TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Carina Ruut 206141IADB

Google Drive’i põhine piltide jagamise veebirakendus

Bakalaureusetöö

Juhendaja: German Mumma

MSc Äriinfotehnoloogia

Tallinn 2023

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: Carina Ruut

14.05.2023

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
GB	Gigabait
HTML	*HyperText Markup Language*, veebilehtede märgendamise keel
HTTP	*HyperText Transfer Protocol,* hüperteksti edastus protokoll
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
MVP	*Minimum viable product*, minimaalne töötav toode
.NET	Microsofti loodud raamistik
Vue	JavaScripti raamistik

Annotatsioon

Käesoleva lõputöö eesmärgiks on luua kasutajasõbralik piltide jagamise veebirakenduse minimaalne töötav toode, mis kasutab Google Drive'i piltide haldamiseks. Minimaalse töötava toote nõuded on pandud paika vastavalt olemasolevates lahendustes esinevatest probleemidest kasutajale.

Loodud minimaalne töötav toode on arendatud vastavalt nõuete ja rakenduse spetsiifikast paika pandud epikute ja kasutajalugude põhjal. Rakendus vastab nõuetele ning on tulevikus edasi arendatav. Tagarakendus on kirjutatud C# koos .NET raamistikuga ning esirakendus TypeScriptis Vue raamistikuga. Rakendus kasutab kasutajate sisselogimiseks Google OAuth API-t ja piltidega seotud toiminguteks Google Drive API-t.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 6 peatükki, 18 joonist, 3 tabelit.

Abstract

Google Drive Based Picture Sharing Web Application

The aim of this thesis is to create a user-friendly minimum viable product of a web application for sharing pictures, which uses Google Drive to manage the pictures. The requirements for the minimum viable product have been established based on the problems identified in existing solutions.

The created minimum viable product has been developed according to the requirements and specifications defined by epics and user stories. The application fulfills the requirements and can undergo further development in the future.

The back-end is written in C# using the .NET framework, and the front-end is written in TypeScript using the Vue framework. The application uses Google OAuth API for user authentication and Google Drive API for image-related operations.

The thesis is written in Estonian and consists of 35 pages of text, 6 chapters, 18 figures, and 3 tables.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva lõputöö eesmärgiks on luua kasutajasõbralik piltide jagamise veebirakenduse minimaalne töötav toode, mis lubab kasutajal luua oma veebileht kasutades Google Drive'i oma piltide haldamiseks, et kasutajatel oleks lihtne ja kiire viis oma pilte jagada.

Töö põhjaks kasutatakse ära Google poolt loodud API lahendusi. Käsitluse alla tulevad Google OAuth API ja Google Drive API, millest esimene omab funktsionaalsust autentida kasutaja kasutades olemasolevat Google kontot ning teine lubab luua ühenduse kasutaja Google Drive'iga.

Tänapäeval on väga populaarne oma piltide jagamine internetis teiste inimestega, kuid olemasolevad lahendused võivad omada mitmeid probleeme:

- piltide kvaliteet on pildi üleslaadimisel halvenenud;
- veebihalduskeskkonnad ja veebilehe teenuste pakkujad võivad osutuda inimesele, kes pole varem sellega kokku puutunud, keeruliseks;
- veebihalduskeskkondade ja veebilehtede teenuste pakkujad võivad küsida suuremate mälumahtude ja pistikprogrammide eest tasu.

Lõputöö on jaotatud kuueks peatükiks. Esimeses peatükis tutvustatakse tööd ning püstitatakse eesmärk. Teises peatükis analüüsitakse probleemi ning pannakse paika rakenduse nõuded. Kolmandas peatükis kirjeldatakse kasutatavaid tehnoloogiaid, tehakse arhitektuuri valik ning kirjeldatakse andmebaasi skeem. Neljas peatükk annab ülevaate lõputöö raames loodud rakenduse minimaalsest töötavast tootest. Viies peatükis analüüsitakse loodud rakenduse tulemusi võrreldes neid paika pandud nõuetega ning tutvustatakse loodud rakendust. Viimases peatükis võetakse töö kokku.

1.1 Lõputöö aktuaalsus

Pilte jagatakse väga tihti internetis. Selleks võivad olla tavainimesed, hobi- ja professionaalsed fotograafid, disainerid (digitaalne kunst ja graafika). Lisaks sellele hoiustavad 2022. aastal ligi 71% inimestest oma pilte pilves ning kõige populaarseim pilv on Google Drive [1].

Kuna pilve kasutamine piltide hoiustamiseks on väga populaarne ükskõik, millisel inimesel, siis on lõputöö raames ära kasutatud just seda teadmist ja oskust kasutada pilve teenuseid, et luua võimalikult lihtne ja väheseid lisa tehnoloogilisi teadmisi nõudev veebirakendus, mida oleks võimeline kasutama iga inimene.

1.2 Lõputöö eesmärgini jõudmine

Lõputöö eesmärgiks on luua kasutajasõbralik piltide jagamise veebirakenduse minimaalne töötav toode, mis kasutab Google Drive'i piltide haldamiseks.

Analüüsi käigus täpsustatakse püstitatud probleemi ning uuritakse olemasolevaid lahendusi ja nende piiranguid. See järel pannakse paika veebirakenduse nõuded.

Enne tarkvaraarendust pannakse paika ka tehnoloogiad ja arhitektuur. Järgmiseks luuakse andmebaasi skeem ja hakatakse rakenduse funktsionaalsust kasutajalugude abil arendama.

Eesmärk loetakse täidetuks, kui rakenduse funktsionaalsus saavutab minimaalse töötava toote nõuded, mis täpsustatakse jaotises 2.2.

2 Probleemi analüüs

Tänapäeval jagatakse väga palju internetis pilte. Piltide jagamine on nii populaarne, et iga päev jagatakse ligikaudu 2,1 miljardit pilti Facebookis ja 1,3 miljardit Instragramis [2]. Piltide jagamise põhjused on väga erinevad, nendeks võivad olla näiteks mingisuguse väärtuse või meelelahutusliku sisu teistega jagamine, enese kirjeldamine või eneseteostus [3].

Pilte hoiustatakse erinevatel viisidel. Neid hoitakse oma seadmes, välisel kõvakettal, mälukaardil või näiteks pilves. Pilves piltide hoidmine on soovituslikult kõige parem meetod, et pildid oleks turvaliselt hoiustatud. Seadmed võivad ootamatult katki minna, mälukaardid ära kaduda, kõvakettad võivad olla tundlikud füüsilistele kahjustustele [4]. Kõige populaarsem pilveteenus aastal 2022 oli Google Drive [1].

2.1 Olemasolevate lahenduste puudused

Lõputöös loodud rakendusele on palju alternatiive, näiteks erinevad sotsiaalmeedia platvormid, veebihalduskeskkonnad ja veebilehe teenuse pakkujad. Järgmiselt tuuakse välja, miks võivad osutuda olemasolevad lahendused keeruliseks inimestele, kes pole varem kokku puutunud oma veebilehe tegemisega ja millised on nende puudused.

2.1.1 Piltide kvaliteedi muutus

Väga paljusid sotsiaalmeedia kanaleid kasutatakse just piltide jagamiseks teistega. Need ei nõua varasemaid teadmisi erinevatest tehnoloogiatest. Kõige populaarsemad sotsiaalmeedia kanalid, kus jagatakse peamiselt pilte, on Instagram ja Facebook [5]. Kuid piltide kvaliteet võib vale resolutsiooni korral üleslaadides halveneda [6].

Instagramil ja Facebookil on soovituslikud piltide resolutsioonid, mis piiravad suuremate piltide üleslaadimist ilma optimeerimiseta. Instagram ei optimeeri pilte kui pildid vastavad järgmistele nõuetele: pildi laius on 360-1080 pikslit ja suhe 1.91:1 ning 4:5 vahel. Alla selle

või üle selle pildid lõigatakse, tõstetakse või alandatakse resolutsiooni. Facebookis on soovitatud piltide suurused arvutitele 479 x 246 pikslit, mobiilidele minimaalseks laiuseks 320 pikslit ning üldine soovitus on 1080 x 1350 pikslit [7, 8].

Seatud piirangud eeldavad seda, et kui kasutaja soovib oma pilte jagada hea kvaliteediga, tuleb tal esmalt neid ise töödelda vastavalt ette antud nõuetele.

2.1.2 Liigsed funktsionaalsused ja pistikprogrammid

Kõige tuntumad veebihalduskeskkonnad on WordPress, Joomla ja Drupal [9]. Neid kasutatakse oma veebilehtede loomiseks, olgu need siis blogid, foorumid või veebipoed.

Lisaks omavad veebihalduskeskkonnad administraatori paneeli, mida saab kasutada veebilehe seadistamiseks. Administraatori paneel võimaldab hallata postitusi, meediat, lehekülgi ja kommentaare.

Veebihalduskeskkondadel on väga palju pistikprogramme. WordPressil on olemas ka pistikprogrammide lisamise võimalus. Tasuta pistikprogramme on üle 60000 ja üle 10000 tasuta teema. Lisaks neile on veel ka tasulisi [10, 11]. Drupalil on üle 40000 laienduse ja 2500 teema [12].

Laiendused on mõeldud veebilehele lisa funktsionaalsuste lisamiseks ja kujundamiseks, kuid siiski võib olla keeruline nii suure hulga seest valida ja leida just see õige.

Lisa funktsionaalsused ja palju pistikprogramme muudavad kasutajale, kes soovib oma pilte jagada, asja keeruliseks, kirjuks ning üleüldsegi ebavajalikuks.

2.1.3 Veebilehe majutamine

Veebihalduskeskkondi ja oma veebilehte kasutades tuleb kasutajal läbi mõelda ka, kus veebilehte majutada ehk hostida. Erinevad halduskeskkonnad pakuvad tasulisi teenuseid, mis selle eest vastutavad, kuid siiski tuleb mõelda, kas tasuline alternatiiv sotisaalmeediale, on tõesti nii suure kasuteguriga. Lisaks tuleb ka osta endale domeen ehk veebileheaadress.

Oma veebilehte on võimalik majutada ka isikliku serveri peal, kuid see nõuab tehnoloogilisi teadmisi selles vallas. Lisaks tuleb kasutajal endale ka sel juhul isiklik server osta.

2.1.4 Mäluruum piltide hoiustamiseks

Kasutades veebileheks WordPressi lahendust on kasutajal tasuta versiooni puhul 1 GB tasuta mäluruumi. Suuremate piltide korral saab mäluruum kiiresti täis ning tuleb valida variant tasulisest versioonidest. Oluline on see, et WordPressil on kaks erinevat versiooni, millest üks on hallatud majutusega ning teine on lihtsalt tarkvara, mis võivad kergesti segamini minna [13].

WordPressile on võimalik piiramatult mäluruumi saada, kui kasutatakse WordPress.orgi ehk ainult WordPressi tarkvara, mis paigaldatakse oma serveri peale. Kuid sellel juhul võib tekkida probleem, et inimesel ei ole piisavalt teadmisi, et ise oma veebileht majutada ning lisa mäluruum jääb saamata.

