TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Villem Mesila 123560IAPB

Matemaatikaülesannete üleslaadimise ja haldamise süsteem

Bakalaureusetöö

Juhendaja: Gunnar Piho
PhD

Tallinn 2021

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: Villem Mesila

18.05.2021

Annotatsioon

Tööl on kaks põhilist eesmärki. Esiteks luua keskkond, kuhu kasutaja saaks üles laadida faile, millest loetakse välja ja salvestatakse matemaatikaülesanded koos vastuste ja lahendustega, mida oleks võimalik samas keskkonnas hallata. Töö teiseks eesmärgiks on leida võimalus failist matemaatiliste sümbolite lugemiseks ja salvestamiseks nii, et neid oleks võimalik brauseris loetaval kujul kuvada.

Lõputöö tulemusena valmis halduskeskkond, kuhu on võimalik ülesandeid üles laadida Microsoft Wordi failidest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 5 peatükki, 6 joonist, 1 tabelit.

Abstract

System for Uploading and Managing Mathematical Tasks

This work has two main purposes. First, to create an application where user can upload files from which the application would read and save mathematical tasks along with answers and solutions. The application should also provide a user interface for managing uploaded tasks, answers and solutions. Second aim of this work is to find a way to read and store mathematical symbols so that they would remain in human readable form when displayed in browser.

As a result an application was created where user can upload tasks from Microsoft Word files.

The thesis is in Estonian and contains 25 pages of text, 5 chapters, 6 figures, 1 table.

Lühendite ja mõistete sõnastik

MVC	*Model-view-controller,* mudel-vaade-kontroller, tarkvara disainimuster
DTO	*Data transfet object,* objekt andmete ülekandmiseks
ORM	*Object-relational mapper,* tehnoloogia, mis seob omavahel andmebaasitabelid ja programmeerimiskeele klassid
MathMl	*Mathematical Markup Language,* märgistuskeel matemaatiliste sümbolite esitamiseks
REST	*Representational state transfer,* tarkvara arhitektuuri stiil
API	*Application programming interface,* liides arvutiprogrammide omavaheliseks suhtlemiseks
CORS	*Cross-Origin Resource Sharing,* päritoluülene ressursside jagamine
OMML	*Office Math Markup Language,* keel, mida Microsoft Word kasutab matemaatiliste sümbolite kirjutamiseks
HTML	*HyperText Markup Language,* keel, millest brauser loeb välja ja kuvab veebilehe sisu
XML	*Extensible Markup Language,* andmete märgistamise keel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Argo kirjastuse poolt publitseeritava raamatuseeria „Eksaminandile matemaatika riigieksamist“ autor Helgi Uudelepp pöördus sooviga avaldada antud matemaatikaülesannete kogumikud virtuaalsel kujul. Selleks oleks vaja luua rakendus, mis koosneb kahest keskkonnast:

1) halduskeskkonnast, mis võimaldaks kasutajal laadida üles faile, kust loetakse välja ja salvestatakse ülesanded koos vastuste ja lahendustega, mida oleks võimalik samas keskkonnas hallata.

2) harjutuskeskkonnast, mille kaudu saab ülesandeid lahendada.

Käesoleva töö eesmärgiks on halduskeskkonna loomine. Teiseks eesmärgiks on leida võimalus failist matemaatiliste sümbolite lugemiseks ja salvestamiseks nii, et need säiliksid arusaadaval kujul. Esialgu saab lisada ja lahendada ainult valikvastustega ülesandeid, aga hiljem on plaanis anda rakendus edasiarendamiseks G. Piho haridusinformaatika doktorantidele, kelle ülesanne on välja töötada matemaatikaülesannete lahenduste masinanalüüs ja õpilasi juhendav automatiseeritud (poolautomatiseeritud) tagasiside. Seda arvestatakse rakenduse loomisel ja seetõttu üritatakse kirjutada kood, mis oleks võimalikult taaskasutatav ja järgiks puhta koodi põhimõtteid.

2 Kasutatud tehnoloogiad

2.1 ASP.NET Core

ASP.NET Core on Microsofti loodud platvormiülene avatud lähtekoodiga raamistik veebirakenduste loomiseks [1]. Antud rakenduses kasutatakse ASP.NET Core'il põhinevat veebirakenduste loomise platvormi ASP.NET Core MVC. See põhineb MVC arhitektuuril, mis jaotab rakenduse kolmeks komponendiks. Kontroller püüab kinni kasutaja päringud, töötleb mudelis olevaid andmeid ja tagastab vaate. Mudel sisaldab andmeid, mis kirjeldavad rakenduse olekut. Vaade koostab etteantud andmete põhjal veebilehe kasutajale näitamiseks [2]. ASP.NET Core MVC'l on ka sisseehitatud funktsionaalsus REST API'de koostamiseks.

