microsoft.Extensions.Identity.Stores* – hõlbustab sisse logitud kasutaja andmete kohandamist ja võimaldab lisada rakendusele sisselogimisfunktsioone [86].

4.2 Kliendirakenduse-poolne arendus

Rakenduse loomise tehnoloogiaks valiti ASP.NET Core MVC mustriga, kus vaated on rakenduse osaks ning ei kasutata välist API-d andmete saamiseks. Vaadete keele süntaksiks

on Razor süntaks. Razor süntaks koosneb Razor märgistusest, C# koodi osadest ja HTML-i märgendist. HTML on süntaksi põhikeeleks, kuhu on võimalik lisada C# avaldisi, mida renderdatakse HTML-väljundisse [30].

Vaated ASP.NET Core rakenduses on genereeritud koos kontrolleritega automaatselt kasutades koodigeneraatorit. Vaikimisi genereeritakse CRUD vaateid ja *Index* vaadet, kus näidatakse kõiki domeeniobjekte andmebaasist. Genereerimise ajal valitakse domeeniobjekti ja andmebaasi kontekst, mille järgi vaateid genereerida. Iga vaate veebilehekülje vastab üks GET ja üks POST meetod, välja arvatud *Index* ja *Details* vaadetel (ainult GET). Kõiki genereeritud kontrollereid salvestatakse *Controllers* kausta ning nendele vastavaid vaated gruppeeritakse kontrolleri nimega kataloogi ja lisatakse *Views* kausta.

Lisaks kontrolleri-vaadetele rakenduses on jagatud-vaadete kogu, mis on samad kõikidel selle ala vaadetel: veebilehekülgede päis ja jalus, validatsiooni skriptid.

4.2.1 Alad

Veebirakenduses on kasutatud alasid funktsionaalsuse mugavaks eraldamiseks. Loodud on kaks ala: klientide ja andministraatorite alad. Kõik kontrollerid, domeeni vaated, jagatud vaated, vaadete mudelid (*ViewModels*) on jagatud nende alade vahel ning asuvad vastavates kasutades: kliendi ala on otse *TimeableAppWeb* projektis ning administraatori ala on *TimeableAppWeb* projektis *Admin* kaustas. Kõik administraatori ala kontrollerid on märgitud *[Area("Admin")]* annotatsiooniga, sest kaustadesse jagamine ei tähenda alasse kuulumist, vaid on mõeldud mugavuse jaoks.

Klientide ala alla kuulub esilehekülg (kus on valik kas sisse logida või näidata tunniplaani vaadet) ja tunniplaani vaade. Kõik need vaated on saadaval sisselogimata kasutajatele.

Administraatorite ala avaneb peale sisselogimist, kuna lugemise õigused on antud kõikidele sisselogitud kasutajatele. Vastavate õigustega andministraatoritel on ekraanide, sündmuste reklaamide, tunniplaani, sündmuste haldamise võimalused ning peaadministraatoril ka kasutajate haldamise võimalus. Peaadministraatori kasutajate halduse kontrollerile on lisatud *[Authorize(Roles = nameof(RoleNamesEnum.HeadAdmin))]* annotatsioon, mis piirab funktsionaalsusele ligipääsu peaadministraatori rolli puudumisel.

4.2.2 Vaadete mudelid

Äriloogika kiht tagastab DTO objekte, mis sobivad kasutamiseks otse kliendirakenduses. Võib piisata ka ühest BLL objektist või selle loendist, kuid tihti on vaja rakenduse ühes vaates muid BLL kihi objekte ning lisaomadusi.

Selle probleemi lahendamiseks on loodud vaadete mudeli klassid, kus kombineeritakse mitme erineva BLL kihi andmeedastus objekte ning lisaomadusi, mida saab siis kasutada vastavates vaadetes. Joonisel 20 on välja toodud üks vaate mudel, mida kasutatakse piltide lisamisel ja muutmisel. Kuna pildid võivad olla nii taustapildid, kui ka ürituste reklaamid, siis vastavalt sellele väärtustatakse *IsBackgroundPicture* omadust ja kuvatakse vastavalt sellele HTML objekte. Kuna taustapilt ei vaja selle näitamisel sekundite valikut, siis viimast nelja omadust taustapildi lisamisel pole kasutatud.

```
public class PictureCreateEditViewModel
{
    public Picture Picture { get; set; } = default!;
    public int ScreenId { get; set; } = default!;
    public bool IsBackgroundPicture { get; set; }
    public SelectList PromotionSecondsSelectList { get; set; } = default!;
    public string? ShowPromotionSecondsString { get; set; }
    public SelectList ScheduleSecondsSelectList { get; set; } = default!;
    public string? ShowScheduleSecondsString { get; set; }
}
```

Joonis 21. Vaate mudeli näide piltide lisamise ja muutmise vaadete jaoks.

4.2.3 Bootstrap-i kasutus

Kasutajaliidese veebilehekülgede arenduseks kasutatakse Bootstrap raamistikku. Bootstrap on Twitteri poolt loodud raamistik, mis üritab luua sarnast kogemust erinevate platvormide kasutajatele, luues ekraani suuruse muudatustele reageeriva disaini [67]. Bootstrap-i versiooniks on võetud antud ajal viimane versioon – 4.4.1.