2.2 Veebirakenduse nõuded

Igal rakendusel peavad olema paika pandud kindlad nõuded, mille põhjal hakatakse rakendus arendama. Nõuded võivad olla funktsionaalsed kui ka mittefunktsionaalsed. Funktsionaalsed nõuded on nõuded, mis aitavad kasutajal saavutada soovitud eesmärki [14]. Mittefunktsionaalsed nõuded on näiteks kättesaadavus, laiendatavus ja kasutatavus [15].

2.2.1 Funktsionaalsed nõuded

Lõputöö käigus loodav rakendus peab vastama funktsionaalsetele nõudmistele, et tagada rakenduse kasulikkus ja eesmärgipärasus. Funktsionaalsete nõuete kirjeldamisel on lähtutud jaotises 2.1 toodud puudustest, mis lahendaks välja toodud probleeme. Rakendus otsustati teha Google Drive'i põhjal, sest sellel on kõige rohkem algselt tasuta mäluruumi võrreldes teiste pilveteenustega. Google Drive'il on alustuseks 15 GB-d mäluruumi, Dropboxil kaks GB-d, OneDrive viis GB-d [16].

Järgmisena on välja toodud nõuded, mida peab rakendus täitma:

- Rakendus pakub võimalust kuvada pilte võimalikult hea kvaliteediga.
- Rakenduse põhifunktsionaalsuseks on piltide jagamine.
- Rakendus peab omama võimalust luua kasutaja.
- Rakendus peab meeles hoidma kasutaja andmeid.
- Rakendus peab kasutama kasutaja Google Drive'i piltide hoidmiseks.

2.2.2 Mittefunktsionaalsed nõuded

Loodaval rakendusel on ka mittefunktsionaalseid nõudeid, mis on loetletud järgmisena.

- Rakenduse laiendatavus erinevate pilve teenustega.
- Rakendust peab olema võimalik kasutada igas tänapäevases veebibrauseris.
- Veebileht peab kuvama sisu õiges formaadis, kas arvutis või nutitelefonis.

2.2.3 Epikud ja kasutajalood

Lõputöös loodava rakenduse kirjutamiseks on valitud agiilne metoodika. Agiilne metoodika koosneb väikestest arenduse iteratsioonidest. Iteratsioon hõlmab endas ühe funktsionaalsuse arendamist. Funktsionaalsuste kirjeldamiseks on loodud epikud ja kasutajalood.

Epikud kirjeldavad rakenduse suuremaid põhifunktsionaalsusi, millest põhinevad kasutajalood. Kaustajalood kirjeldavad funktsionaalsusi, mida peab vastava epiku põhifunktsionaalsus endas omama. Tabelites 1 ja 2 on kirjeldatud epikud ja kasutajalood, mida loetakse ridade haaval vasakult paremale ning need koosnevad lauseosadest, mis vastavad küsimustele kellena, mida, miks. Kasutajalugude tabelis on viidatud epikule kasutades epiku ID.

Tabel 1. Epikud.

Epik ID	**Roll <kellena>**	**soovin ma <teha midagi>,**	**et saaksin <mida>**
1	Kasutajana	soovin ma sisse logida oma Google kontoga,	et saaksin ennast autentida.
2	Sisse logitud kasutajana	soovin ma kasutada oma Google Drive'i,	et saaksin seal olevaid pilte oma veebilehel kuvada.
3	Sisse logitud kasutajana	soovin ma luua oma profiili,	et saaksin oma informatsiooni kuvada ja muuta.
4	Kasutajana	soovin ma näha teiste profiile,	et saaksin nende pilte vaadata.

Tabel 2. Kasutajalood.

Epik ID	**Prioriteet**	**Kasutaja-loo ID**	**Roll <kellena>**	**soovin ma <teha midagi>,**	**et saaksin <saavutada midagi>**
1	Kõrge	1	Kasutajana	soovin ma valida oma Google'i kontode vahel,	et saaksin sisse logida sobiva kasutajaga.
1	Kõrge	2	Sisse logitud kasutajana	soovin ma automaatselt osa informatsiooni Google'i kontost saada,	et saaksin kiiremini oma kasutaja seadistada.
2	Kõrge	3	Sisse logitud kasutajana	soovin ma valida milliseid pilte oma veebilehele laadida,	et saaksin jagada ainult soovitud informatsiooni.

Epik ID	Prioriteet	Kasutaja-loo ID	Roll <kellena>	soovin ma <teha midagi>,	et saaksin <saavutada midagi>
2	Madal	4	Sisse logitud kasutajana	soovin ma sorteerida oma pilte kaustadesse,	et saaksin kergemini neid hallata ja sorteerituna veebilehel kuvada.
2	Kõrge	5	Sisse logitud kasutajana	soovin ma oma pilte veebilehel kuvada,	et teised saaksid neid vaadata.
3	Kõrge	7	Sisse logitud kasutajana	soovin ma lisada enda leheküljele lühitutvustust,	et teised teaksid, millega ma tegelen või kes ma olen.
3	Kõrge	8	Sisse logitud kasutajana	soovin ma enda veebilehele nime panna,	et saaksin anda informatsiooni oma lehekülje kohta paremini (näiteks, kui olen ettevõte).
4	Kõrge	9	Kasutajana	soovin ma näha teiste veebilehti,	et saaksin vaadata nende pilte.
4	Madal	10	Sisse logitud kasutajana	soovin ma jälgida teiste veebilehti,	et saaksin kiiresti nendele navigeerida.

2.2.4 Skoop

Kuna lõputöös loodavat rakendust on võimalik lõpmatult arendada, lisades sellele erinevaid funktsionaalsusi, siis on pandud paika rakenduse skoop ehk milline osa rakendusest tehakse valmis antud töö raames.

Lõputöö eesmärgiks on luua minimaalne töötav toode ehk MVP. Arvestades ajalist piirangut, tuleb luua võimalikult kompaktne rakendus, mis lubab kasutajal luua oma veebileht piltide jagamiseks Google Drive'i põhjal.

Kokkuvõttes jääb skoopi tabelites 1 ja 2 kirjeldatud epikutest ja kasutajalugudes kõik epikud ja kõrge prioriteediga kasutajalood.

2.2.5 Rakenduse laiendatavus

MVP-st jäävad välja mitmed funktsionaalsused, mida saab rakendusele tulevikus lisada. Rakendusele saab lisada erinevate pilveteenuste toed, näiteks nagu OneDrive'i, Dropboxi ja Amazon Drive'i.

Tulevikus saab ka rakendusele lisada teiste profiilide jälgimise, kus on võimalus kasutajal jälgida omale meelepäraseid inimeste profiile ning nendele ka kiiresti navigeerida.

Piltide kuvamise poole pealt saab lisada sorteerimise võimalused – pildi lisamise kuupäeva järgi, kaustade kaupa, kus rakendus tuvastab lisatud kaustad ja kuvab neid pilte ka veebilehel gruppides.

2.3 Kasutusvõimalused

Loodaval rakendusel võib olla mitmeid erinevaid kasutusvõimalusi erinevateks olukordadeks. Seda saavad kasutada nii üksikisikud kui ka organisatsioonid.

Esiteks saab rakendust kasutada lihtsalt isiklike piltide jagamiseks oma tuttavatele. Lehel kuvatakse soovitud pildid ning kasutaja saab saata oma profiili lingi ning teised saavad vaadata kasutaja lisatud pilte.

Teiseks saab rakendus kasutada oma portfooliona. Ilmselt eelkõige kasutaksid rakendus niimoodi inimesed, kes tegelevad mingit moodi kunstiga ning vajavad kodulehte enda piltide jaoks. Fotograafid saavad jagada oma töid potentsiaalsete klientidega, samamoodi saavad ka disainerid. Lisaks saaksid ka inimesed jagada oma hobiga seonduvaid töid.

Kolmandaks saaksid rakendust kasutada organisatsioonid, kes soovivad pilte jagada oma liikmetega. Selleks võivad olla ettevõtted kui ka näiteks tudengiorganisatsioonid, kes sooviksid kiirelt paljusid pilte korraga jagada.

3 Tehnoloogiate analüüs

Enne rakenduse kirjutamist otsustakse, milliseid programmeerimiskeeli kasutatakse, millise arhitektuuriga rakendus luuakse ja milline on andmebaasi skeem.

3.1 Programmeerimiskeeled

On erinevaid võimalusi ja keeli veebirakenduste programmeerimiseks. Kõige populaarsemateks on näiteks JavaScript ja TypeScript, HTML ja CSS, Python ja C# [17]. Kuna loodav rakendus peab omama taga- ja esirakendust, tuleb valida mõlema jaoks sobiv keel.

Loodava rakenduse jaoks on valikus kaks erinevat keelt ja raamistikku, Java ja Spring Boot ning teiseks C# ja .NET. Valik tuleneb autori läbitud õppekavast, milles kasutati mõlemat kombinatsiooni veebirakenduste arendamiseks.

Lõputöös loodava rakenduse jaoks kasutatakse C# ja .NET-i, sest nendega on autoril kõige rohkem veebirakenduste kirjutamise kogemust ja on suurem tõenäosus, et funktsionaalsused jõutakse lisada.

Kuna lõputöös loodaval rakendusel peab olema ka esirakendus, siis selle kirjutamiseks kasutatakse Node.js ja TypeScripti Vue raamistikuga. TypeScripti kasutamine teeb koodist arusaamist kergemaks ja aitab paremini JavaScriptis tihti esinevaid vigu, sealhulgas ka keelelisi eripärasid, vältida, rakendades ranget tüüpimise printsiipi [18]. Näiteks JavaScript ei näita, kui üritatakse ühele muutuja tüübile hiljem määrata teist tüüpi väärtust, mis võib suure rakenduse korral tekitada vea, mida on keeruline leida. Enamus vead selguvad JavaScriptis alles kompileerimisel. TypeScript leiab selle vea koha ülesse ning kuvab selle vea koha juba enne kompileerimist.

3.1.1 Teegid

Rakenduse üheks nõudeks on kasutajate haldamine. Rakenduse arendamiseks kasutatakse C# koos .NET raamistikuga, kuhu integreeritakse ASP.NET Core Identity teek, mis loob võimaluse kasutajatel ennast rakenduses autentida.

ASP.NET Core Identity tuleb kasutajate haldamise põhjaga, kus tehakse enamus kasutajatega seotud taustatööd ära. Sinna alla kuuluvad näiteks kasutajate sisselogimise informatsioon, rollid ja nõudepõhine identiteet [19].

Lisaks peab olema võimalus ka Google'iga sisse logida ning selleks peab kasutama teeki Google.Apis.Auth, et valideerida kasutaja juurepääsu tokenit ehk JWT, mis lubab osapoolte vahel turvaliselt informatsiooni saata ning selle üks kõige tavapärasemaid viise autentida kasutajat [20]. Seda kirjeldatakse täpsemalt Google API peatükis jaotises 3.3.

Rakenduse arenduse käigus võib tulla vajadus ka teiste teekide järele, mis on kirjeldatud jaotises 4.