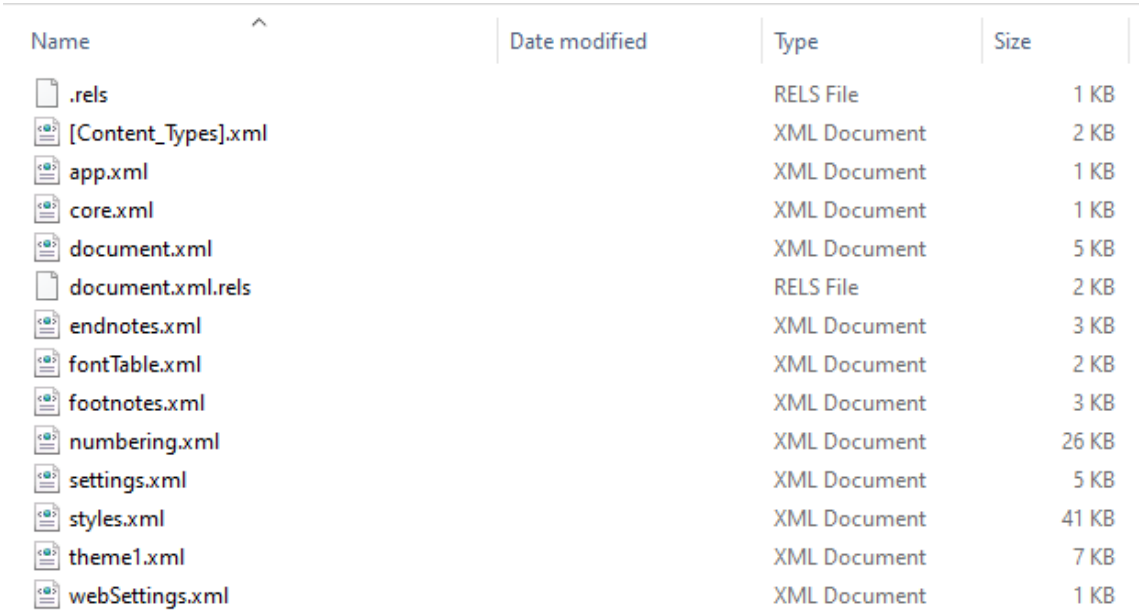
2.2 Entity Framework Core

Entity Framework on avatud lähtekoodiga platvormiülene ORM [3]. ORM on tehnoloogia, mis seab omavahel vastavusse klassid ja andmebaasitabelid. Entity Frameworkil on selle seose loomiseks 3 lähenemist: *Code First*, *Database First* ja *Model First* [4]. Esimese lähenemise korral kirjutatakse kõigepealt valmis klassid, millest genereeritakse andmebaas. Teisel juhul luuakse klassid juba olemasoleva andmebaasi põhjal. Kolmanda lähenemise puhul luuakse disainitööriista kasutades objektid, millest genereeritakse klassid ja andmebaas. Käesoleva projekti loomisel kasutatakse *Code First* lähenemist.

Entity Framework toetab paljusid andmebaasi teenusepakkujaid. See aitab abstraheerida spetsiifilise andmebaasiga seotud detaile ja võimaldab arendajatel teostada andmebaasioperatsioone koodis defineeritud klassidega. Andmebaasipakkuja vahetamiseks tuleb muuta ainult vastav väli konfiguratsioonifailis.

2.3 Microsoft Word

Microsoft Word on tekstitöötlusprogramm. Sellel on 2 võimalikku failinime laiendit: DOC ja DOCX. Alguses kasutati vanemat DOC laiendit, aga 2007. aastast on standardiks Open XML'il põhinev DOCX [5]. Erinevalt DOC'ist, mis on binaarfail, koosneb DOCX hulgast XML ja binaarfailidest, mis on kokkupakitud ZIP failiks. Seetõttu on DOCX formaadis failide sisu võimalik lugeda, ilma et arvutisse oleks installitud Microsoft Word.



Name	Date modified	Type	Size
.rels		RELS File	1 KB
[Content_Types].xml		XML Document	2 KB
app.xml		XML Document	1 KB
core.xml		XML Document	1 KB
document.xml		XML Document	5 KB
document.xml.rels		RELS File	2 KB
endnotes.xml		XML Document	3 KB
fontTable.xml		XML Document	2 KB
footnotes.xml		XML Document	3 KB
numbering.xml		XML Document	26 KB
settings.xml		XML Document	5 KB
styles.xml		XML Document	41 KB
theme1.xml		XML Document	7 KB
webSettings.xml		XML Document	1 KB

Joonis 1 DOCX fail lahtipakitud kujul

2.4 Matemaatika märgistuskeeled

OMML on keel, mida Microsoft Word kasutab matemaatiliste sümbolite märgistamiseks.

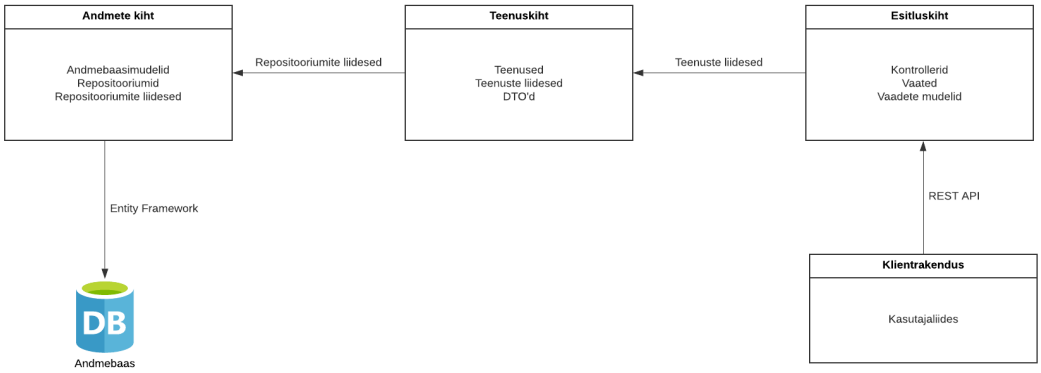
MathMl on märgistuskeel, mis on mõeldud matemaatiliste sümbolite kuvamiseks veebilehtedel [6]. Enamik populaarseid brausereid ei toeta hetkel MathMl'i.

