

УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ



АУТОМАТИЗАЦИЈА ПРОВЕРЕ ЗНАЊА НА КУРСЕВИМА КОЈИ ИЗУЧАВАЈУ БАЗЕ ПОДАТАКА

Мастер рад

Ментор:

проф. др Милош Цветановић

Кандидат:

Стефан Ђорђевић
3114/2017

Београд, Јул 2020.

САДРЖАЈ

САДРЖАЈ	I
1. УВОД.....	1
2. ОПИС ФУНКЦИОНАЛНОСТИ	2
2.1. АНАЛИЗА ПОСТОЈЕЋИХ РЕШЕЊА.....	2
2.2. АУТЕНТИКАЦИЈА	3
2.3. УСАВРШАВАЊЕ ЗНАЊА	3
2.4. ЕВАЛУАЦИЈА СИНТЕТИЧКОГ ЗНАЊА	4
2.4.1. Функционалности намењене студенту	4
2.4.2. Функционалности намењене професору.....	4
2.4.3. Процес евалуације	5
2.5. ЕВАЛУАЦИЈА АНАЛИТИЧКОГ ЗНАЊА	6
2.5.1. Функционалности намењене студенту	6
2.5.2. Функционалности намењене професору.....	7
2.5.3. Процес евалуације	7
3. ПРИМЕРИ УПОТРЕБЕ АПЛИКАЦИЈЕ	9
3.1. ПРИСТУП И АУТЕНТИКАЦИЈА	9
3.1.1. Неаутентикован корисник.....	9
3.1.2. Аутентикован корисник.....	10
3.1.3. Аутентикација и сагласност	10
3.2. ПРИМЕРИ УПОТРЕБЕ АПЛИКАЦИЈЕ ОД СТРАНЕ ПРОФЕСОРА	11
3.2.1. Шаблони	11
3.2.2. Задаци	15
3.2.3. Испит синтезе	16
3.2.4. Испит анализе.....	18
3.2.5. Моделирање базе података	19
3.2.6. Попуњавање садржаја базе података	20
3.2.7. Додељивање испита студентима	22
3.2.8. Резултати испита.....	23
3.3. ПРИМЕРИ УПОТРЕБЕ АПЛИКАЦИЈЕ ОД СТРАНЕ СТУДЕНТА	25
3.3.1. Преглед вежби	25
3.3.2. Преглед испита	26
3.3.3. Инстанца	26
3.3.4. Испит синтезе	27
3.3.5. Испит анализе.....	27
3.4. ПРИМЕРИ УПОТРЕБЕ АПЛИКАЦИЈЕ ОД СТРАНЕ АДМИНИСТРАТОРА	27
4. ТЕХНИЧКИ АСПЕКТИ ИМПЛЕМЕНТАЦИЈЕ	30
4.1. ТЕХНОЛОГИЈЕ И АЛАТИ	30
4.1.1. Клијентски део апликације	30
4.1.2. Серверски део апликације.....	31
4.1.3. Складиштење података	31
4.1.4. Алати	32
4.2. КЛИЈЕНТ-СЕРВЕР КОМУНИКАЦИЈА	32
4.3. ИНТЕГРАЦИЈА СА AZURE ACTIVE DIRECTORY	33
4.4. БАЗЕ ПОДАТАКА АПЛИКАЦИЈЕ	37
4.4.1. Примарна база података.....	37
4.4.2. Динамичке базе података и права приступа.....	39

4.5.	АРХИТЕКТУРА.....	40
4.6.	ЕВАЛУАЦИЈА ИСПИТА	42
5.	ЗАКЉУЧАК.....	45
	ЛИТЕРАТУРА.....	46
	СПИСАК СКРАЋЕНИЦА	48
	СПИСАК СЛИКА.....	49

1. Увод

Потребе индустрије за различитим информационим системима су у непрекидном порасту већ дужи низ година. Развој софтвера је напредовао у тој мери да готово не постоје техничке препреке за примену у било којој области. Много чешћа препрека јесте дефицит стручњака за развој информационих система. Као очигледно решење намеће се проширење капацитета образовних установа. Иако ово решава проблем са квантитативног аспекта, индиректно доводи до пада квалитета наставе. Већи број полазника подразумева већи обим посла за наставнике и то највише у контексту евалуације знања. Тиме се наставницима смањује простор за унапређење садржаја наставе и праћење новина из стручних области.

Примена информационих система у различитим областима довела је до стварања различитих програмских језика и платформи за развој софтвера. За функционисање већине оваквих система неопходна је нека врста базе података. Из тог разлога базе података су једна од основних области образовних програма који изучавају развој софтвера.

Циљ овог мастер рада јесте развој информационог система који би имао реалну примену у настави. Реч је о интернет апликацији чија је основна намена аутоматизација процеса евалуације знања. Функционалности система прилагођене су курсу који се бави изучавањем релационих база података, а чије је циљно окружење Мајкрософтов *SQL Server* [1].