3.2 Arhitektuur

Lõputöös loodava rakenduse jaoks peab valima arhitektuuri, mille põhjal rakendus üles ehitatakse.

Kuna rakenduse mittefunktsionaalsetes nõuetes on öeldud, et rakendus peab olema laiendatav ja kättesaadav, sobib rakenduse arhitektuuri klassifikatsioonis, hajus arhitektuur. Hajusa arhitektuuri põhimõte on jagada töö mitme erineva teenuse vahel, mis jagavad klientrakendust ja ühist andmebaasi. Hajussüsteem on tuntud oma laiendatavuse, vea taluvuse ja elastsuse poolest. Lisaks võimaldab see kiiresti leida õiged kohad muudatuste jaoks ja lihtsustab ka rakenduse testimist, sest iga teenus vastutab ühe kindla funktsionaalsuse eest [21].

Rakenduse arhitektuuriks sobivad kihiline ja ruumipõhine (*space-based*) arhitektuur. Kihiline arhitektuur tähendab seda, et kogu rakendus on jagatud loogilisteks osadeks, millest

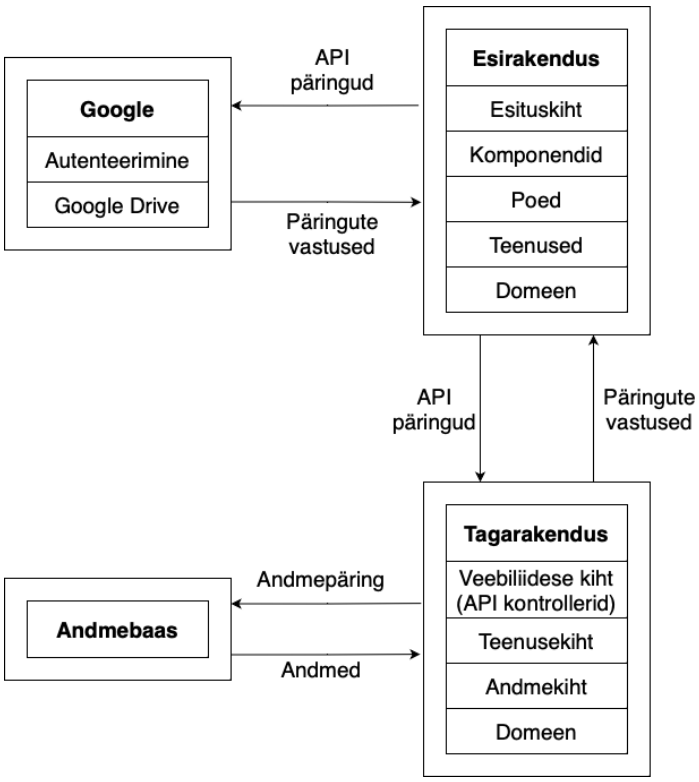
igaüks vastutab kindla funktsionaalsuse eest. Erinevad osad on näiteks veebikiht, teenusekiht, andmekiht ja domeen [22].

Ruumipõhist arhitektuuri kasutavad enamus veebipõhised rakendused, sest rakenduse käitumine on kõigil enam vähem sarnane. Põhilise käitumise all mõistetakse seda, et veeb saadab päringu serverile, siis rakenduse serverile ja lõpuks andmebaasi [23].

Käitumine on mõlemal arhitektuuril sarnane, päringud liiguvad veebist andmebaasi ja vastused sealt tagasi kliendile, kuid peamine erinevus on see, et ruumipõhine arhitektuur kasutab jagatud mäluga paralleel protsessori kontseptsiooni, kus andmebaas eemaldatakse süsteemi transaktsioonilistest protsessidest, mis muudab rakenduse väga skaleeritavaks kasutajate arvu suhtes [24].

Lõputöös loodava rakenduse jaoks sobib kõige paremini kihiline arhitektuur, sest rakendus ei vaja nii palju kiirust ja skaleeritavust kasutajate arvu suhtes, kuna rakendus kasutab Google API-sid, et laadida suurem osa rakenduse mahust tehes päringuid vastu Google servereid. Google serverite päringu kiiruse töötlemise ja vastuse saatmise suhtes ei saa rakendust drastiliselt kiiremaks muuta, kuna kõik sõltub nende jõudlusest. Lisaks kui tekib soov rakendust edasi arendada ja laiendada ka kasutust teiste pilveteenustele, siis jääb andmebaasi mudel samasuguseks, lisanduvad ainult päringud nende pilveteenuste vastu.

Rakenduse arhitektuuri üldistatud skeem on kujutatud joonisel 1.



Joonis 1. Rakenduse arhitektuuri skeem.

3.3 Google API

Lõputöös loodava rakenduse nõuetes on öeldud, et rakenduses peab kasutama Google Drive'i. Google Drive'is saab teatud toiminguid teha, kui kasutaja on sisse logitud ehk rakenduses peab saama autentida Google'i kontoga ja saama ka vajalikku informatsiooni Google Drive'ist. Mistõttu tuleb lähemalt analüüsida Google API võimalusi.

Esmalt tuleb luua projekt Google Cloudis. Projekt on vaja luua, sest siis on võimalik kasutada Google API-t ja lisaks see autoriseerib loodava rakenduse Google'is. Seal määratakse ka rakenduse skoobid ehk milliseid toiminguid peab kasutaja lubama, et rakendust kasutada.

3.3.1 Google OAuth API

Google'i dokumentatsioon pakub erinevaid võimalusi autentimiseks. Loodava rakenduse jaoks kasutatakse *server-side web apps* API-t, kuna rakendus omab ka tagarakendust, mis peab olema võimeline seostama Google'i kontot tagarakenduses loodava kasutajaga.

Autentimiseks saadetakse päring Apigee serverisse koos vajalike andmetega, mis tagastab ligipääsu tokeni, mida kasutatakse kasutaja andmete töötlemiseks [25]. Ligipääsu tokeni jaoks peab olema kasutajal autoriseerimiskood, mis tuleb lisada juurdepääsu tokeni küsimise päringu parameetritesse koos kasutaja informatsiooniga [26].

3.3.2 Google Drive API

Google Drive API võimaldab API vahendusel teha erinevaid toiminguid Drive'is, mida saab teha ka käsitsi läbi Google Drive veebirakenduse.

Google Drive API omab erinevaid skoope, milleks on tavalised vaata, lisa, uuenda ja kustuta faile, failide metaandmete toimingud, failide allalaadimine ja paljud teised [27]. Rakendus kasutab failide nägemise, loomise, kustutamise skoopi. Seda on vaja selleks, et rakendus saaks luua spetsiaalse kausta piltide hoiustamiseks, teisi toiminguid rakendus ei kasuta.

3.3.3 Google API integreerimine

Loodava rakenduse nõuetes on öeldud, et rakendus peab võimaldama luua kasutajaid ja hoidma nende andmeid, millised kasutajad on juba loodud, mis andmetega ning millised on uued kasutajad.

Google API-ga saadava juurdepääsu tokeni autentimiseks, tuleb kasutada teeki Google.Apis.Auth. Antud teek lubab tagarakendusel valideerida saadud Google'i ligipääsu tokeneid. Kui token on valiidne luuakse uus ASP.NET Identity Core kasutaja või logitakse sisse olemasolev.

3.4 Andmebaasi mudel

Lõputöös loodava rakenduse üheks nõudeks on kasutajate andmete kuvamine, mis tähendab, et rakendus vajab kohta, kus andmeid süstemaatiliselt hoida. Et luua võimalikult kompaktne, kuid laiendatav põhi, tuleb läbi mõelda, milliseid andmeid rakenduses hoida tuleb.

Kuna rakendus kasutab ASP.NET Core Identity teeki annab see baasi kasutajate hoidmiseks, sinna tuleb integreerida rakenduses vajaminevad andmed.

Võttes arvesse erinevate teiste rakenduste kasutajate profiile, lisaks mainitud rakenduse spetsiifilisi, peab rakendusse salvestama:

- kasutajanime;
- kasutaja lühitutvustuse;
- lehekülje pealkirja;
- kontaktandmed, sealhulgas sotsiaalmeedia lingid ja personaalsed kontaktid;
- Google Drive'i kausta ID.

Need on põhilised andmed, mida kuvatakse kasutaja kohta ja võimaldavad kasutajal enda veebilehte ja tööd tutvustada. Unikaalne kasutajanimi võimaldab identifitseerida kasutajaid. Lehekülje pealkiri lisatakse selleks, et kui tekib juhus, et soovitud kasutajanimi on võetud ja ei taheta kuvada kasutajanime, näiteks sunnitud kasutajanimi on MinuPildid123, on võimalus lisada lehekülje pealkiri, mis ei pea olema unikaalne ning siis kuvatakse kasutajanime asemel hoopis seda. Google Drive'i kausta ID on oluline, et rakendus teaks, millise kausta piltide, mida tahab kasutaja veebilehele laadida, peab ta kasutama.

Joonisel 2 on kirjeldatud rakenduse andmebaasi mudel, millest sinise taustaga on kujutatud ASP.NET Core Identity olemid ning valge taustaga rakenduse spetsiifilised olemid [28].