MathJax[1] on avatud lähtekoodiga jacascripti teek, mis aitab tänapäevastel brauseritel korrektselt renderdada erinevates matemaatilistes märgistuskeeltes, sealhulgas ka MathMl'is defineeritud matemaatilisi sümboleid [7].

[1] https://www.mathjax.org/

3 Rakendus

3.1 Arhitektuur



Joonis 2 Kolmekihilise arhitektuuri kasutamine

Rakenduse loomisel kasutati kolmekihilist arhitektuuri, mis koosneb andmete kihist, teenuste kihist ja esitluskihist [8].

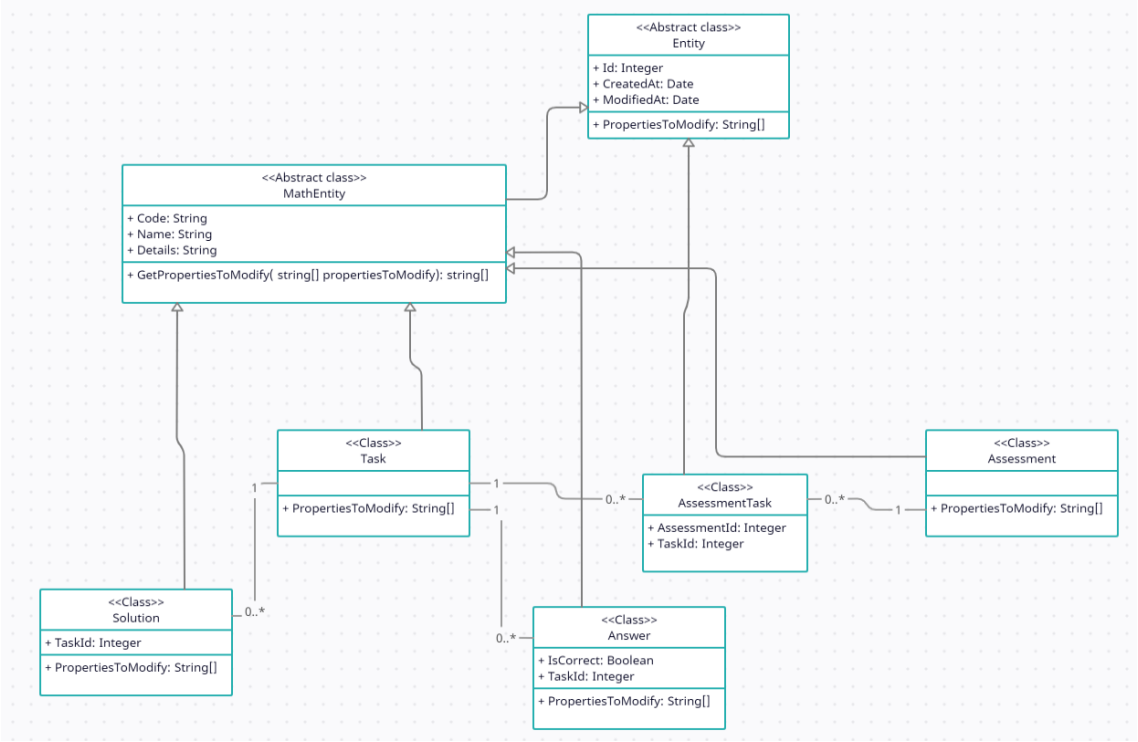
Andmete kiht on Entity Frameworki abstraktsioon, mille kaudu saab teha vajalikke andmebaasipäringuid. Selleks kasutati *Repository* ja *Unit of work* disainimustreid [8]. *Repository* peidab klientkoodi eest ära andmebaasiga suhtlemise detailid. Selleks on kasutusel repositooriumid - klassid, mis defineerivad vajalikud andmebaasiopetsioonid meetoditena. *Unit of work'*i eesmärgiks on jälgida andmebaasiobjektide muutumist ja salvestada muudatused ühtse transaktsioonina. Kuigi Entity Framework realiseerib juba ise need disainimustreid, võimaldab selle ümbritsemine lisakihiga jätta teenuskihi koodi muutumatuks juhul, kui tekib vajadus kasutada mõnda muud ORM'i teenusepakkujat.

Teenuste kiht on vahelüliks ülejäänud andmete kihi ja esitluskihi vahel. Siin toimub põhiline andmete töötlus ja asub suurem osa loogikast. Esitluskihist tulevate päringute põhjal küsitakse repositooriumi kaudu andmebaasiobjektid, teisendatakse DTO mudeliteksja antakse esitluskihile kasutamiseks (vt Lisa2). DTO'd on objektid andmete ülekandmiseks teenuskihist esitluskihti. Nende kasutamine peidab andmebaasi struktuuri detailid esitluskihi eest ja ühtlasi aitab mõnel juhul vähendada andmete ülekandmise mahtu: mitmest omavahel seotud klassist valitakse välja soovitud väljad, millest täidetakse kompaktsem klass.