Kuna Bootstrap-i kasutus pakub reageerivat kasutajaliidese, kuid piirab selle kohandamist, siis lisaks sellele oli võetud kasutusele kohandatud stiilid, mis olid lisatud *.css* failidesse. Muudatused on tehtud elementide värvides, kuna rakenduse kasutajaliidese disaini loomisel oli arvestatud Bootstrap-i kasutus ning elementide kujud pärinevad sellest raamistikust.

Kõik kasutatavad stiiliga seotud raamistikud ja CSS (*Cascading Style Sheets*) failid hoitakse *TimeableAppWeb* projektis *wwwroot* kataloogis.

4.3 Testimine

Käesoleva lõputöö raames toimub rakenduse manuaalne testimine. Manuaalne testimine toimub uue funktsionaalsuse lisamisel: kasutades erinevaid sisendeid ja tegevusi kontrollitakse viimase muudatuse korrektset toimimist ning vajadusel täiendatakse või muudetakse rakenduse funktsionaalsust.

Peale paljude muudatuste tegemist pöörab arendaja tagasi vanema funktsionaalsuse poole ning käib läbi nii viimaste muudatuste toimimist, kui ka vanemate funktsioonide töökindluse säilitamist. Selline lähenemine annab ülevaadet sellest, kuidas viimati lisatud funktsionaalsus mõjutab rakenduse tööd. Kõikide testide läbiviimine toimub arendusega koos ning on viidud läbi arendaja poolt.

Terve rakenduse testimisega kontrollitakse järgmiste funktsioonide säilitamist:

- Ekraani vaate lisamine, muutmine kasutades erinevaid parameetreid, aktiveerimine.
- Taustapildi muutmine.
- Reklaamipiltide lisamine, kustutamine ning seadistamine.
- Tunniplaani loengute lisamine, muutmine ning tunniplaanist kustutamine.
- Tunniplaani ürituste lisamine, muutmine ning tunniplaanist ürituste kustutamine.
- Peaadministraatoril kasutajate lisamine ja kustutamine, nende õiguste muutmine.
- Sisseloginud kasutaja parooli ja informatsiooni muutmine.
- Tunniplaani automaatne allalaadimine.
- Tunniplaani automaatse uuendamise teenuse õigeaegne väljakutsumine.
- Tunniplaani ja reklaamipiltide õigeaegne kuvamine sisselogimata kasutaja vaates.

Antud rakenduse automaattestimise vajadus on väike, kuna arendusega tegeleb ainult üks inimene, suurte muudatuste tekkimise tõenäosus on väike, kuid rakenduse kasutajate ja arendajate arvu suurendamisel muutub automaattestimine vajalikuks rakenduse osaks.

5 Hinnang loodud veebirakendusele

Lõputöö raames loodud rakendust võib hinnata oodatud ja saavutatud funktsionaalsuse alusel. Hinnangu võib anda tehnoloogiate valikule: selle modernsusele, toetusele arendajate poolt ning kasutajate baasi suurusele. Lisaks võib hinnata rakenduse paindlikkust tulevastele muudatustele.

Arendusprotsessi jooksul töö autor täiendas teadmisi puhta arhitektuuri printsiipidest ning kasutatavatest mustritest selle saavutamiseks. Lisaks täienesid teadmised jQuery raamistikust HTML elementide animeerimisel, kuna varasem kogemus töö autoril selles valdkonnas puudus. Lõputöö raames õpiti muuhulgas ka rakenduse haldamist: omandati teadmised serverite erinevate tüüpide ja nende seadistamise kohta, samuti sai selgeks rakenduse jooksutamine Linux masinas kasutades Nginx veebiserverit.

12. mail 2020 toimus rakenduse tutvustamine Tallinna Tehnikaülikooli IT Kolledži töötajatele, kes on selle rakenduse kasutamises huvitatud. Selle tutvustamise tulemuseks oli saadud tagasiside, mis on lisatud Lisas 3.

5.1 Saavutatud kasutatavus

Nõuded, mis on kirjeldatud epikutena ja kasutajalugudena (Lisa 1), on enamuses täidetud lõputöö kirjutamise protsessis, parallelselt rakenduse arenduse jooksul. Tunniplaani vaate, tunniplaani haldamise ning andmebaasi uuendatavuse teenusega jätkub töö käesoleval hetkel, rakenduse valmimise tähtajaks on määratud 2020-nda aasta kevade lõpp. Ülejäänud funktsionaalsus on toimiv ning vastab püstitatud nõuetele.

Rakendus on hetkel kasutatav ning raskem osa selle arendusest on valminud vastavalt planeeritule. Rakendus oli jagatud kahe ala vahel: tavakasutaja ala ehk tunniplaani vaade ja administraatorite ala. Esilehele oli lisatud sisselogimise funktsionaalsus, mis edukal sisselogimisel suunab ümber administraatori pealeheküljele, kus kohe avaneb ekraani

seadistustega lehekülg. Täiesti toimiv on ekraani seadistamise funktsionaalsus: ekraani puudumisel suunatakse kasutaja ekraani lisamisele, pealeheküljel ekraani olemasolul on võimalus muuta ekraani sätteid, lisada või kustutada sündmuste reklaami, muuta taustapilt ning (de)aktiveerida ekraani. Peaadministraatorile on lisatud võimalus rakendusest lisada ja kustutada kasutajaid ja muuta õigusi.