У поглављу 2. овог документа биће анализирана идеја аутоматизације процеса евалуације знања. Као резултат ове анализе биће дефинисане функционалности које апликације треба да омогући.

Начин на који апликација остварује поменуте функционалности биће описан у поглављу 3. овог документа. Биће појашњени ентитети које апликација дефинише као и релације међу њима. Значајан део поглавља фокусираће се на опис графичког корисничког интерфејса као и начине интеракције корисника са системом.

У поглављу 4. биће објашњени основни технички аспекти везани за имплементацију функционалности. Део поглавља изложиће архитектуру апликације и употребу готових решења њеној имплементацији. Остатак поглавља фокусираће се на анализу појединачних техничких проблема и начина на који су превазиђени.

2. ОПИС ФУНКЦИОНАЛНОСТИ

У овом поглављу ће бити изложени описи функционалности које апликација пружа. Најпре ће бити изложена кратка анализа постојећих софтверских решења. Затим ће бити описана аутентикација као једна од основних функционалности. Затим ће бити описане функционалности које омогућавају усавршавање знања студената. На крају ће бити описане функционалности везане за евалуацију знања студената. За сваку функционалност биће размотрена мотивација која је довела до њеног настанка.

2.1. Анализа постојећих решења

Постоји велики број софтверских решења која имају примену у образовању. Једна група софтвера оријентисана је ка процесу учења. Платформа *W3 Schools* нуди курсеве о различитим технологијама везаним за интернет апликације. Бесплатна је, а курсеви су организовани тако да излажу појединачне концепте на конкретним примерима. У случају курса који се бави релационим базама података постоји могућност измене и извршавања упита који су дати као примери. Платформе *SQLBolt* [2] и *SQLZOO* [3] такође су бесплатне и нуде сличан приступ. Након изложеног концепта од полазника се очекује да синтетише један или више упита који испуњавају одређене захтеве. Платформе аутоматски проверавају исправност ових упита. Излагање садржаја у виду темељно припремљених видео записа је нешто што карактерише платформе као што су *Pluralsight* [4] и *Udemy* [5]. Одређени курсеви су интерактивни од којих неки подржавају и анализу кода. Када се ради о курсевима који се тичу језика *SQL* (*Structured Query Language*) интерактивност се своди на оно што нуде платформе *SQLBolt* и *SQLZOO*. Одређени курсеви су бесплатни, али већина захтева накнаду. Поменуто платформе примарно су намењене за примену у процесу учења, али се може уочити да су одређени концепти применљиви и за проверу знања.

Са друге стране постоје софтверска решења оријентисана ка процесу евалуације знања. *Exam Builder* [6] представља комбинацију софтвера за учење и евалуацију. Евалуација се заснива на скупу питања различитог типа. Подржани облици питања су одабир једног или више тачних тврдњи, одређивање да ли је тврдња тачна или нетачна, попуњавање недостајућег појма и повезивање тврдњи. Оваква питања превасходно су погодна за евалуацију теоријског знања. Платформа *TALVIEW* [7] нуди и могућност полагања испита у виду есеја, математичких испита као и испита кодирања у преко 52 програмска језика. Ове и сличне платформе нуде и велики број пратећих функционалности као што су различите врсте провере идентитета, заказивање испита, временска ограничења и статистичка анализа. Посебно се истиче платформа *Moodle* [8] која омогућава надоградњу инсталацијом различитих додатака. Један такав додатак јесте *Code Runner* [9]. Подржава језике као што су *Python*, *Java*, *C*, *C++*, *PHP*, *Java Script* укључујући и *SQL*. Након што синтетише упит полазник има могућност да провери исправност свог одговора. Уколико је упит нетачан приказује се очекивани и добијени резултат. Након нетачног одговора полазник може да коригује упит и покуша поново. Свака провера за коју се испостави да одговор није тачан носи казнене поене.

Поменута решења на различите начине приступају евалуацији знања. Фокус ових решења је провера теоријског знања или способности синтетисања решења. У пракси ове вештине нису довољне. Стручњаци велики део свог времена проводе надограђујући постојеће софтверске системе и отклањајући грешке. Способност аналитичког размишљања је пресудна за успешно обављање оваквих задатака.

2.2. Аутентикација

Апликација препознаје две врсте корисника: професор и студент. Разлика између ове две врсте корисника огледа се у томе што је већина функционалности доступна или једној или другој врсти корисника. Како би извршила рестрикцију приступа функционалностима апликацији је неопходна информација о врсти корисника.

Поред рестрикције приступа функционалностима треба размотрити и рестрикцију приступа подацима. Део те рестрикције такође се заснива на врсти корисника - неки подаци су доступни само професорима. У случају професора, између појединачних корисника не постоји разлика у правима приступа подацима. Са друге стране, у случају студената, неки подаци су доступни свим док су други доступни појединачним корисницима. Како би извршила рестрикцију приступа подацима апликацији је неопходна информација о врсти и идентитету корисника.