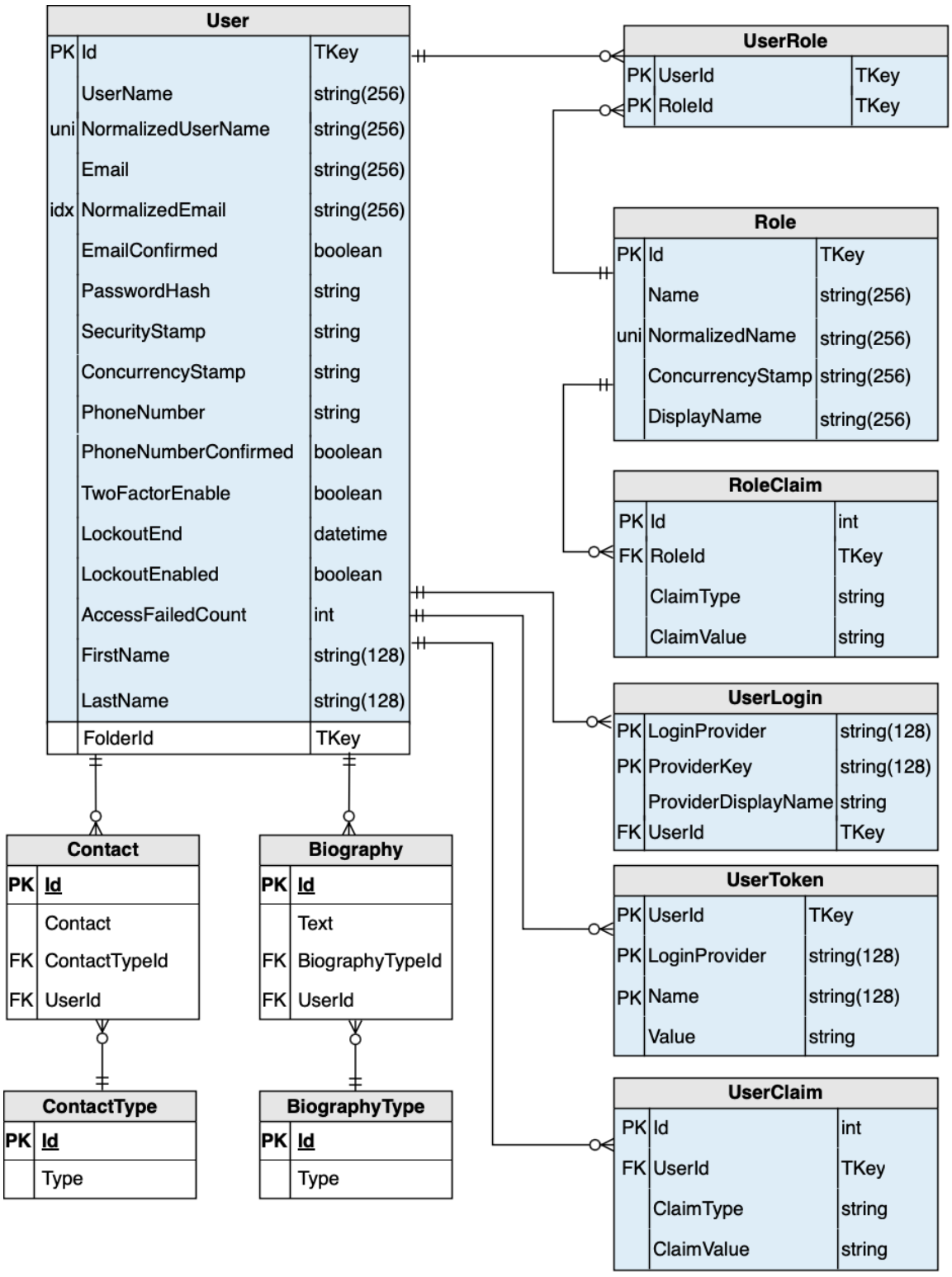
Kasutaja ehk *user* olemile on lisatud väli *FolderId*. See väli näitab, millise kausta on rakendus kasutajale loonud. Selle kasuta ID järgi leitakse Google Drive'ist kaust, mis hoiab pilte, mida veebilehele laetakse. Kõik ülejäänud vajalik informatsioon kasutaja kohta on juba ASP.NET Core Identity väljadega kaetud. Üleliigseid väljasid ignoreeritakse.

FolderId on lisatud just kasutajale, sest kasutajal saab hetkel olla ainult üks *FolderId*. Kui rakendus kasutataks erinevaid pilvesid, tuleb luua eraldi tabelid, näiteks *Folder* ja *Cloud*, kus esimesse on salvestatud kausta ID ja teise on lisatud erinevad pilved. Need olemid peavad olema omavahel seoses, tagamaks seda, et iga kausta ID on seotud ühe pilvega. Nii saavad tulevikus kasutajad kasutada erinevaid pilveteenuseid, kust võetakse soovitud pildid veebilehele erinevatest pilvedest, kuid siis tuleb seostada erinevad kontod, näiteks Google'i ja Microsofti kontod, ühe ASP.NET Idenitity Core'i kasutajaga.

Loodud on ka neli lisa tabelit. *Contact* vastutab kasutaja kontaktandmete eest, milleks on kasutaja teised veebilehed ning üldised andmed. Lisaks on *Contact* seoses *ContactType* olemiga, milles määratakse kontakti tüübid. Algselt on lisatud sinna populaarsemad kanalid nagu näiteks Instagram, Facebook ja LinkedIn. Sinna salvestatakse ka kasutaja telefon ning meiliaadress, sest võib juhtuda, et Google'ist saadud andmed võivad olla isiklikud ja soovitakse lisada muud kontaktid, mida avalikult jagada.

Teised kaks loodud tabelit on *Biography* ja *BiographyType*, mis vastutavad kasutaja biograafia eest. Biograafia on rakenduse kontekstis sisend, mida kasutaja tahab enda kohta veebilehel jagada. Sinna salvestatakse lühitutvustus ja lehekülje pealkiri, mis on *BiographyType* tabelis eristatud erinevate tüüpidega.

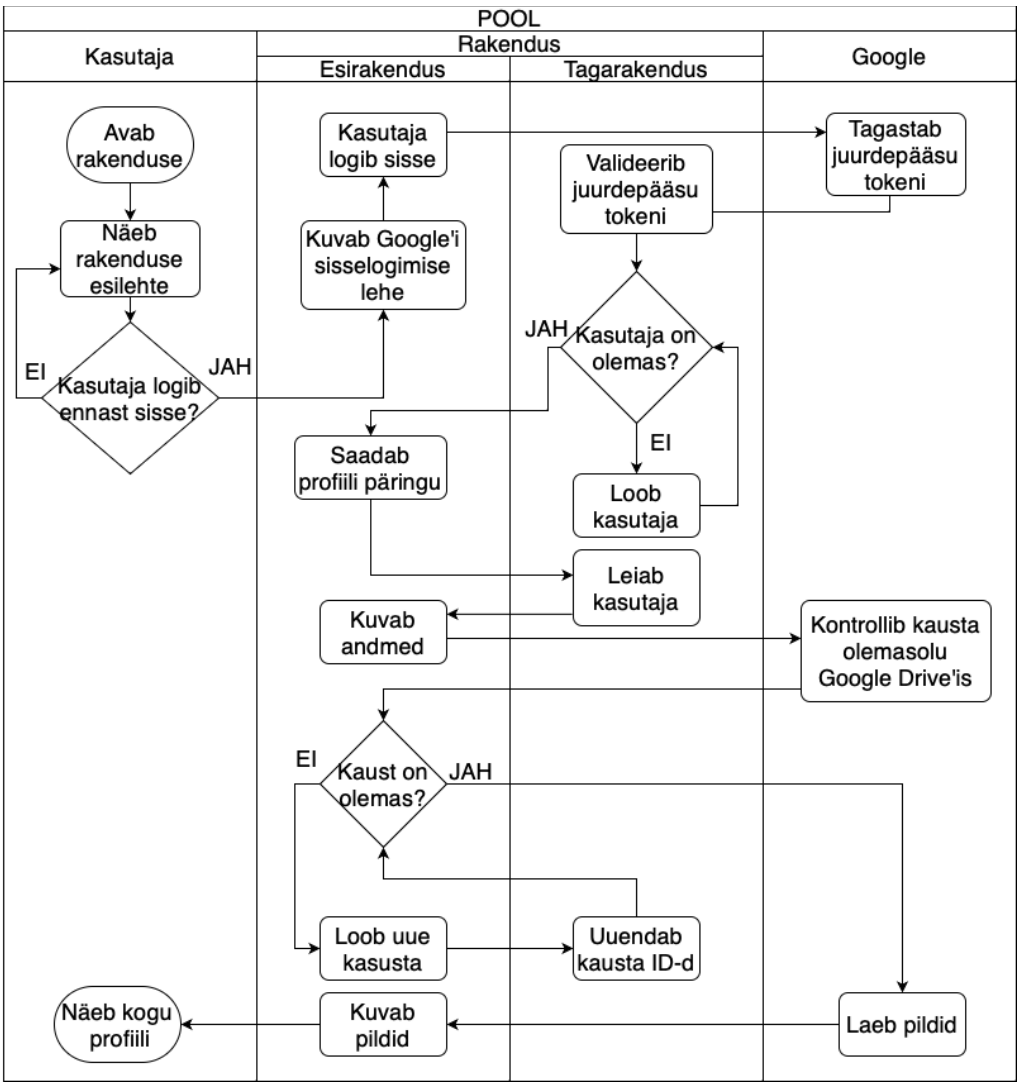
Nii *Contact* kui ka *Biography* on seotud kasutajaga nii, et kasutajal võib olla null kuni mitu erinevat kirjet ehk kasutaja ei pea soovi korral neid lisama, kuid selleks on loodud võimalus. Võimalus lisada kontakte või biograafiat lubab kasutajal oma veebilehte tutvustada ning teenuse pakkumise korral on teistel kasutajatel võimalus kiiresti lehelt leida kontaktid, mille kaudu saab veebilehe omanikuga ühendust.



Joonis 2. Rakenduse olemi-suhte diagramm.

4 Rakenduse loomine

Rakenduse töövoog jaguneb nelja osa vahel, milleks on kasutaja, rakendus, sealhulgas on eristatud esi- ja tagarakendus, ning Google. Sisse loginud kasutaja töövoog omab mitmeid navigatsioone esirakenduse, tagarakenduse ja Google'i vahel, mis jäävad kasutajale nähtamatuks. Kirjeldatud töövoo kohta on koostatud protsessidiagramm, mis on toodud välja joonisel 3.



Joonis 3. Kasutaja sisselogimise töövoog.

4.1 Tagarakendus

Tagarakendus on arendatud vastavalt kihilisele arhitektuurile, mis koosneb domeenist, andmekihist, äriloogika kihist ja avalikust kihist. Domeenis on kirjeldatud vastavalt andmebaasi skeemile (joonis 2) olemid, mille järgi ehitatakse ka andmebaas. Andmekiht vastutab andmebaasi loomisele ja vastavatele tegevustele andmebaasiga, näiteks andmete küsimine ja salvestamine, ning koosneb repositooriumitest.

Andmebaas ja migratsioonid genereeritakse vastavalt *AppDbContext*ile, kus määratake, millised olemid andmebaasi tabeliteks genereeritakse. Arendamiseks kasutatakse lokaalset SQLite andmebaasi, mis genereeritakse joonisel 4 kasutatud käsklustega.

```
dotnet ef migrations add --project App.DAL.EF --startup-project
Uphoto Initial
dotnet ef database update --project App.DAL.EF --startup-project
Uphoto
```

Joonis 4. Andmebaasi genereerimise käsklused.

Äriloogika koosneb teenustest. See kiht töötleb andmebaasi saadetavaid andmeid ja sealt tulenevaid andmeid. Avalik kiht omab versiooni domeenist, mida antakse edasi API-le ainult vajaliku informatsiooniga, see tagab, et API ei avalikustaks informatsiooni, mida ei ole vaja klientrakendusel teada.

Erinevate kihtide omavaheliseks sidumiseks, peab ühest klassist teise andmed kaardistama. Selle jaoks on kasutusel .NETi *AutoMapper*. Igale kihile on tehtud vastav kaardistaja konfiguratsiooni klass, milles konverteeritakse vastavad andmed soovitud andmeteks ja vastupidi. Joonisel 5 olev kood kirjeldab, milline näeb välja konfiguratsiooni klass, kus toimub ümber kaardistamine domeenist andmekihti ja vastupidi.

```
public class AutomapperConfig : Profile
{
    public AutomapperConfig()
    {
CreateMap<Biography, App.Domain.Biography>().ReverseMap();
CreateMap<BiographyType, App.Domain.BiographyType>().ReverseMap();
CreateMap<Contact, App.Domain.Contact>().ReverseMap();
CreateMap<ContactType, App.Domain.ContactType>().ReverseMap();
CreateMap<AppUser, App.Domain.Identity.AppUser>().ReverseMap();
    }
}
```

Joonis 5. *AutomapperConfig* klass domeeni ja andmekihi vaheliseks ümber kaardistamiseks.