Esitluskihiks on ASP.NET Core MVC. Mudelid on andmete ülekandmiseks sobivas formaadis. Vaade kasutab etteantud andmeid veebilehe koostamiseks. Kontrollerid küsivad andmeid teenuskihist. Selles rakenduses on kahte tüüpi kontrollereid. Ühed annavad saadud andmed edasi vaadetele ja teised REST API kaudu kliendirakendusele. Esitluskiht kasutab mudelitena võimalusel DTO'sid. Vajadusel on DTO ümber ehitatud *View Model*: mudel, mis sisaldab konkreetse vaate jaoks vajalikke välju. Antud rakenduses kasutatakse seda juhul, kui on vaja lisada midagi spetsiaalselt andmete sisestamise vormi jaoks, näiteks failide üleslaadimise väli.

Erinevad kihid kasutavad omavahel suhtlemiseks sõltuvuste vähendamist (*dependency injection*). Sõltuvuste vähendamine on tarkvara disainimuster, mille eesmärgiks on vähendada erinevate komponentide otsest sõltuvust üksteisest ja lihtsustada seeläbi ühiktestide kirjutamist. ASP.NET Core'i on sisse ehitatud funktsionaalsus, mille abil on ühes kohas võimalik defineerida seosed liideste ja neid implementeerivate klasside vahel. Kui klassi A konstruktorile lisada parameetrina klassiga B seostatud liides, algväärtustatakse B automaatselt (vt Lisa 3).