Nginx proksiserver on konfigureeritud VirtualBox-is asuvas Linux masinas. Rakenduse valmimise hetkel Linux-i masinasse lisatakse rakenduse kompilleeritud kood, migreeritakse andmebaas, eksporditakse Linux masina OVA-failina, mis lisatakse Tallinna Tehnikaülikooli IT Kolledži maja serverisse.

5.2 Kasutatavad tehnoloogiad

Arenduse põhimõtteks oli kasutada laialt levinuid tehnoloogiad, mille dokumenteeritus on põhjalik ning kasutajabaas suur. ASP.NET Core on uusim .NET keskkonna veebirakenduste raamistik, mis vastab täielikult ülaltoodud põhimõtetele. Arenduseks oli valitud 3.1 versioon, mis tuli välja 2019 aasta lõpus ning mis on märgitud pikaajalise toega raamistikuks.

Administreerimise poolest oli valitud Nginx proksiserveri lahendus, millesse .NET rakenduse integreerimine ning koos SQL Server Express andmebaasiga kasutamine on valutu protsess ning .NET rakenduste tugi on tõenäoliselt olemas ka edaspidi.

Kokkuvõtvalt võib öelda, et kasutatavad tehnoloogiad on aktuaalsed antud aja raames ning suure tõenäosusega rakenduse raamistike uuendamine ja toetamine tulevikus võimalik ja lihtne.

5.3 Edasiarendus

Kõige suuremaks prioriteediks antud rakenduse arenduses on puuduvate funktsioonide lisamine veebirakendusse ning reliisi tegemine GitHub koodihaldus keskkonnas. Siia kuuluvad tunniplaani halduse toetamise lisamine, andmebaasis tunniplaani uuendust käivitava teenuse lisamine, tunniplaani vaate kasutajaliidese loomine. Antud funktsionaalsus

on esimese reliisi osaks ning selle lõpetamisega ilmub rakenduse esimene versioon, mis on kasutatav kõikides Tallinna Tehnikaülikooli korpustes.

Võimalikuks edasiarenduseks võiks olla mitme ekraani tugi. Kuna antud lõputöö skoobis on määratud ühe tunniplaani vaate olemasolu, siis teise ekraani lisamine andmebaasi on piiratud ning vaate avamisel puudub ekraani valimise funktsionaalsus. Vaatamata sellele on andmebaasi mudel loodud sellise optsiooni lisamise võimalusega. Suuremat tööd tuleb teha andmete juurdepääsu ja äriloogika kihis ning kasutajaliideses, kuhu tuleb esilehekülje lisada ekraani otsimise identifikaatori väljad.

Hetkel on antud rakenduses osaliselt toetatud eesti ja inglise keel ning andmebaasi salvestatakse ühe keeleversiooni kirjeid. Võimalikuks edasiarenduseks on andmebaasi erinevate salvestatavatele kirjetele keelte toe lisamine ning kasutajaliideses ka teistesse keeltesse staatilistele elementidele tõlgete lisamine.

Vajadusel saab Timeable rakendusele lisada kliendirakendusi, näiteks nutiseadmetele mõeldud rakendus. Kuna ASP.NET Core raamistik on paindlik ja toetab nii MVC mustrit, kui ka RESTful API-t, siis API kontrollerite lisamine on lihtne ning koodigeneraatoriga automatiseeritud. Selline lahendus laiendab rakenduse kasutajakogemust ning võib olla heaks lahenduseks ülikoolides, kus kasutatakse nutiseadmeid toimuvate loengute kuvamiseks.

Rakenduse edasiarendus peab kindlasti kaasama .NET raamistiku uuendamist peale .NET 5.0 versiooni ametlikku reliisi. Kuna arvutitarkvara raamistikuks oli valitud .NET Core 3.1, siis .NET arendajate sõnul selle uuendamine on lihtsam, võrreldes teistega .NET raamistikudega.

Timeable rakenduse kood on avatud ning selle täiendamisega tulevikus võivad tegeleda ka teised arendajad. Lisas 2 on rakenduse koodi link GitHub keskkonnas.

Kokkuvõte

Bakalaureusetöö käigus loodi Timeable rakendus. Rakenduse töövoog on järgmine: õppeinfosüsteemist saadakse värske tunniplaan *iCalendar* faili sisu parsimisega, see analüüsitakse ja grupeeritakse, salvestatakse andmebaasi ning kuvatakse ekraanile. Rakenduse administraatoril on võimalus käsitsi muuta, kustutada ja lisada toimuvaid loenguid, kuid üldiselt rakenduse töövoog on täiesti automatiseeritud. Lisaks andmete muutmisele on administraatoril õigus muuta kliendi vaates taustapilt, asendada tunniplaani uudiste või reklaamiga. Peaadministraatoril on õigus hallata kasutajaid: lisada, kustutada, ning muuta õigusi.

Tunniplaani rakendusega on saavutatud töö alguses püstitatud eesmärk. Esiteks, IT Kolledži multimeediaspetsialist säästab igapäevaselt pool tundi kuni tund aega, kuna tunniplaani sisestamine on automatiseeritud. Teiseks, päeva käigus ilmunud tunniplaani muudatused kuvatakse automaatselt ekraanidele, kust üliõpilased, õppejõud, külalised kiirelt ja mugavalt informatsiooni kätte saavad. Kolmandaks, töö käigus tekitati töötav lahendus, millest saab funktsioonaalsuselt mugavam alternatiiv Risevision rakendusele, mis viimasel ajal on muutunud tasuliseks, kuid funktsionaalsus on jäänud samaks.