Lisaks kihtidele on veel lepingute ehk liideste kiht. Seal olevad klassid kirjeldavad ära, milliseid funktsioone peavad nendest liideseid pärinevad klassid omama. See tagab selle, et rakendus on laiendatav ja testitav.

Rakendus kasutab üldiseid ehk *generic* baase teenuste ja repositooriumite jaoks, et vältida koodi dubleerimist. Näiteks *BaseEntityRepository* klass omab endas funktsionaalsust teha andmebaasi päringuid erinevate tabelite vastu vastavalt sisse antud andmetüübi põhjal ning sellest pärinevad kõik repositooriumi klassid.

4.1.1 ASP.NET Core Identity

Selleks, et lisada ASP.NET Core Identityle vajaminevad väljad, on täiendatud olemasolevat kasutajat. Selleks on loodud *BaseUser* klass, mis pärineb ASP.NET Core Identity *IdentityUser*it ning *AppUser* klass, mis omakorda pärineb *BaseUser*ist. *AppUser* klassile on lisatud vajaminevad väljad, milleks rakenduses on *FolderId.* Nii saab teek aru, et kasutajale tuleb juurde lisada *FolderId* väli.

ASP.NET Core Identity pakub erinevaid võimalusi kasutajate haldamiseks. Selleks võib genereerida identiteedi lehed, kuid antud rakendus peab suhtlema läbi API esirakendusega. See tõttu kasutatakse kaasa tulevaid *SignInManager*i ja *UserManager*i funktsionaalsusi.

Kõik meetodid, mis vajavad kasutaja autentimist, kasutavad JWT-d ehk JSON *Web Token*it, mis lubab osapoolte vahel turvaliselt informatsiooni saata. JWT koosneb päisest, andmetest

ja signatuurist [20]. Selleks, et JWT-d kasutada lisatakse projektile teek Microsoft.AspNetCore.Authentication.JwtBearer, millega genereeritakse kasutaja sisselogimisel talle JWT ning lisatakse päringu päisesse, et ligi pääseda autentimist nõudvatele API lõpp-punktidele, kui see päisest puudub, ei saa kasutaja vastavat lõpp-punkti kasutada.

4.1.2 API lõpp-punktide genereerimine

Taga- ja esirakenduse vaheliseks suhtluseks kasutatakse tagarakenduse API lõpp-punkte. API lõpp-punktide genereerimiseks on kasutatud Microsoft.VisualStudio.Web.CodeGenerators.Mvc teeki, millel on API kontrollerite genereerimise võimalus.

Kontrollerid on genereeritud andmebaasi mudelitest *Contact, ContactType, Biography ja BiographyType*'ile terminali käskluse üldistatud kuju põhjal joonisel 6. Kasutatud käsklus genereerib meetodid vastavalt igale domeeni mudelile, näiteks genereeritud *Contact* olemi API kontrolleri meetodid on järgmised: *GetContacts()*, *GetContact(Guid id)*, *PutContact(Guid id, Contact contact)*, *PostContact(Contact contact)*, *DeleteContact(Guid id), ContactExists(Guid id)*.

```
dotnet aspnet-codegenerator controller -name  {domeeni mudeli
nimetus} Controller -actions -m {domeeni mudeli asukoht} -dc
AppDbContext -outDir ApiControllers -api --useAsyncActions -f
```

Joonis 6. API kontrolleri genereerimise käskluse šabloon.

Genereeritud lõpp-punktide meetodeid on modifitseeritud vastavalt rakenduse vajadustele. Meetodid *GetBiographies()* ja *GetContacts()* on muudetud vastavalt *GetBiographies(string userName)* ja *GetContacts(string userName)*. Nende muudatustega on võimalik leida kasutajanime järgi kõik andmed kasutaja kohta, mis asuvad tabelites *Biography* ja *Contact*. Muutmise, lisamise ja kustutamise meetoditele on lisatud meetodi peale annotatsioon *[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)],* mis lubab

ainult sisse logitud kasutajal neid kasutada. See annotatsioon saab aru, et kasutaja on sisse logitud selle järgi, et kasutajal on päringu päises olemas JWT.

Lisaks genereeritud olemite API kontrolleritele on loodud ka *UserController* ja *GoogleAuthController*. *UserController* vastutab kasutajat nime järgi leidmise eest ning kasutaja andmete uuendamise eest.

GoogleAuthController vastutab kasutaja sisselogimise ja registreerimise eest. API lõpp-punktideks on *Account(GoogleToken googleToken)*, mis valideerib Google'ist saadud ligipääsu tokeni ja kontrollib kasutaja olemasolu, *Login([FromBody] GoogleToken googleToken)* otsib ülesse olemasoleva kasutaja ja logib sisse, *NewLogin([FromBody] TokenAndFolderId tokenAndFolderId)* registreerib uue kasutaja.

Kokku on rakenduses 25 erinevat API lõpp-punkti.

4.1.3 API versioneerimine

Tagarakenduse API on versioneeritud, et tagada rakenduse stabiilsus uuenduste korral ja säilitada vanade versioonide töötamine. Selle jaoks kasutatakse Microsoft.AspNetCore.Mvc.Versioning.ApiExplorer teeki, mis võimaldab määrata versioone nii meetoditele kui ka kontrolleritele.

Igale kontrollerile on määratud tema versioon annotatsiooniga *[ApiVersion("1.0")]*, mille järgi otsustab rakendus päringuid tehes, millist kontrollerit kasutada. Et tuvastada versioon, mida päringu jaoks kasutatakse, on lisatud päringu aadressile versiooni number. Kõik API lõpp-punktide aadressid vastavad järgmisele mudelile *api/v{api versioon}/[kontroller].*

4.2 Esirakendus

Esirakendus on loodud Vue raamistikuga kasutades Node.js, TypeScript programmeerimiskeelt ja Bootstrap 5.

Esirakenduse loomisel alustati tagarakendusega suhtlemise eest vastutavate osade kirjutamisest. Tagarakendusega suhtlemiseks on kasutusel Axios teek, mis on HTTP klient, ning see hoiab endas tagarakenduse baas linki, millele tehakse API päringuid ning sätestab ka iga päringu juurde kuuluvad päised.

Axios teeki kasutatakse JavaScripti *fetch* meetodi asemel, sest Axios töötab kõikide veebilehitsejatega, mis on üks rakenduse mittefunktsionaalsetest nõetest. Axios konverteerib saadud andmed ka kohe JSON-i kujule. Lisaks kaitseb see *Cross-Site Request Forgery* vastu automaatsel [29]. *Cross-Site Request Forgery* on rünnak, kus tehakse päringu võltsimine, mille kaudu saab sundida rakendust tegema ebasoovitud toiminguid, kuhu kasutaja on sisse logitud [30]. Joonisel 7 on toodud rakenduse Axiose HTTP klient, mida kasutatakse läbivalt üle rakenduse, et saata erinevaid päringuid tagarakendusele.

```
import axios from "axios";
export const httpClient = axios.create({
baseURL: "https://localhost:7297/api/v1",
headers: {
"Content-type": "application/json",
},
});
export default httpClient;
```

Joonis 7. Esirakenduse HTTP klient.

Loodud on ka vastavad andmetüübid ehk domeeni klassid, mis kirjeldavad, millises vormis vastused tagarakendusest kuvatakse. Andmetüübid on loodud, et oleks arusaadav, milliseid välju muutuja omab, kui talle on määratud tüüp, mida võimaldab teha TypeScript. Joonisel 8 on toodud kasutaja ehk *User*i andmetüüp.

```
export interface IUserInfo {
  id?: string;
  email: string;
  firstName: string;
  lastName: string;
  userName: string;
  folderId: string;
}
```

Joonis 8. *UserInfo* andmetüüp.

4.2.1 Google API autoriseerimine ja päringud

Esirakenduses on kasutatud Google'i sisselogimise kuvamiseks teeki *vue3-google-signin*, milles on erinevaid viise Google'isse sisenemiseks. Rakenduses on kasutuses *useCodeClient*, et saada juurdepääsu tokeni päringu jaoks vajalik kood. Järgmisena on tehtud päring *https://oauth2.googleapis.com/token* lehel, kus saadakse vastuseks Google juurdepääsu token. Päringusse on lisatud veel *client_id* ja *client_secret*, mis pärinevad Google Cloudist, mis valideerib rakenduse identiteedi. Joonisel 9 on toodud päring juurdepääsu tokeni saamiseks.

```
const token = await axios.post(
  "https://oauth2.googleapis.com/token",
  new URLSearchParams({
    code: code,
    client_id:
      "{client_id}",
    redirect_uri: "http://localhost:5173",
    grant_type: "authorization_code",
    client_secret: "{client_secret}",
  }).toString(),
  {
    headers: {
      "Content-Type": "application/x-www-form-urlencoded",
    },
  }
);
```

Joonis 9. API päring Google'i juurdepääsu tokeni saamiseks.

Juurdepääsu token koosneb paljudest andmetest, kuid rakenduses on kasutatud ainult vajaminevaid väärtusi, milleks on *access_token, refresh_token*, *id_token*, *expires_in* ja *scope*. Saadud vastus saadetakse tagarakendusele autentimiseks ja kasutajaga seostamiseks või registreerimiseks. *Id_token*is on kasutaja andmed, kuhu kuuluvad näiteks email, eesnimi ja perenimi.

Tagarakendus valideerib juurdepääsu tokeni ja saadab esirakendusse JWT-na tagasi kasutaja infomatisooni, milleks on email, eesnimi, perenimi ja kasutajanimi. Kõik peale kasutajanime on saadud *id_token*ist.

4.2.2 Esirakenduse ülesehitus

Esirakendus koosneb domeenist, ruuterist, teenustest, poodidest, komponentidest ja vaadetest. Domeen kirjeldab ära andmetüübid, mida rakenduses kasutatakse. Domeenis on kirjeldatud andmebaasi olemid ning lisaks ka eraldi Google token ja Google Drive'i pildi klass. Pildi klassis on kirjeldatud pildi ID, nimi ja pisipildi link. Domeen on loodud, et TypeScripti abil vältida kirja- ja andmetüüpide vigu.

Ruuteriks ehk vaadete vahel navigeerimise võimaluse loomiseks on kasutatud *vue-router*it, mis lubab luua üheleherakenduse, mis tähendab seda, et rakenduse vaadete vahel navigeerides ei värskendata lehte [31]. Ruuteris on kirjeldatud erinevad teed ja millised vaated nendele teedele vastavalt kuvatakse. Joonisel 10 on toodud üks ruuteri teekond ja sellele määratud vaade, mis on määratud *component*ile.

```
{
   path: "/",
   name: "home",
   component: HomeView,
},
```

Joonis 10. Ruuteri üks teekond ja sellele vastav vaade.