3.2 Andmete struktuur

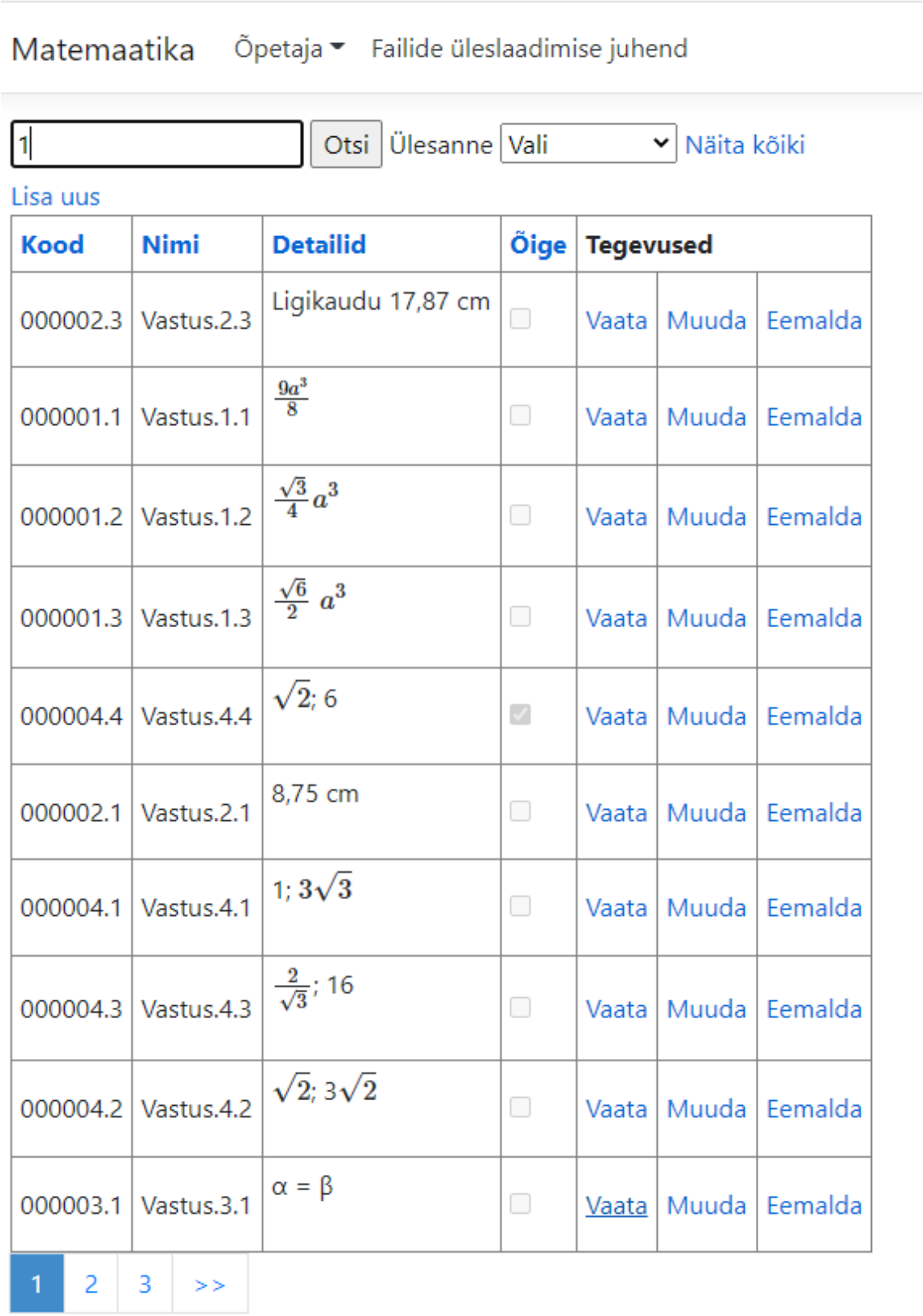


Joonis 3 Klassidragramm

Praegu on süsteemis 2 objekti, mis saavad iseseisvalt eksisteerida: kogumikud ja ülesanded. Kogumiku (*Assessment*) eesmärgiks on ülesannete grupeerimine. Ülesanne (*Task*) võib, aga ei pruugi kuuluda ühte või mitmesse kogumikku. Kogumiku ja ülesande vahel on mitu-mitmele seos. Selle seose kirjeldamiseks on lisatud abiobjekt (*AssessmentTask)*, mis sisaldab omavahel seotud kogumiku ja ülesande ID-sid. Ülesandega on seotud valikvastused (*Answer*), millest üks on õige ja ülejäänud valed ning üks või mitu lahendust (*Solution*). Kõik eelpool nimetatud objektid v.a *AssessmentTask* pärinevad abstraktsest matemaatika baasobjektist (*MathEntity*), kus on defineeritud nende ühised väljad. Matemaatika baasobjekt pärineb omakorda baasobjektist (*Entity*), mis sisaldab välju, millel on tähendus mis tahes objekti jaoks sõltumata valdkonnast.

3.3 Kasutajaliidese funktsionaalsus

Halduskeskkonna kasutajaliides on iga objektitüübi jaoks 5 vaadet: list objektidest, lisamine, detailid, muutmine ja kustutamine.



Joonis 4 Vastuste listivaade

Listivaates on näha leheküljetäis antud tüüpi objekte. Listivaatest pääseb ligi kõigile teistele vaadetele. Kogumiku ja ülesande listivaade sisaldab lisaks muule ka

kliendirakendusse suunavat linki selle konkreetse kogumiku või ülesande lahendamiseks. Listivaates on võimalik andmeid sorteerida erinevate väljade järgi ja leida soovitud tulemusi sisestades otsingukasti märksõna. Ülesannete lehel saab tulemusi filtreerida ka kogumiku järgi, vastuste ja lahenduste lehel aga ülesande järgi.

Lisamise vaade võimaldab kasutajal laadida üles sobivas formaadis Microsoft Wordi dokumente. Kogumiku üleslaadimisel salvestatakse uus kogumik, millega seostatakse failist loetud ülesanded. Ülesande või kogumiku üleslaadimisel salvestatakse ka ülesannetega seotud vastused ja lahendused. Vastuse ja lahenduse lisamisel tuleb menüüst valida ülesanne, millele soovitakse seda vastust/lahendust lisada, sest eraldiseisvana ei oleks neil objektidel mingit mõtet.

Muutmise vaates peab ülesannete, vastuste ja lahenduste tekstilise osa muutmiseks üles laadima uue faili, kuna tekst võib sisaldada matemaatilisi sümboleid, mille kirjutamine brauseris eeldab MathMl'i tundmist.

Kasutajaliidese menüüs on ka juhend failide üleslaadimiseks koos üleslaadimise malli ja õiges formaadis näidisfailidega.

3.4 Suhtlemine klientrakendusega

3.4.1 REST API

REST on kliendi ja serveri vaheliseks suhtluseks mõeldud arhitektuuristiil [9]. Tavaliselt kasutatakse REST API'des HTTP meetodeid, et teostada operatsioone andmetega. Antud süsteemis on klientrakendus tarbija rollis, seega kasutatakse ainult andmete pärimiseks mõeldud GET meetodit.

REST API võimaldab kliendirakendusel pärida halduskeskkonnast vajalikke andmeid. Selleks peab kliendirakendus tegema HTTP GET päringu halduskeskkonna pihta, mille tulemusena tagastatakse JSON formaadis vastus. Järgnevalt on välja toodud API otspunktid.

Tabel 1 API otspunktide kirjeldus

URL	Kirjeldus	Tagastatav vastus (JSON)

/api/assessments?page={0}&search={1}	Tagastab leheküljetäie kogumikke leheküljenumbri ja otsingusõna järgi. „page“ ja „search“ pole kohustuslikud parameetrid: vaikimise väärtused on 1 ja tühi string	Massiiv
/api/assessments{id}	Tagastab kogumiku id järgi	objekt või staatuskood 404
/api/assessments/{id}/tasks	Tagastab kogumiku id järgi kogumikku kuuluvad ülesanded	Massiiv
/api/tasks/{id}	Tagastab ülesande id järgi	objekt või staatuskood 404
/api/tasks/{id}/answers	Tagastab ülesande vastused suvalises järjekorras	Massiiv
/api/tasks/{id}/solutions	Tagastab ülesande lahenduse(d)	Massiiv

3.4.2 Ligipääs

Internetis on kasutusel turvameede, mis ei luba teha brauserist päringuid teise domeeni pihta (*same-origin policy*). Selleks, et võimaldada klientrakendusel andmeid pärida, tuli seadistada CORS. See on mehhanism, mis võimaldab serveril tuvastada, kas päringu päritolu (protokoll, host, port) kuulub lubatud nimekirja ja otsustada selle põhjal, kas lubada ligipääs ressurssidele [10].