Kasutatud kirjandus

[1] „Digital signage doesn't have to be difficult," [Võrgumaterjal]. Available: https://www.risevision.com/about. [Kasutatud 28 02 2020].

[2] Risevision, [Võrgumaterjal]. Available: https://www.risevision.com/pricing?_ga=2.2474096.1308817897.1582890476-1207695917.1581534804. [Kasutatud 28 02 2020].

[3] „Binary Fortress Software," Binary Fortress Software, [Võrgumaterjal]. Available: www.facebook.com/pg/BinaryFortress/about/?ref=page_internal. [Kasutatud 29 02 2020].

[4] „Binary Fortress Software," Binary Fortress Software, [Võrgumaterjal]. Available: www.binaryfortress.com. [Kasutatud 29 02 2020].

[5] „Flipnode LLC," [Võrgumaterjal]. Available: www.linkedin.com/company/flipnode-llc/about/. [Kasutatud 29 02 2020].

[6] B. Kohan, „Guide to Web Application Development," Comentum Corp, [Võrgumaterjal]. Available: https://www.comentum.com/guide-to-web-application-development.html. [Kasutatud 15 03 2020].

[7] „Web programming languages: the best languages for web development," 11 03 2019. [Võrgumaterjal]. Available: https://www.ionos.com/digitalguide/websites/web-development/web-programming-languages/. [Kasutatud 16 03 2020].

[8] „PHP,“ The PHP Group, [Võrgumaterjal]. Available: https://www.php.net/. [Kasutatud 16 03 2020].

[9] „ Developer Survey Results 2019,“ Stack Overflow, [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2019#technology. [Kasutatud 18 03 2020].

[10] „C#,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/C_Sharp. [Kasutatud 18 03 2020].

[11] M. Hornostaiev, „Java, Python, and PHP: Which is Better for Server Backends?,“ Erminesoft, [Võrgumaterjal]. Available: https://erminesoft.com/java-python-and-php-which-is-better-for-server-backends/. [Kasutatud 18 03 2020].

[12] E. Campos, „Is PHP Hard to Learn?,“ [Võrgumaterjal]. Available: https://study.com/academy/popular/is-php-hard-to-learn.html. [Kasutatud 18 03 2020].

[13] CodeGym, „The Best Way To Learn Java Programming,“ [Võrgumaterjal]. Available: https://dev.to/codegym_cc/the-best-way-to-learn-java-programming-2p2c. [Kasutatud 18 03 2020].

[14] C. Achard, „Is JavaScript a hard language to learn?,“ [Võrgumaterjal]. Available: https://dev.to/chrisachard/comment/eidf. [Kasutatud 18 03 2020].

[15] A. Joy, „Is Python Hard to Learn?,“ [Võrgumaterjal]. Available: https://pythonistaplanet.com/is-python-hard-to-learn/. [Kasutatud 18 03 2020].

[16] „NuGet,“ [Võrgumaterjal]. Available: https://www.nuget.org/. [Kasutatud 18 03 2020].

[17] „MSBuild,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/MSBuild. [Kasutatud 21 04 2020].

[18] „.NET Framework,“ Windows, [Võrgumaterjal]. Available: https://dotnet.microsoft.com/download/dotnet-framework. [Kasutatud 22 03 2020].

[19] „.NET Core,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/.NET_Core. [Kasutatud 22 03 2020].

[20] „.NET Core,“ Windows, [Võrgumaterjal]. Available: https://dotnet.microsoft.com/download/dotnet-core. [Kasutatud 22 03 2020].

[21] „Choosing between .NET Core and .NET Framework for server apps,“ Microsoft, 19 06 2018. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server?toc=%2Faspnet%2Fcore%2Ftoc.json&bc=%2Faspnet%2Fcore%2Fbreadcrumb%2Ftoc.json&view=aspnetcore-3.1. [Kasutatud 22 03 2020].

[22] R. Lander, „Introducing .NET 5,“ Microsoft, 06 05 2019. [Võrgumaterjal]. Available: https://devblogs.microsoft.com/dotnet/introducing-net-5/. [Kasutatud 22 03 2020].

[23] L. Talbot, „Future of .NET (.NET 5?) — Microsoft Build 2019 from a .NET developer point of view,“ 28 05 2019. [Võrgumaterjal]. Available: https://medium.com/capgemini-dynamics-365-team/future-of-net-net-5-microsoft-build-2019-from-a-net-developer-point-of-view-7a1158fb0691. [Kasutatud 22 03 2020].

[24] S. Hunter, „Announcing .NET 5 Preview 1,“ Microsoft, 16 03 2020. [Võrgumaterjal]. Available: https://devblogs.microsoft.com/dotnet/announcing-net-5-0-preview-1/. [Kasutatud 22 03 2020].

[25] „Introduction to Razor Pages in ASP.NET Core,“ Microsoft, 12 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-3.1&tabs=visual-studio. [Kasutatud 22 03 2020].

[26] „Choose between ASP.NET 4.x and ASP.NET Core,“ Microsoft, 12 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-3.1#framework-selection. [Kasutatud 22 03 2020].

[27] „.NET Standard,“ Microsoft, 13 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/standard/net-standard. [Kasutatud 22 03 2020].

[28] jonpryor, „.NET Standard Versions,“ 30 08 2019. [Võrgumaterjal]. Available: https://github.com/dotnet/standard/blob/master/docs/versions.md. [Kasutatud 22 03 2020].