Teenused vastutavad tagarakendusele päringute esitamise eest. Teenused pärinevad baasteenuste klassist, mis kirjeldab ära üldised päringud andmetega. Baaspäringud omavad endas järgmisi päringuid: *put*, *add*, *delete*, *get, getAll, getAllByUserName*. Esimesed viis

päringut omavad päises JWT-d, millega on sisse logitud kasutajal võimalik pääseda nendele API lõpp-punktidele ligi, sest tagarakenduses on märgitud need kasutatavaks ainult sisse logitud kasutajatele. Joonisel 11 on toodud ka näide, milline näeb välja päring koos JWT-ga. *GetAllByUserName* leiab kasutaja informatsiooni ilma JWT-i olemasoluta, sest profiilil peab kuvama ka mitte sisse logitud kasutajatele. Igale teenusele on võimalik juurde kirjutada ka spetsiifilisi päringuid, mis ei rahulda baaspäringutes kirjeldatud päringud.

```
async get(id: string): Promise<TEntity> {
  const response = await httpClient.get(`/${this.path}/${id}`, {
    headers: {
    Authorization: "Bearer " +
    this.identityStore.$state.jwt?.token,
    },
  });

  const res = response.data as TEntity;
  return res;
}
```

Joonis 11. *Get* päringu näide koos JWT-ga.

Lisaks baasteenustele, mis suhtlevad andmebaasiga, on loodud ka *FolderService* ja *IdentityService*. *FolderService* vastutab Google Drive'i API päringute eest. Selle vastutusalasse kuuluvad Google Drive'i kausta olemasolu kontrollimine, uue kausta loomine ning kaustast piltide lugemine.

Google Drive API päringute põhi link on *https://www.googleapis.com/drive/v3/files/*, millele on lisatud veel kausta ID. Kausta loomiseks on kasutatud Google'ilt saadud juurdepääsu tokenit ning sinna lisatakse ka kausta õigused. Loomise päringule lisatakse päisesse veel kausta nimi ja *mimeType*, mis on Google Drive'i faili tüüp. Kausta tüübiks on *application/vnd.google-apps.folder*.

Õiguste lisamiseks tehakse uus päring, kus põhi lingile on lisatud *{folderId}/premissions* ning päisesse roll ja tüüp. Rakendus kasutab kausta loomisel rolliks *reader* ja tüüpiks *anyone*, sest muidu ei ole võimalik pilte kaustast küsida, kui vaatajal ei ole kasuta sisu lugemise luba.

Kuna pildid võivad olla väga suure mahuga ja kõrge resolutsiooniga, on piltide laadimiseks kasutatud piltide pisipilte, et suurema kvaliteediga ja paljude piltide korral suudaks rakendus laadida pildid piisavalt kiiresti. Piltidele vajutades laetakse alles täismahus pilt. Selle jaoks on täpsustatud piltide küsimise päringus, et võetakse ainult pisipilt. Pisipildi suurust saab ka muuta lisades päringule soovitud pildi suuruse pikslites.

IdentityService vastutab Google'ilt juurdepääsu tokeni küsimise ja kasutaja informatsiooni küsimise ning selle uuendamise eest.

Poed on loodud, et hoida meeles rakenduse mingit olekut. Selleks on loodud poe klassid, mis kasutavad Pinia teeki. Pinia on olekuhaldusraamistik, mis lubab hoida rakenduse ühe elutsükli ehk nii kaua kui lehekülge ei värskendata, meeles sinna salvestatud andmeid. Pinia lubab meeles hoitud andmeid jagada komponentide ja vaadete vahel, mis vähendab omakorda päringuid tagarakendusele ja Google serveritele [32].

Sisse logitud kasutaja JWT ja Google tokenit on vaja meeles hoida, et päringutes neid kasutada ning selle eest vastutab *IdentityStore*. Loodud on ka *PictureStore*, et vähendada päringuid, mis küsivad Google Drive'ilt pilte. Sama eesmärgiga on loodud ka *Biography*, *Contact* ja nende tüüpide poed, mis küsivad andmeid rakenduse andmebaasilt.

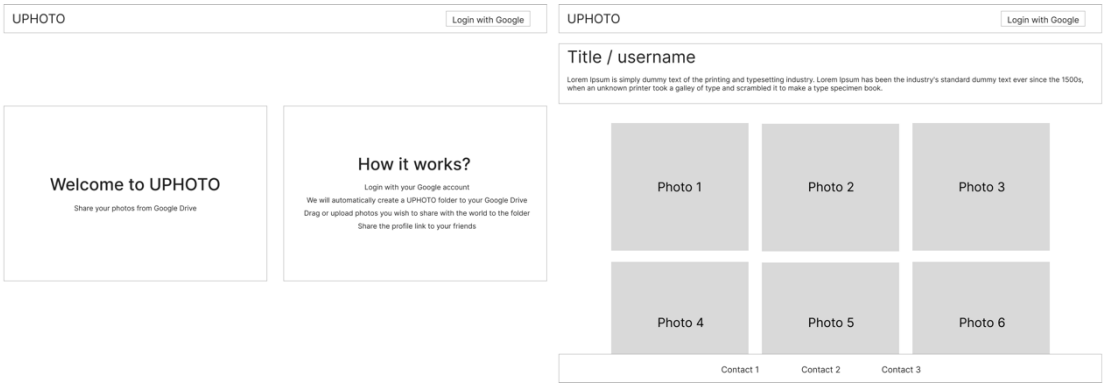
Komponente kasutatakse ühe kindla kujunduse osa ja loogika eristamiseks, et vältida pikkasid HTML lõike vaadetel. Lisaks on ka komponentideks tehtud igal vaatel kuvatavad osad, milleks on lehekülje päis ja jalus ning ka sisselogimise nupp juhul, kui kasutaja ei ole ennast veel sisse loginud. Täpsemad komponendid selguvad esirakenduse kujunduse kirjutamise käigus.

Vaated on mõeldud eelkõige rakenduses navigeerimise jaoks. Vaated omavad komponente ja ka HTML-i, mis on kirjeldab ühe vaate spetsiifilist väljanägemist ja ei pea olema mitmes kohas kasutatavad nagu komponendid. Vaateid on kaks, millest üks on profiili vaade ehk vaade, kus kuvatakse kasutaja informatsiooni ja ka pilte. Teine on kodu vaade, mis on veebilehe koduleht ning tutvustab veebilehte.

4.2.3 Esirakenduse kujundus

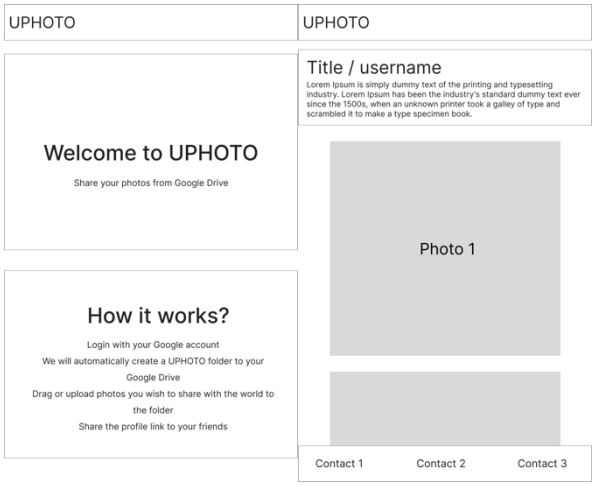
Et visualiseerida erirakenduse kujundust on disainitud Figma rakenduses algsed vaadete prototüübid, mille põhjal on loodud rakenduse kujundus. Kujunduses põhjaks on Bootstrap 5 ning vastavalt vajadusele on lisatud puuduvad kujunduse klassid.

Kujunduse eesmärgiks on luua võimalikult arusaadav ja lihtne lahendus, mis oleks üheselt mõistetav. Selle jaoks on loodud võimalikult minimaalne disain, mis on toodud joonisel 12.



Joonis 12. Vaadete paigutus.

Kuna üheks rakenduse mittefunktsionaalseks nõudeks on see, et veebilehte peab olema võimalik mugavalt vaadata ka nutiseadmest, siis on loodud ka prototüüp väiksema ekraaniga mõeldud seadmetele. Prototüüp on toodud joonisel 13.



Joonis 13. Telefoni vaadete paigutus.

4.2.4 Komponendid ja vaated

Rakendusse on loodud prototüübi alusel kaks vaadet, milleks on avaleht ja profiil ning ka vaade kasutaja informatsiooni muutmise jaoks. Komponendid on loodud vastavalt sellele, kas seda koodiosa tuleks kuvada mitmel lehel ja kas see omab enda sees mingisugust suuremat loogikat, mis muudaks vaate keerulisemaks ja pikemaks ilma seda eraldamata [33]. Esirakenduse jaoks on loodud järgmised komponendid: *ContactFooter*, *Header*, *ProfileHeader*, *SignInButton*, *UnAuthorized*.

ContactFooter on kasutuses ainult profiili vaates ning kuvab kasutaja kontaktide ikoone, sellele jaoks on tehtud eraldi komponent, sest see omab enda sees loogikat, mis küsib tagarakenduse käest kasutaja andmeid. *Header*is on lehekülje päis, kus on lehekülje nimi ja *SignInButton*i komponent, mis kuvab sisselogimis nuppu, hiljem saab sinna lisada ka otsingu nupu, millega saab teisi kasutajaid otsida. *ProfileHeader* koosneb kasutaja profiili andmetest, milleks on kasutajanimi või lehekülje pealkiri, biograafia, piltide uuesti laadimise nupp ja sisse loginud kasutaja puhul ka nupp kasutaja informatsiooni muutmiseks. *UnAuthorized* komponenti kasutatakse, kui mitte sisse loginud kasutaja üritab teha toimingut, mida saaks ainult sisse loginud kasutaja teha, näiteks üritab sisse logimata kasutaja muuta oma andmeid, sest link on kiirsalvestatud veebibrauseris.

Selleks, et kuvada õigeid vaateid kasutajale vastavalt aadressile, kus ta hetkel viibib on *App.vue* klassis määratud *<RouterView />* element. *App.vue* klassi põhjal ehitatakse ka veebirakendus. *RouterView* komponent on Vue Router teegi võimalus kuvada rakenduse ruuteris määratud teekonnale vastavat vaadet või komponenti. Joonisel 14 on näidatud, milline on *App.vue* klassi HTML-i osa, mis kujutab endast, kuidas kasutatakse *Header* komponenti ning *RouterView*d.

```
<template>
  <div className="container">
    <Header />
    <main role="main" className="pb-3">
      <RouterView />
    </main>
  </div>
</template>
```

Joonis 14. *App.vue* HTML-i osa.

Kõik vaated ja komponendid koosnevad *template* elemendist ning ka *script* elemendist. *Template* elemendi sees on kogu veebilehe skelett ehk HTML. *Script* element koosneb imporditud teekidest ja komponentidest ning funktsioonidest, mida *template* vajab. Lisaks on ka seal ka muutujad, mida on vaja lehekülje laadimisel lehel kuvada.

Kujunduses on kasutatud Bootstrap 5 teeki, et säästa aega HTML elementidele käsitsi stiili kirjutamisel ning kasutada ka seal pakutavaid ikoone.