3.5 Matemaatikaülesannete lugemine ja salvestamine failidest

3.5.1 Faili struktuur ja andmete lugemine

Iga ülesandega on seotud vastused ja lahendused. Seega peab kogumiku või ülesande üleslaadimiseks fail olema struktureeritud nii, et seal oleks võimalik eristada ülesandeid, mille all oleks alampunktidena võimalik eristada vastuseid ja lahendusi. Microsoft Wordis saab selliseks jaotamiseks kasutada stiile. Rakendus eeldab, et ülesanne on stiili all „Pealkiri1“ („Heading1“) ning vastused ja lahendused stiili all „Pealkiri2“ („Heading2“) ja need kõik oleks õigesti nimetatud (vastavalt „Ülesanne“, „Vastused“ (valed vastused), „Vastus“ (õige vastus) ja „Lahendus“).

.NET’il on olemas teek[1], mille abil saab töödelda Open XML formaadis Wordi dokumente. Andmete lugemiseks tükeldab rakendus faili lõikudeks. Iga lõigu juures vaadatakse lõigu stiili ja nimetust. Kui on tegemist ülesandega, luuakse uus ülesanne. Kui on tegemist vastuse või lahendusega, luuakse uus objekt ja lisatakse see viimati loodud ülesandele. Muudel juhtudel lisatakse lõigu sisu viimati loodud objekti teksti väljale. Lõpuks salvestatakse kõik ühe transaktsioonina andmebaasi.

Vastustel ja lahendustel pole seotud alamobjekte, seega nende üleslaadimiseks piisab sellest, kui faili sisuks on vastava vastuse või lahenduse tekst.

3.5.2 Andmete formaat

Microsoft Word kasutab matemaatiliste sümbolite kirjeldamiseks OMML keelt. Brauserid ei oska OMML’I lugeda. Teksti brauserisõbralikumale kujule saamiseks kasutab rakendus vabavaralist faili[2], mis transformeerib OMML’i MathML’iks.

[1] https://docs.microsoft.com/en-us/dotnet/api/documentformat.openxml?view=openxml-2.8.1

[2] https://github.com/transpect/docx2hub/blob/master/xsl/omml2mml/omml2mml.xsl

Ülesannete, vastuste ja lahenduste tekst sisaldab erinevas formaadis andmeid (tavaline tekst, pildid, tabelid, MathML). Seetõttu teisendatakse andmed enne salvestamist HTML'iks, et neid saaks brauseris kuvada.

3.6 Üldistamine

Korduva koodi vältimine on üks põhilisi puhta koodi kirjutamise põhimõtteid. Kui mitmes kohas on kasutusel sama kood, on tavaliselt võimalik kirjutada uus meetod ja tõsta korduv kood sinna sisse. Mõnikord kasutavad erinevat tüüpi objektid sama funktsionaalsust, kusjuures selle korduva funktsionaalsuse kasutamisel on vaja täpselt teada, mis tüüpi objektiga on tegemist. Ka sellisel juhul on võimalik koondada funktsionaalsus ühte meetodisse. Selleks kasutatakse üldistatud programmeerimise võtet (*Generics*), mis võimaldab lisada klassile tüübi parameetri ja kasutada seda klassi meetodites [11]. Parameetri tüüp määratakse klassi algväärtustamise hetkel. Antud rakenduses kasutatakse üldistamist peamiselt andmebaasioperatsioonides, aga ka listivaadete koostamisel.

3.7 Ühiktestid

Tagamaks rakenduse korrektset töötamist, prooviti võimalikult suur osa koodist katta ühiktestidega. Selleks kasutati Visual Studio poolt vaikimisi määratud testimise raamistikku MSTest.

Testide koostamisel kasutati AAA (*Arrange, Act, Assert*) mustrit [12]. See jagab iga ühiktesti kolmeks osaks:

1) *Arrange*: algväärtustatakse kõik objektid, mida kasutatakse selle testi jooksutamise käigus

2) *Act*: kutsutakse välja kood, mida soovitakse testida

3) *Assert*: Valideeritakse tulemusi. Kui eelmises punktis tagastatakse mingi väärtus, kontrollitakse, kas see on võrdne oodatud väärtusega.

Antud rakenduses on ühiktestid seatud vastavusse meetoditega: iga ühiktest kontrollib, kas testitav meetod töötab nagu peab. Meetodil võib olla palju väliseid sõltuvusi, mis võib muuta testimise tülikaks. Seetõttu kasutati mockimist: loodi võltsobjektid, millel on sama

signatuur pärisobjektidega. Selleks kasutati Moq[1] raamistikku, mis võimaldab defineerida mock-objektid koos tagastatavate väärtustega.

4 Tulemuste valideerimine

4.1 Koodi analüüs

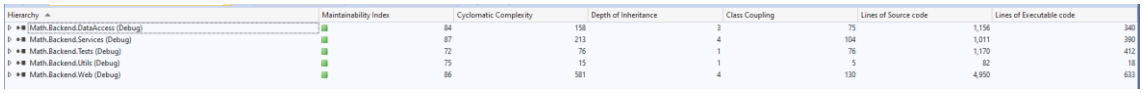
Koodi kvaliteedi ja hallatavuse hindamiseks kasutati Visual Studios olevaid analüüsitööriistu.