[29] R. Oamkumar, „MVC vs. Web API: Which ASP.NET technology should you use?,“ YUHIRO Group, 10 04 2018. [Võrgumaterjal]. Available: https://www.software-developer-india.com/mvc-vs-web-api-which-asp-net-technology-should-you-use/. [Kasutatud 23 03 2020].

[30] T. M. D. V. Rick Anderson, „Razor syntax reference for ASP.NET Core,“ Microsoft, 12 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-3.1. [Kasutatud 23 03 2020].

[31] „JavaScript,“ Wikipedia, [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/JavaScript. [Kasutatud 23 03 2020].

[32] „What are the best C# IDEs?,“ [Võrgumaterjal]. Available: https://www.slant.co/topics/4118/~c-ides. [Kasutatud 23 03 2020].

[33] „Toolbox Subscription,“ JetBrains, [Võrgumaterjal]. Available: https://www.jetbrains.com/rider/buy/#personal?billing=yearly. [Kasutatud 23 03 2020].

[34] „ReSharper,“ Wikipedia, [Võrgumaterjal]. Available: https://ru.wikipedia.org/wiki/ReSharper. [Kasutatud 23 03 2020].

[35] „Compare Visual Studio 2019 Editions,“ Microsoft, [Võrgumaterjal]. Available: https://visualstudio.microsoft.com/vs/compare/. [Kasutatud 23 03 2020].

[36] W. Hardwick-Smith, „Visual Studio or Rider? Comparing the old and the new,“ 06 07 2018. [Võrgumaterjal]. Available: https://medium.com/@whardwicksmith/visual-studio-vs-rider-904d63d88f53. [Kasutatud 23 03 2020].

[37] „What are the best hosted version control services?,“ [Võrgumaterjal]. Available: https://www.slant.co/topics/153/~best-hosted-version-control-services. [Kasutatud 25 03 2020].

[38] „Git,“ Wikipedia , [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Git. [Kasutatud 25 03 2020].

[39] „GitHub is how people build software,“ GitHub Inc., [Võrgumaterjal]. Available: https://github.com/about. [Kasutatud 25 03 2020].

[40] „Pricing plans for all developers,“ GitHub Inc, [Võrgumaterjal]. Available: https://github.com/pricing. [Kasutatud 25 03 2020].

[41] „Configuring dotnet CLI for use with GitHub Packages,“ GitHub Inc, [Võrgumaterjal]. Available: https://help.github.com/en/packages/using-github-packages-with-your-projects-ecosystem/configuring-dotnet-cli-for-use-with-github-packages. [Kasutatud 25 03 2020].

[42] „Plans and pricing,“ Atlassian, [Võrgumaterjal]. Available: https://bitbucket.org/product/pricing. [Kasutatud 25 03 2020].

[43] „Jira Software,“ Atlassian, [Võrgumaterjal]. Available: https://www.atlassian.com/software/jira/bitbucket-integration. [Kasutatud 25 03 2020].

[44] „The entire DevOps lifecycle in one application.,“ GitLab, [Võrgumaterjal]. Available: https://about.gitlab.com/stages-devops-lifecycle/. [Kasutatud 25 03 2020].

[45] „GitLab.com Feature Comparison,“ GitLab, [Võrgumaterjal]. Available: https://about.gitlab.com/pricing/gitlab-com/feature-comparison/. [Kasutatud 25 03 2020].

[46] „Types of databases,“ Arjun Panwar, 07 03 2020. [Võrgumaterjal]. Available: https://www.c-sharpcorner.com/UploadFile/65fc13/types-of-database-management-systems/. [Kasutatud 25 03 2020].

[47] „DB-Engines Ranking,“ 03 2020. [Võrgumaterjal]. Available: https://db-engines.com/en/ranking. [Kasutatud 25 03 2020].

[48] „Comparing document-oriented and relational data,“ [Võrgumaterjal]. Available: https://developer.couchbase.com/documentation/server/3.x/developer/dev-guide-3.0/compare-docs-vs-relational.html. [Kasutatud 29 03 2020].

[49] „Oracle Database,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Oracle_Database. [Kasutatud 26 03 2020].

[50] „Oracle Database 19c,“ Oracle, [Võrgumaterjal]. Available: https://www.oracle.com/database/technologies/. [Kasutatud 26 03 2020].

[51] „Oracle NoSQL Database Cloud Service,“ Oracle, [Võrgumaterjal]. Available: https://www.oracle.com/database/nosql-cloud.html. [Kasutatud 26 03 2020].

[52] C. Arsenault, „The Pros and Cons of 8 Popular Databases,“ 20 04 2017. [Võrgumaterjal]. Available: https://www.keycdn.com/blog/popular-databases. [Kasutatud 26 03 2020].

[53] „Oracle Technology Global Price List,“ Oracle, 01 03 2020. [Võrgumaterjal]. Available: https://www.oracle.com/assets/technology-price-list-070617.pdf. [Kasutatud 26 03 2020].

[54] „MySQL,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/MySQL. [Kasutatud 26 03 2020].

[55] „SQL,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/SQL#Procedural_extensions. [Kasutatud 29 03 2020].

[56] „MariaDB Server: The open source relational database,“ MariaDB Foundation, [Võrgumaterjal]. Available: https://mariadb.org/. [Kasutatud 29 03 2020].