5 Tulemused

Selleks, et analüüsida saad tulemust ja kas loodud rakendus täidab oma eesmärki võrreldakse jaotises 2.2.1 välja toodud nõudeid loodud rakenduse funktsionaalsustega tabelis 3, kus on välja toodud nõue, kas see on täidetud ja täpsustus.

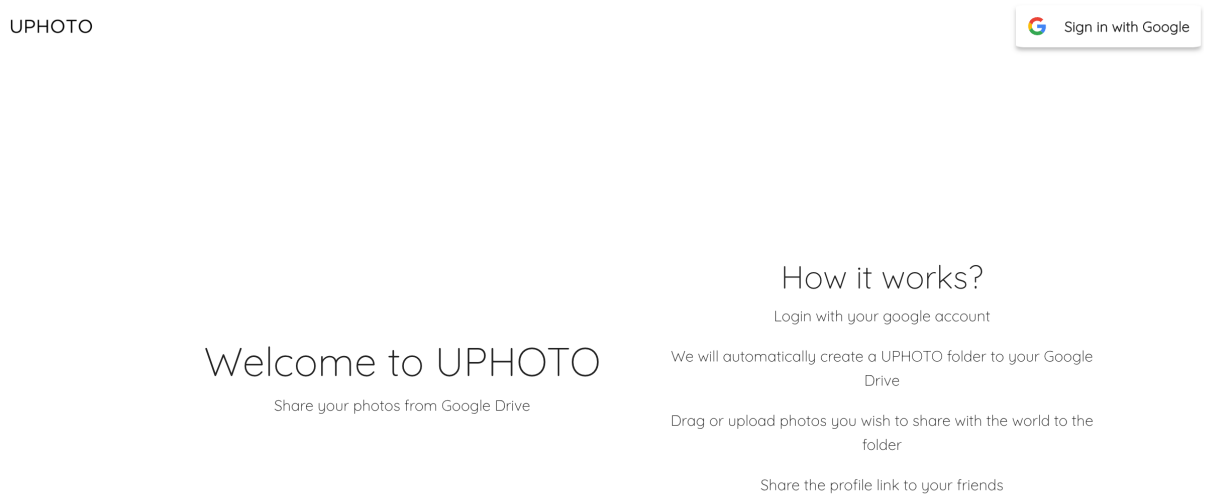
Tabel 3. Nõuete ja rakenduse funktsionaalsuse võrdlus.

Nõue	**Täidetud**	**Täpsustus**
Rakendus pakub võimalust kuvada pilte võimalikult hea kvaliteediga.	Jah	Rakenduse kiiruse hoidmiseks laetakse profiile alguses pisipildid ning taustal laetakse täis suuruses pildid. Pisipildile vajutades avatakse pilt täis suuruses, kuid määratud on maksimaalne kõrgus, et pilt ekraanile mahuks. Muid optimeerimise toiminguid pildiga ei tehta.
Rakenduse põhifunktsionaalsuseks on piltide jagamine.	Jah	
Rakendus peab omama võimalust luua kasutaja.	Jah	Kasutajate loomiseks on kasutatud ASP.NET Core Identity ning vue3-google-signin teeke. Viimane neist on esirakenduse teek, mis võimaldab Google'i kasutajaga sisse logida ning see seostatakse tagarakenduses *Core Identity* kasutajaga.
Rakendus peab meeles hoidma kasutaja andmeid.	Jah	Rakenduse andmebaas on võimeline kasutaja informatsiooni salvestama, seda saab kuvada ning kasutaja saab oma informatsiooni muuta.

Nõue	Täidetud	Täpsustus
Rakendus peab kasutama kasutaja Google Drive'i piltide hoidmiseks.	Jah	Kasutaja registreerudes luuakse talle tema Google Drive'i spetsiaalne kaust, mis on ilma sisselogimiseta avalik lugeda kõigile. Loodud kasuta ID salvestatakse andmebaasi. Kasutades salvestatud kausta ID-d loeb rakendus pildid, mida kasutaja tahab kuvada. Pilte andmebaasi ega kuhugi mujale rakendus ei salvesta. Esirakendus hoiab ainult ühe elutsükli ajal meeles linke piltidele, et vältida korduvat küsimist Google Drive'ist lehel navigeerides.
Rakenduse laiendatavus erinevate pilve teenustega.	Ei	Jäi lõputöö skoobist välja, kuid ASP.NET Identity Core teek lubab logida sisse mitmete teiste kolmandate osapooltega, mida saab kasutada erinevate pilvedega sidumiseks.
Rakendust peab olema võimalik kasutada igas tänapäevases veebibrauseris.	Jah	Esirakendus kasutab Axios teeki, mis võimaldab kasutada kõiki veebibrausereid.
Veebileht peab kuvama sisu õiges formaadis, kas arvutis või nutitelefonis.	Jah	Veebirakenduse vaated on tehtud responsiivseks. Need skaleeruvad vastavalt kasutaja seadmele ja veebiakna suurusele.

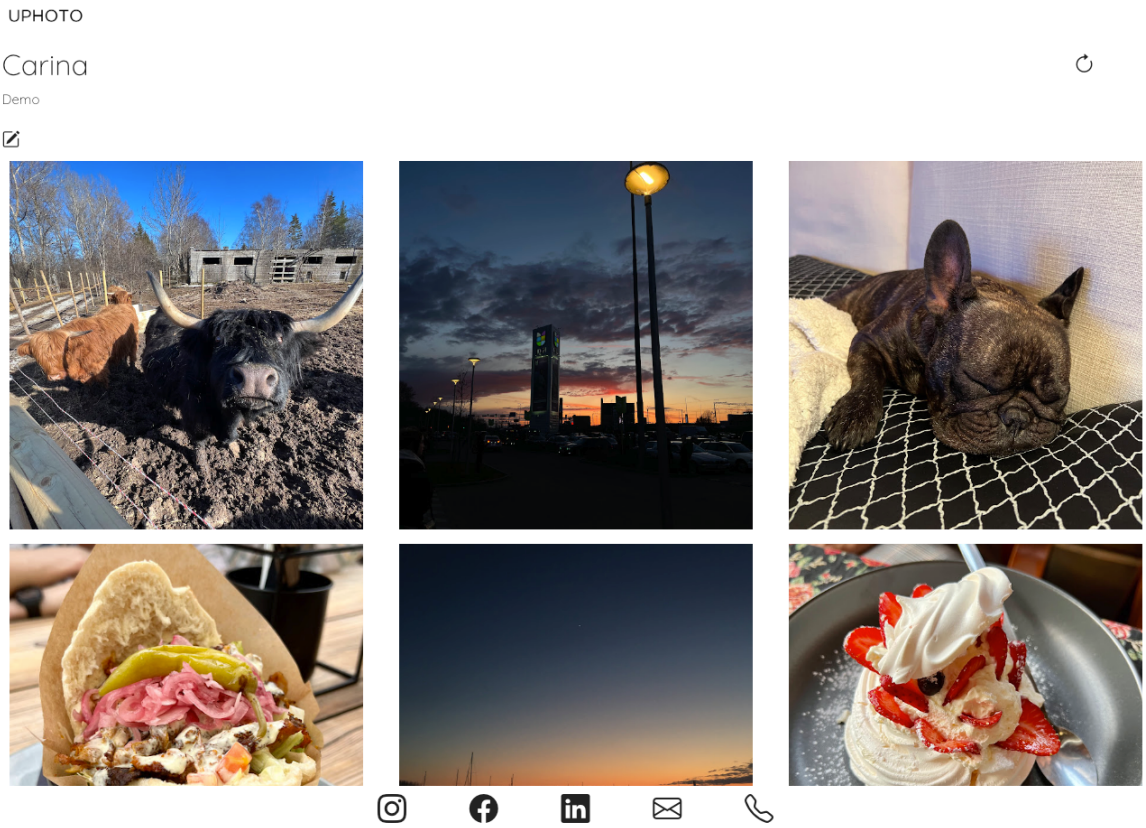
Rakenduse kujundus loodi joonistel 12 ja 13 näidatud prototüüpide alusel. Joonisel 15 kujutatud avaleht tutvustab kasutajale rakendust ning kuidas see töötab. Üleval paremas nurgas asub Google'i sisselogimise nupp, mis avab Google'i hüpikakna, kus on toodud kasutaja veebibrauserisse lisatud kasutajad. Pärast kasutaja valmist kuvatakse kasutajale nõusoleku leht, kus on välja toodud informatsioon või funktsionaalsused, mida rakendus

sooviks kasutada. Rakenduse töötamiseks vajaminevad funktsionaalsused, mille nõusolekut kasutajalt küsitakse, on kirjeldatud jaotises 3.3.2.



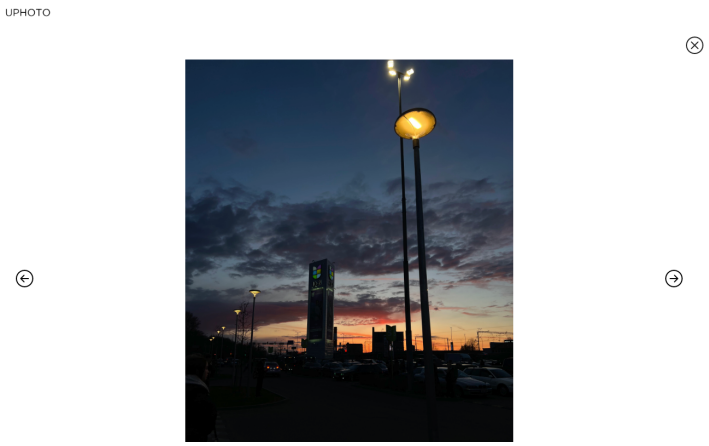
Joonis 15. Avaleht.

Joonisel 16 on profiili vaade sisse logitud kasutaja korral. Kasutaja informatsiooni päises on toodud kasutajanimi, lehekülje pealkirja olemasolul pealkiri, ning lühitutvustus. Selle all olev pliiatsi ikoon viitab kasutaja informatsiooni muutmise lehele ning paremal pool asuv nupp on piltide uuesti laadimiseks, et vältida terve lehe uuesti laadimist, kui Google Drive'i on lisatud pilte või on sealt kustutatud. Lehekülje jaluses asuvad kontakti ikoonid, mis viitavad kasutaja sisestatud lehekülgedele või muudele kontaktidele. Joonisel on näha kõiki kontakti ikoone, mida rakendus hetkel võimaldab sisestada. Hiljem on võimalik rakendusse lisada ka teisi kontakte, kuid hetkel on lisatud põhilised. Ikoonid ilmuvad ainult nende andmete kohta, mille on kasutaja sisestanud. Kõik ikoonid on Bootstrap 5 ikoonide kogust.