Kõigepealt vaadati testidega kaetavust.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
ville_LAPTOP-RCUARVKC 2021-0...	1761	49.87%	1770	50.13%
math.backend.dataaccess.dll	900	96.15%	36	3.85%
math.backend.services.dll	682	72.63%	257	27.37%
math.backend.tests.dll	60	4.89%	1167	95.11%
math.backend.web.dll	119	27.74%	310	72.26%

Joonis 5 Koodi kaetavus testidega rakenduse kihtide kaupa

Eesmärgiks oli katta võimalikult suur osa koodist ühiktestidega. Testitega kaetud koodi osakaal osutus loodetust väiksemaks. Testidega kaeti suur osa autori enda kirjutatud koodist, aga ei leitud lihtsat moodust geneerilise koodi testimiseks ja see jäi ajapuuduse tõttu teostamata.



Joonis 6 Koodimeetrikate analüüsi tulemus

Järgnevalt genereeriti Visual Studios koodimeetrikate analüüs, mis koosneb viiest komponendist [13]:

1) Hooldatavuse indeks (*Maintainability index*) – kokkuvõtlik number vahemikus 0-100, mis koondab endas erinevate meetrikate tulemused ja näitab, kui lihtne on

[1] https://github.com/moq/moq4

koodi hallata (mida suurem number, seda lihtsam). Heaks tulemuseks loetakse 20 või rohkem.

2) Tsükliline keerukus (*Cyclomatic complexity*) – number, mis näitab, mitu erinevat võimalikku teekonda on koodi läbimiseks. Mida suurem number, seda raskem on hallata.
3) Pärimise sügavus – maksimaalne pärimise hierarhia pikkus, liikudes alamklassist juurklassini. Väiksem number on parem.
4) Klasside seotus – mitut klassi antud klass kasutab. Väiksem number on parem.
5) Koodiridade arv
6) Käivitatavate koodiridade arv

Ülaltoodud jooniselt võib näha, et kokkuvõttes on koodimeetrikate analüüsi tulemus hea.

4.2 Failidest teksti lugemine sümboleid säilitades

Eesmärgiks oli võimaldada üleslaadida faile, kust loetakse välja ja salvestatakse ülesanded koos vastuste ja lahendustega, kusjuures säilima peaksid kõik matemaatilised sümbolid. Wordi failidest teksti lugemine osutus edukaks.

Katsetati ka erinevaid teenusepakkujaid PDF failide lugemiseks (nt iText[1], PdfSharp[2] ja PdfPig[3]) ja uuriti võimalusi, kuidas ise faile töödelda. Autoril ei õnnestunud leita meetodit, kuidas PDF failidest kadudeta matemaatilisi sümboleid lugeda.

4.3 Tagasiside

Rakendus anti testida kahele matemaatikaõpetajale. Üks õpetajatest pakkus välja võimaluse, kuidas rakendust kasutajasõbralikumaks muuta: kogumikke võiks saada grupeerida teemade ja sihtrühmade kaupa, et õpetajal oleks neid mugavam leida.

Testimise käigus tulid välja mõned vead, mis said parandatud. Näiteks ei arvestanud programm esialgu Wordi keele eripäradega. Töö loomise käigus kasutati inglise keelset

[1] https://github.com/itext

[2] http://www.pdfsharp.net/

[3] https://github.com/UglyToad/PdfPig

Wordi. Koodi debugimisel olid lõikude stiilide nimetused vastavalt „Heading1“ ja „Heading2“. Selgus, et eestikeelses Wordis loodud failides olid nende asemel „Pealkiri1“ ja „Pealkiri2“.

5 Kokkuvõte

Tööl oli kaks põhilist eesmärki. Esiteks luua keskkond, kuhu kasutaja saaks üles laadida faile, millest loetakse välja ja salvestatakse matemaatikaülesanded koos vastuste ja lahendustega, mida oleks võimalik samas keskkonnas hallata. Töö teiseks eesmärgiks on leida võimalus failist matemaatiliste sümbolite lugemiseks ja salvestamiseks nii, et need säiliksid arusaadaval ja loetaval kujul.

Lõputöö tulemusena valmis halduskeskkond, kuhu on võimalik ülesandeid üles laadida Microsoft Wordi failidest ja neid samas keskkonnas hallata. Failidest loetud ülesanded salvestatakse koos seotud vastuste ja lahendustega andmebaasi. Matemaatilised sümbolid salvestatakse sellises formaadis, et neid on võimalik brauseris loetaval kujul esitada.

Kasutatud kirjandus

[1] „Introduction to ASP.NET Core,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0.

[2] S. Smith, „Overview of ASP.NET Core MVC,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0.

[3] „Entity Framework Core,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/ef/core/.

[4] „Code-First vs Model-First vs Database-First: Pros and Cons,“ [Võrgumaterjal]. Available: https://www.ryadel.com/en/code-first-model-first-database-first-vs-comparison-orm-asp-net-core-entity-framework-ef-data/.