[57] „Transact-SQL,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Transact-SQL. [Kasutatud 26 03 2020].

[58] „SQLite,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/SQLite. [Kasutatud 29 03 2020].

[59] „Appropriate Uses For SQLite,“ SQLite, [Võrgumaterjal]. Available: https://www.sqlite.org/whentouse.html. [Kasutatud 29 03 2020].

[60] „The database for modern applications,“ MongoDB Inc., [Võrgumaterjal]. Available: https://www.mongodb.com/. [Kasutatud 29 03 2020].

[61] „Host and deploy ASP.NET Core,“ Microsoft, 07 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/?view=aspnetcore-3.1#set-up-a-process-manager. [Kasutatud 29 03 2020].

[62] „Host ASP.NET Core on Windows with IIS,“ Microsoft, 07 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/?view=aspnetcore-3.1. [Kasutatud 29 03 2020].

[63] „Host ASP.NET Core on Linux with Apache,“ Microsoft, 05 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-apache?view=aspnetcore-3.1. [Kasutatud 29 03 2020].

[64] „Host ASP.NET Core on Linux with Nginx,“ Microsoft, 05 02 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-3.1. [Kasutatud 29 03 2020].

[65] A. Kuusmaa ja A. Käver, „R.C.Martini "Clean Architecture",“ 30 12 2018. [Võrgumaterjal]. Available: https://github.com/akaver/CleanArchitectureDemo/blob/master/Docs/Clean%20Architecture.pdf. [Kasutatud 31 03 2020].

[66] „dotnet aspnet-codegenerator,“ MIcrosoft, 07 04 2019. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/tools/dotnet-aspnet-codegenerator?view=aspnetcore-3.1. [Kasutatud 04 04 2020].

[67] „Introduction,“ [Võrgumaterjal]. Available: https://getbootstrap.com/docs/4.4/getting-started/introduction/. [Kasutatud 04 04 2020].

[68] „ASP.NET MVC,“ Wikipedia, [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/ASP.NET_MVC. [Kasutatud 13 04 2020].

[69] „AutoMapper,“ AutoMapper, [Võrgumaterjal]. Available: https://automapper.org/. [Kasutatud 15 04 2020].

[70] „Entity Framework Core,“ Microsoft, 27 10 2016. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/ef/core/. [Kasutatud 15 04 2020].

[71] „Indexes,“ Microsoft, 16 12 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/ef/core/modeling/indexes. [Kasutatud 15 04 2020].

[72] „Package Manager Console Commands for Migrations,“ [Võrgumaterjal]. Available: https://www.entityframeworktutorial.net/efcore/pmc-commands-for-ef-core-migration.aspx. [Kasutatud 15 04 2020].

[73] „HTTP-küpsis,“ WIkipedia, [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/HTTP-k%C3%BCpsis. [Kasutatud 26 04 2020].

[74] „EU cookie law - a right to privacy,“ 01 11 2019. [Võrgumaterjal]. Available: https://www.cookiebot.com/en/cookie-law/. [Kasutatud 26 04 2020].

[75] „Customize your build,“ Microsoft, 13 06 2019. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/visualstudio/msbuild/customize-your-build?view=vs-2019. [Kasutatud 22 04 2020].

[76] M. Eland, „Safer Code with C# 8 Non-Null Reference Types,“ 14 09 2019. [Võrgumaterjal]. Available: https://dev.to/integerman/safer-code-with-c-8-non-null-reference-types-4f2c. [Kasutatud 22 04 2020].

[77] „Жизненный цикл зависимостей,“ 07 11 2019. [Võrgumaterjal]. Available: https://metanit.com/sharp/aspnet5/6.2.php. [Kasutatud 16 04 2020].

[78] „Overview of ASP.NET Core Security,“ Microsoft, 24 10 2018. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-3.1. [Kasutatud 18 04 2020].

[79] „ aspnet/Identity,“ GitHub Inc., 20 06 2017. [Võrgumaterjal]. Available: https://github.com/aspnet/Identity/blob/rel/2.0.0/src/Microsoft.Extensions.Identity.Core/PasswordHasher.cs. [Kasutatud 18 04 2020].

[80] „Raw SQL Queries: Passing parameters,“ Microsoft, 08 10 2019. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/ef/core/querying/raw-sql#passing-parameters. [Kasutatud 18 04 2020].

[81] R. Anderson, „Prevent Cross-Site Scripting (XSS) in ASP.NET Core,“ Microsoft, 02 10 2018. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-3.1. [Kasutatud 18 04 2020].

[82] „Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core: ASP.NET Core antiforgery configuration,“ Microsoft, 05 12 2020. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery?view=aspnetcore-3.1#aspnet-core-antiforgery-configuration. [Kasutatud 18 04 2020].

[83] „Prevent open redirect attacks in ASP.NET Core,“ Microsoft, 07 07 2017. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/security/preventing-open-redirects?view=aspnetcore-3.1. [Kasutatud 18 04 2020].

[84] A. Thahir, „Which Is Superior: Saving Files In A Database Or In A File System?,“ [Võrgumaterjal]. Available: https://stackoverflow.com/questions/3748/storing-images-in-db-yea-or-nay/3751#3751. [Kasutatud 19 04 2020].

[85] „Microsoft.VisualStudio.Web.CodeGeneration.Design,“ NuGet , [Võrgumaterjal]. Available: https://www.nuget.org/packages/Microsoft.VisualStudio.Web.CodeGeneration.Design/. [Kasutatud 18 04 2020].