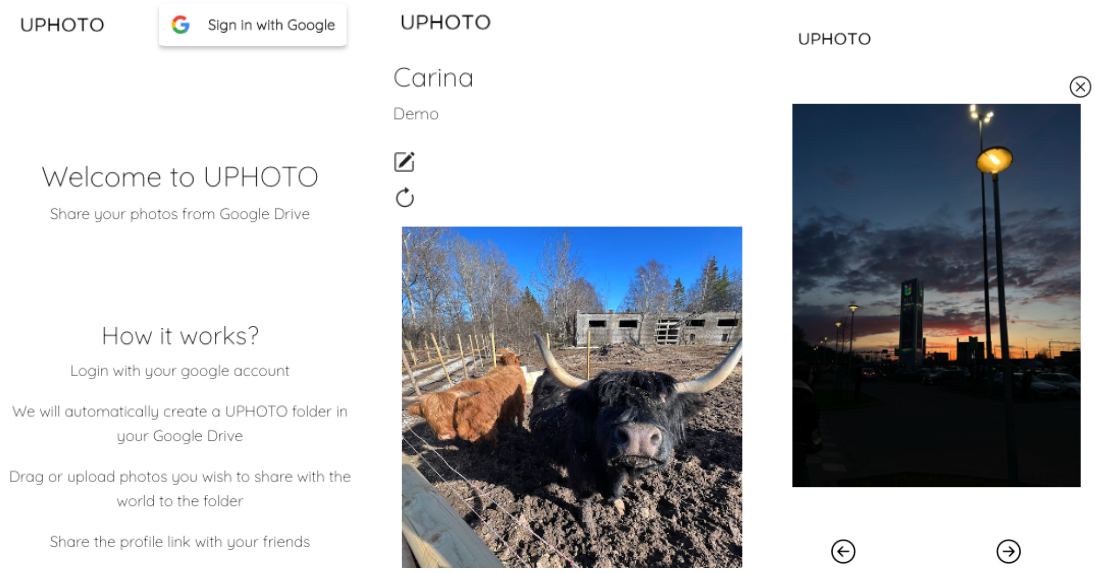
Joonis 16. Profiili vaade.

Täismahus piltide vaade on näidatud joonisel 17. See vaade kuvatakse, kui kasutaja vajutab profiilil olevale pildile. Mugavuseks on lisatud navigeerimise nupud, mis viitavad eelmisele või järgmisele pildile.



Joonis 17. Avatud pildi vaade.

Telefoni vaadete loomiseks on kasutatud piltide ruutude jaoks Bootstrap 5 *row* klassi, mis võimaldab muuta lehekülje responsiivseks ekraani või veebiakna suhtes kasutades *grid* ehk ruudustiku moodulit. *Row* klass omab erinevaid murdekohti, kus lehekülje paigutus muutub [34]. Telefoni vaated on toodud joonisel 18.



Joonis 18. Telefoni vaated.

6 Kokkuvõte

Lõputöö eesmärgiks on luua kasutajasõbralik piltide jagamise veebirakenduse minimaalne töötav toode, mis kasutab Google Drive'i piltide hoiustamiseks.

Lõputöö raames loodi veebirakendus, mis võimaldab kasutajal kiiresti oma pildid avalikustada nii, et nende kvaliteet säiliks, ei oleks lisatööd oma veebilehe majutamisega ja oleks algselt ilma lisakulutusteta. Veebirakendus koosneb andmebaasist, taga- ning esirakendusest, kus tagarakendus on kirjutatud kasutades C# koos .NET raamistikuga ning esirakendus TypeScriptis Vue raamistikuga.

Töös on paika pandud rakenduse nõuded, mis tulenevad olemasolevate lahenduste analüüsist ning nende probleemsematest kohtadest, mis võivad muuta kasutajale piltide jagamise ja oma profiili loomise keerulisemaks. Kõik nõuded suudeti minimaalse töötava toote loomiseks rakendada. Rakendus on laiendatav ning sinna on võimalik lisada veel erinevaid võimalusi piltide jagamiseks ning on võimalik arendada edasi ka laiendatavust teistele pilve teenustele, mis omavad API-t.

Kasutatud kirjandus

[1] I. Blagojević, „Cloud Computing Statistics,“ [Võrgumaterjal]. Available: https://99firms.com/blog/cloud-computing-statistics/#gref. [Kasutatud 27 märts 2023].

[2] M. Broz, „Number of Photos (2023): Statistics, Facts, & Predictions,“ 10 märts 2023. [Võrgumaterjal]. Available: https://photutorial.com/photos-statistics/. [Kasutatud 22 aprill 2023].

[3] L. Moss, „Social Media Sharing: The Psychology of Why We Share,“ 6 detsember 2022. [Võrgumaterjal]. Available: https://everyonesocial.com/blog/the-psychology-of-how-and-why-we-share/. [Kasutatud 22 aprill 2022].

[4] C. Bryan-Smith, „Best Way to Store Photos in 2023 (6 Photo Storage Tips),“ [Võrgumaterjal]. Available: https://expertphotography.com/best-way-to-store-photos/. [Kasutatud 22 aprill 2023].

[5] Adobe Express, „The eight top social media sites you should prioritize in 2023,“ 8 September 2022. [Võrgumaterjal]. Available: https://www.adobe.com/express/learn/blog/top-social-media-sites. [Kasutatud 27 märts 2023].

[6] M. Walters, „Uploading to Instagram without Losing Image Quality,“ [Võrgumaterjal]. Available: https://mikewalterz.com/uploading-to-instagram-without-losing-image-quality/. [Kasutatud 2023 märts 27].

[7] Instagram, „Image resolution of photos you share on Instagram,“ [Võrgumaterjal]. Available: https://help.instagram.com/1631821640426723. [Kasutatud 27 märts 2023].

[8] Meta, „Recommended minimum image pixel requirements across placements,“ [Võrgumaterjal]. Available: https://www.facebook.com/business/help/469767027114079?id=271710926837064. [Kasutatud 27 märts 2023].

[9] W. Morris, „14 Best CMS Platforms to Start a Website in 2023,“ 30 märts 2023. [Võrgumaterjal]. Available: https://www.hostinger.com/tutorials/best-cms. [Kasutatud 28 märts 2023].

[10] WordPress, „Plugins,“ [Võrgumaterjal]. Available: https://wordpress.org/plugins/. [Kasutatud 28 märts 2023].

[11] WordPress, „Themes,“ [Võrgumaterjal]. Available: https://wordpress.org/themes/#. [Kasutatud 28 märts 2023].

[12] Drupal, „Overview of Drupal, The software you need to get is free,“ [Võrgumaterjal]. Available: https://www.drupal.org/docs/understanding-drupal/overview-of-drupal. [Kasutatud 22 aprill 2023].

[13] T. Thibodeau, „WordPress.com vs WordPress.org: What's the Difference?," [Võrgumaterjal]. Available: https://wordpress.com/go/website-building/wordpress-com-vs-wordpress-org/. [Kasutatud 23 aprill 2023].

[14] J. L. Vanessa Vorteil, „Tarkvara analüüs ja testimine. Informaatika valikkursus gümnaasiumile.," Tallinna Ülikool, [Võrgumaterjal]. Available: https://web.htk.tlu.ee/digitaru/testimine/chapter/tarkvara-arendusnouded/. [Kasutatud 29 märts 2023].

[15] S. Paradkar, „Introducing NFRs," %1 *Mastering Non-Functional Requirements*, 2017.

[16] A. Spadafora, „Best cloud storage in 2023," 17 aprill 2023. [Võrgumaterjal]. Available: https://www.tomsguide.com/buying-guide/best-cloud-storage. [Kasutatud 22 aprill 2023].

[17] Ø. Forsbak, „Top 8 Programming Languages for Web Development in 2023," 25 november 2022. [Võrgumaterjal]. Available: https://www.orientsoftware.com/blog/programming-languages-for-web-development/. [Kasutatud 2 aprill 2023].

[18] TypeScript, „TypeScript is JavaScript with syntax for types.," [Võrgumaterjal]. Available: https://www.typescriptlang.org/. [Kasutatud 2 aprill 2023].

[19] Microsoft, „Identity model customization in ASP.NET Core," [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/customize-identity-model?view=aspnetcore-7.0. [Kasutatud 4 aprill 2023].

[20] JWT, „Introduction to JSON Web Tokens," [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 13 aprill 2023].

[21] M. Richards, „Architectural Structures and Styles, Architecture Classification, Distributed Architectures," %1 *Software Architecture Patterns, 2nd Edition*, O'Reilly Media, Inc., 2022.

[22] M. Richards, „Layered Architecture," %1 *Software Architecture Patterns, 2nd Edition*, O'Reilly Media, Inc., 2022.

[23] M. Richards, „Space-Based Architecture," %1 *Software Architecture Patterns, 2nd Edition*, O'Reilly Media, Inc., 2022.

[24] M. Richards, „Space-Based Architecture, Topology and Components," %1 *Software Architecture Patterns, 2nd Edition*, O'Reilly Media, Inc..

[25] Google, „Implementing the client credentials grant type," [Võrgumaterjal]. Available: https://cloud.google.com/apigee/docs/api-platform/security/oauth/oauth-20-client-credentials-grant-type. [Kasutatud 4 aprill 2023].

[26] Google, „Using OAuth 2.0 to Access Google APIs," [Võrgumaterjal]. Available: https://developers.google.com/identity/protocols/oauth2. [Kasutatud 4 aprill 2023].

[27] Google, „OAuth 2.0 Scopes for Google APIs,“ [Võrgumaterjal]. Available: https://developers.google.com/identity/protocols/oauth2/scopes#drive. [Kasutatud 16 aprill 2023].

[28] D. Anderson, „Using your own database schema and classes with ASP.NET Core Identity and Entity Framework Core,“ 21 jaanuar 2018. [Võrgumaterjal]. Available: http://danderson.io/posts/using-your-own-database-schema-and-classes-with-asp-net-core-identity-and-entity-framework-core/. [Kasutatud 5 aprill 2023].

[29] P. Yadav, „Fetch vs. Axios for making HTTP requests,“ 29 november 2022. [Võrgumaterjal]. Available: https://www.tothenew.com/blog/fetch-vs-axios-for-making-http-requests/. [Kasutatud 23 aprill 2023].

[30] R. Ranjan, „Cross site request forgery (CSRF) attack,“ 27 august 2022. [Võrgumaterjal]. Available: https://medium.com/@rajeevranjancom/cross-site-request-forgery-csrf-attack-6949edb9e405. [Kasutatud 23 aprill 2023].

[31] Vue Router, „Introduction,“ [Võrgumaterjal]. Available: https://router.vuejs.org/introduction.html. [Kasutatud 23 aprill 2023].

[32] Pinia, „Why should I use Pinia?,“ [Võrgumaterjal]. Available: https://pinia.vuejs.org/introduction.html#why-should-i-use-pinia. [Kasutatud 15 aprill 2023].

[33] Vue.js, „Components Basics,“ [Võrgumaterjal]. Available: https://vuejs.org/guide/essentials/component-basics.html. [Kasutatud 25 aprill 2023].

[34] Bootstrap v5.0, „Grid system,“ [Võrgumaterjal]. Available: https://getbootstrap.com/docs/5.0/layout/grid/. [Kasutatud 17 aprill 2023].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Carina Ruut

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Google Drive'i põhine piltide jagamise veebirakendus", mille juhendaja on German Mumma
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

14.05.2023

[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.