[5] [Võrgumaterjal]. Available: https://docs.fileformat.com/word-processing/docx/.

[6] „Mathematical Markup Language (MathML) Version 3.0 2nd Edition,“ [Võrgumaterjal]. Available: https://www.w3.org/TR/MathML3.

[7] „What is MathJax,“ [Võrgumaterjal]. Available: http://docs.mathjax.org/en/latest/basic/mathjax.html.

[8] M. Fowler, Patterns of Enterprise Application Architecture, 2002.

[9] W. i. REST. [Võrgumaterjal]. Available: https://restfulapi.net/.

[10] „Cross-Origin Resource Sharing (CORS),“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

[11] „Generics (C# Programming Guide),“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/.

[12] „Unit test basics,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2019.

[13] „Code metrics values,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2019.

[14] [Võrgumaterjal]. Available: https://h5p.org/getting-started.

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

Mina, Villem Mesila

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Matemaatikaülesannete üleslaadimise ja haldamise süsteem“, mille juhendaja on Gunnar Piho
   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.

18.05.2021

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.

Lisa 2 Andmete liikumise näide

1) Kasutaja teeb veebilehel päringu: https://localhost:44315/Tasks/Details/169

2) Kontroller püüab päringu kinni, küsib andmed teenuskihist ja annab hiljem vaatele edasi:

```
public ActionResult Details(int id)
{
    var task = _taskService.GetById(id);
    return View(task);
}
```

3) Teenus küsib andmed repositooriumist ja hiljem tagastab kontrollerile DTO'na:

```
public TDto GetById(int id)
{
    var dbEntity = _mathRepository.GetById(id);
    return _mathEntityCreator.FromDbObj(dbEntity);
}
```

4) Repositoorium küsib andmebaasist soovitava objekti:

```
public T GetById(int id, string[] relatedEntities = null)
{
    if (relatedEntities == null)
        return _dbSet.Find(id);
    var queryable = _dbSet.AsQueryable();
    foreach (var relatedEntity in relatedEntities)
    {
        queryable = queryable.Include(relatedEntity);
    }
    return queryable.FirstOrDefault(x => x.Id == id);
}
```

5) Repositoorium tagastab tulemuse teenusele.

6) Teenus teisendab andmebaasiobjekti DTO'ks:

```
public TDto FromDbObj(TDbObj dbObj)
{
    var dto = new TDto();
    ObjectHelper.CopyPropertyValues(dbObj, dto);
    return dto;
}
```

7) Teenus tagastab tulemuse kontrollerile

8) Kontroller annab saadud tulemuse edasi vaatele ja vaade koostab veebilehe:

```
@using Math.Backend.Services.Models
@using Math.Backend.Utils
@model TaskDto

@Html.DisplayFor(x => x.Code)
@Html.DisplayFor(x => x.Name)
@Html.DisplayFor(x => x.Details)
<a href="@Url.Action("Index", "Tasks")">@Constants.BackToList</a>
```

Lisa 3 Sõltuvuste vähendamise näide

Registreeritakse sõltuvused:

```
// This method gets called by the runtime. Use this method to add
services to the container.
public void ConfigureServices(IServiceCollection services)
{
    //Näite jaoks ebaoluline kood on välja jäätud
    AddDependencyInjection(services);
}
private void AddDependencyInjection(IServiceCollection services)
{
    services.AddScoped<IAnswerService, AnswerService>();
    services.AddScoped<IAssessmentService, AssessmentService>();
    services.AddScoped<ISolutionService, SolutionService>();
    //Näite jaoks lühendatud
}
```

Kontrolleri konstruktoritele antakse parameetrina kaasa liidesed, sealhulgas IAnswerService. Süsteem näeb registreeritud sõltuvusi, taustal luuakse uus AnswerService'i eksemplar.

```
[Route("api/[controller]")]
public class TasksController : Controller
{
    private readonly IAnswerService _answerService;
    private readonly ISolutionService _solutionService;
    private readonly ITaskService _taskService;
    public TasksController(IAnswerService answerService, ISolutionService
   solutionService, ITaskService taskService)
    {
        _answerService = answerService;
        _solutionService = solutionService;
        _taskService = taskService;
    }

    [HttpGet("{id}/answers")]
    public List<AnswerApiModel> GetAnswers(int id)
    {
        //Siin on teada, et tuleb pöörduda AnswerService'I meetodi poole
        return _answerService.GetTaskAnswersForApi(id);
    }
    //Näite jaoks lihtsustatud ja lühendatud

}
```

Jõutakse siia:

```
    public class AnswerService : MathBaseService<Answer, AnswerDto>,
    IAnswerService
    {
        private readonly IMathRepository<Answer> _answerRepo;

        public AnswerService(IMathRepository<Answer> answerRepo)
        {
            _answerRepo = answerRepo;
        }

        public List<AnswerApiModel> GetTaskAnswersForApi(int taskId)
        {
            var answers = _answerRepo.Find(x => x.TaskId == taskId);
            Random rnd = new Random();
            return answers.OrderBy((x) => rnd.Next()).Select(x => new
AnswerApiModel { Id = x.Id, Details = x.Details, IsCorrect = x.IsCorrect
}).ToList();
        }
        //Näite jaoks lihtsustatud ja lühendatud
    }
```