[86] „Microsoft.Extensions.Identity.Stores,“ NuGet, [Võrgumaterjal]. Available: https://www.nuget.org/packages/Microsoft.Extensions.Identity.Stores/. [Kasutatud 18 04 2020].

Lisa 1 - Nõudeid kirjeldavad kasutajalood

Epik 1: Üliõpilasena tahan näha kehtivat tunniplaani, mis oleks mugavalt loetav ja eristatud värvidega, et kiiremini leida vajalikku informatsiooni.

- **K1:** Üliõpilasena tahan näha tänase päeva kehtivat tunniplaani, et olla kursis viimaste muudatustega.
- **K2:** Üliõpilasena tahan eristada loenguid, mis algavad vähem kui 15 ja 5 minuti pärast, et planeerida loenguni jäänud aeg.
- **K3:** Üliõpilasena tahan eristada juba alanud loenguid, et teada, kas hilinen loengule või mitte.
- **K4:** Kasutajana tahan näha tunniplaanis ainult alanud või algavaid loenguid, et lõppenud loengud ei segaks vaadet.
- **K5:** Kasutajana tahan televiisorist näha kuupäeva ja hetke kellaaega, et see informatsioon oleks koos tunniplaaniga kättesaadav ilma teiste seadmete (telefon, kell, arvuti) kasutamist.

Epik 2: Timeable kasutajana tahan näha informatsiooni loengu kohta, et leida sealt kõike vajalikku.

- **K1:** Kasutajana tahan tunniplaanis näha ainenimetust ja ainekoodi, mille järgi loeng oleks lihtne leitav tunniplaanist.
- **K2:** Kasutajana tahan teada, mis kell algab ja lõpeb iga loeng, et teada, kui pikk on loeng ning millal õppejõud ja ruumid on hõivatud.
- **K3:** Kasutajana tahan näha, kes viib läbi iga loengu, et teada millal õppejõud on hõivatud.
- **K4:** Kasutajana tahan näha, mis kabinetis mingi loeng toimub, et teada kus loeng toimub ning millal ruumid on hõivatud.

- **K5:** Halva nägemisega kasutajana tahan hästi loetavat teksti, et informatsioon oleks nähtav ka minule.

Epik 3: Timeable rakenduse kasutajana soovin teada tunniplaanist informatsiooni ürituste kohta, et saaks võtta neist osa.

- **K1:** Kasutajana tahan teada, mis üritused antud majas lähiajal toimuvad, et valida endale huvitavaid üritusi, kust osa võtta.
- **K2:** Kasutajana tahan teada, millal ja mis kell üritused on, et planeerida enda aega võttes arvesse üritust, kuhu tahan minna.
- **K3:** Kasutajana soovin teada, kus toimuvad mind huvitavad üritused, et teada kuhu peab tulema ürituse alguseks.

Epik 4: Timeable administraatorina soovin muuta enda andmed, et nad oleksid kehtivad.

- **K1:** Sisse loginud kasutajana, soovin muuta e-posti, millega minu konto on seotud, et kõik teated oleksid saadetud õigele e-posti aadressile juhul, kui see on vahetunud.
- **K2:** Sisse loginud kasutajana soovin muuta enda parooli, et parooli unustamisel või selle äraarvamise riski tekkimisel oleks võimalik seda vahetada.
- **K3:** Sisse loginud kasutajana soovin muuta minu ees- või perekonnanime, et selle muutmise puhul need oleksid ajakohased.

Epik 5: Timeable administraatorina soovin hallata ekraanile kuvatavat informatsiooni, et lisada ja muuta informatsiooni vastavalt vajadusele.

- **K1:** Timeable administraatorina soovin lisada uue ekraani, kui see on puudu.
- **K2:** Timeable administraatorina soovin lisada reklaamipilte ekraanile, et teavitada kasutajaid üritustest, töötubadest, töövõimalustest, jne.
- **K3:** Timeable administraatorina soovin kustutada reklaamipilte ekraanilt, et reklaamitava aegumisel reklaamipilt poleks enam näidatud.
- **K4:** Timeable administraatorina soovin määrata, kui kaua ekraanil püsib iga reklaampilt, et kasutajal oleks piisavalt aega informatsiooniga tutvuda.

- **K5:** Timeable administraatorina soovin määrata, kui kaua ekraanil püsib tunniplaan, et kasutajal oleks piisavalt aega leida vajalikku informatsiooni loengute kohta.
- **K6:** Timeable administraatorina soovin määrata ekraani taustapilti, et teha visuaalselt Timeable rakendust atraktiivseks.
- **K7:** Timeable administraatorina soovin määrata, millise Tallinna Tehnikaülikooli korpuse tunniplaan kuvatakse, et oleks võimalus kasutada Timeable rakendust ka teistes TalTech korpustes.

Epik 6: Timeable administraatorina soovin muuta informatsiooni loengu või ürituse kohta, kui nende informatsioon on puudulik või vale.

- **K1:** Timeable administraatorina soovin kustutada loenguid tunniplaanist, kui need loengud jüäävad ära või toimuvad teises korpuses.
- **K2:** Timeable administraatorina soovin lisada loenguid tunniplaani, kui on lisatud uus loeng, mille kohta informatsioon tunniplaanis puudub.
- **K3:** Timeable administraatorina soovin muuta informatsiooni loengu kohta, kui selle toimumiskoht, kellaaeg, läbiviiv õppejõud või muu informatsioon on muutunud.
- **K4:** Timeable administraatorina soovin lisada üritusi tunniplaani, kui ürituse läbiviimine on kokkulepitud ning toimimise koht, aeg ja ürituse nimetus on teada.
- **K5:** Timeable administraatorina soovin muuta informatsiooni ürituse kohta, kui ürituse toimumise koht, kellaaeg või selle nimetus on muutunud.
- **K6:** Timeable administraatorina soovin kustutada üritust, kui selle läbiviimine on ära jäetud.

Epik 7: Peaadministraatorina tahan hallata kasutajaid ja nende õigusi, et anda kasutajatele eri informatsiooni haldamise võimalust.

- **K1:** Peaadministraatorina soovin lisada kasutajaid, et nendel oleks võimalus hallata Timeable rakenduse kasutajatele kuvatavat informatsiooni.
- **K2:** Peaadministraatorina soovin kustutada kasutajaid süsteemist, et võtta ära õigusi kasutajatelt, kui nende roll ülikoolis on muutunud.

- **K3:** Peaadministraatorina soovin anda kasutajale ekraanile kuvatava informatsiooni haldamise õigust.
- **K4:** Peaadministraatorina soovin anda tunniplaani informatsiooni haldamise õigust.
- **K5:** Peaadministraatorina soovin anda kasutajale ürituste haldamise õigust.

Lisa 2 – Rakenduse versioonihaldus

Reliisid: https://github.com/shade-less/Timeable

Lisa 3 – Rakenduse tulevaste kasutajate tagasiside

Allpool on väljatoodud tulevaste kasutajate kaks tagasisidet. Tööautori poolt oli tehtud rakenduse esitus kõige rohkem huvitatud Tallinna Tehnikaülikooli IT Kolledži töötajatele, kelleks on turundusjuht Marje Meenov ja multimeediaspetsialist Joel Kivi.

Tervist!

Minu tagasiside:
Töö eesmärgiks oli teostada IT Kolledžile lahendus millega näidata infoekraanidele tunniplaani.
Töö teostaja lisas omalt poolt rohkem võimalusi kui esialgselt planeeritud oli.

Lugupidamisega,
Joel Kivi

TalTech IT Kolledži IT süsteemide arenduse õppekava tudeng Elina Antonova diplomitöö raames loodud rakendus on suurepärane. Rakendus lahendab ära Tallinna Tehnikaülikooli ja IT Kolledži liitumise järgselt tekkinud probleemi - IT Kolledži infoekraanidel korrektse info kuvamine, sh ühildumine tehnikaülikooli õppeinfosüsteemiga.
Senine infoekraanide lahendus ei võimaldanud korrektse info kuvamist ja nõudis erakordselt palju aega, sest suur osa andmeid oli vaja käsitsi sisestada.

Elina Antonova loodud rakendus võtab automaatselt andmed tehnikaülikooli õppeinfosüsteemist ja ei vaja enam infoekraanide administraatori poolt lisatööd.
Rakenduse administraatoril/haldajal on lihtne vajadusel muuta ja lisada andmeid, mida infoekraanidel kuvada. Lisaks on rakenduse administraatori/haldaja töölaua vaade lihtsa ja loogilise ülesehitusega ja seal saab mugavalt toimetada iga töötaja, sest see ei eelda mahukaid eelteadmisi, piisab kasutusjuhendi läbitöötamisest või peaadministraatori juhendamisest.

Lisaks tunniplaani ja sündmuste kuvamisele saab ekraanidele lisada info/reklaamplakateid ning panna need tunniplaaniga vahelduma. Mugavalt saab valida aja, mitme sekundi/minuti tagant tunniplaan digiplakatiga vaheldub. Visuaalselt annab palju juurde ka tunniplaani vaates inforidade värvilahendus. Saab eristada äsja alanud sündmusi, loenguid ja peagi tulevaid sündmusi. Positiivne on see, et juba toimunud sündmused kustuvad ekraanidelt ära ja ei koorma ekraanipinda kasutu infoga. Siiani see võimalus puudus ning ajas infotarbijad segadusse.

Kõige enam meeldib mulle see, et tegemist on praktilise lahendusega ja tudeng on loonud midagi sellist, mida tegelikult ka vaja läheb – lahendab ära päris elu probleemi. Ning on teinud seda suuremas mahus, kui diplomitöö nõuab. Igati tubli tulemus ja jään ootama, et saaksime infoekraanide rakendust peagi kasutama hakata.

Oluline lisandväärtus on ka see, et antud rakendust saab kasutada vajadusel kõikides TalTechi õppehoonete ekraanidel, mis kuvavad tunniplaaniinfot, mille algandmed tulevad TalTechi õpeinfosüsteemist. Usun, et rakendust veidi kohendades on seda võimalik kasutada ka teistes õppeasutustes. Ehk tuleviku potentsiaali rakenduse edasiarendamisel jagub küllaga.

Lisaks soovitan tulevikus, kui on plaanis rakendust edasi arendada, kaaluda videote lisamise võimalust. Ehk ekraanidel vahelduksid tunniplaan, digiplakatid ja videod ning nende kuvamist oleks lihtne kombineerida.

Lühidalt kokkuvõttes - suurepärane, praktilise väärtusega tulemus.

Marje Meenov, TalTech IT Kolledži turundusjuht