Аутентикација подразумева да корисник унесе своје креденцијале (корисничко име и лозинку) на основу којих се утврђује његов идентитет. На основу идентитета одређује се и врста корисника којој идентитет припада. Будући да сви студенти Електротехничког факултета данас поседују корисничке налоге на платформи AAD (*Azure Active Directory*) [10] исте креденцијале могу користити како би се аутентиковали на апликацију која је предмет овог рада. На тај начин се корисницима олакшава процес аутентикације из неколико аспеката. Најпре, не постоји потреба да корисници пролазе кроз процес креирања засебног налога пре првог коришћења апликације. Затим, за кориснике није неопходно да знају још једне креденцијале и самим тим се смањује ризик да их изгубе или забораве. На крају, аутентикацијом на неку од апликација у оквиру AAD организације корисник аутоматски постаје аутентикован на ову апликацију (и обрнуто) па се тиме смањује учесталост уношења креденцијала. Исти принцип важи и код одјављивања из апликације.

2.3. Усавршавање знања

У овом одељку биће размотрене функционалности које омогућавају студентима да усаврше стечена теоријска знања као и начин на који професори управљају овим процесом.

Замисао је да када студенти стекну одређена теоријска знања иста могу да провере и додатно утврде кроз праксу односно извршавањем *SQL* упита. Текст упита се уноси у текстуално поље које је специфично по томе што препознаје и истиче кључне речи *SQL* језика. Текстуално поље за унос *SQL* упита пружа и могућност аутоматског довршавања кључних речи, назива табела и колона. Ово у великој мери олакшава писање упита. Са друге стране на студенту је да познаје семантику и начин конструисања *SQL* наредби. Тако састављен упит студент може да изврши и види резултат његовог извршавања. Дозвољено је да упит чита или мења податке. Поред извршавања упита студент има могућност да види или измени садржај сваке табеле користећи графички кориснички интерфејс. На располагању је и текстуални опис модела базе као и могућност чувања састављених упита.

Базе над којима студенти извршавају упите креирају професори. За сваку од ових база професор дефинише модел базе извршавањем *SQL* наредби. Поред модела, професор дефинише јединствено име базе и текстуални опис њеног модела. Могуће је дефинисати и почетне податке у бази извршавањем *SQL* наредби или коришћењем графичког корисничког интерфејса.

Над овако креираним базама студенти немају директан приступ. У супротном би долазило до преклапања. Студенти би једни другима мењали сачуване упите и податке у бази. Сви студенти имају приступ списку база које су професори креирали у сврху усавршавања знања студената. На основу сваке од тих база студент може да креира једну или више копија над којом само он има приступ. Овим се у потпуности гарантује изолованост окружења као и очувања оригиналне базе креиране од стране професора. Приликом копирања студент наводи јединствен назив своје копије како би разликовао копије креиране од исте оригиналне базе.

2.4. Евалуација синтетичког знања

Подразумева процену способности студента да састави (сингетише) упит који ће вратити тражени скуп података. У овом одељку ће најпре бити описане функционалности намењене професору. Затим ће бити описане функционалности намењене студенту. На крају ће бити описан процес евалуације синтетичког знања.

2.4.1. Функционалности намењене студенту

Студенту је на располагању текстуални опис модела базе и текст самог задатка. Резултат упита који студент сингетише треба да буде једна табела. Формат табеле (имена колона) и опис тражених података дефинишу се у тексту задатка. На студенту је да најпре разуме описани модел и текст задатка, а затим састави упит који враћа тражени резултат. На располагању је и графички кориснички интерфејс којим се може видети садржај сваке табеле.

Студент има могућност да изврши упит и види резултат његовог извршавања. У овом случају није дозвољено да упит мења податке базе. Након извршавања упита студенту је познато да ли је резултат задовољио тражени формат. Уколико решење не задовољава тражени формат није могуће предати упит на оцењивање. Студенту није познато да ли се у резултату извршавања налази тражени скуп података.

2.4.2. Функционалности намењене професору

Професор креира испит синтетичког знања. Неопходно је да дефинише име испита, модел базе, текстуални опис модела, текстуални опис задатка, решење задатка у виду *SQL* упита и два скупа података: јавни и тајни. Јавни скуп података је доступан студенту приликом синтезе решења, а користи се и приликом процеса евалуације. Тајни скуп података није доступан студенту и користи се искључиво приликом процеса евалуације. Замисао је да се у тајном скупу података дефинише нека врста граничног случаја који ће довести до тога да извршавање студентовог упита резултује грешком или табелом која није у складу са захтевима дефинисаним у задатку. На овај начин се од студената захтева да предвиде могуће варијације у скуповима података над којим ће се њихов упит извршавати. Упити које састављају треба да буду тачни у општем случају, а не да се фокусирају на неколико пробних сценарија.

Модел дефинисан за испит синтезе заједно са јавним скупом података може се ставити на располагање студентима као база над којом могу да усавршавају знања. На тај начин студенти долазе на испит синтезе где им је модел базе унапред познат. На овај начин ставља се акценат на решавање проблема који дефинише текст задатка.

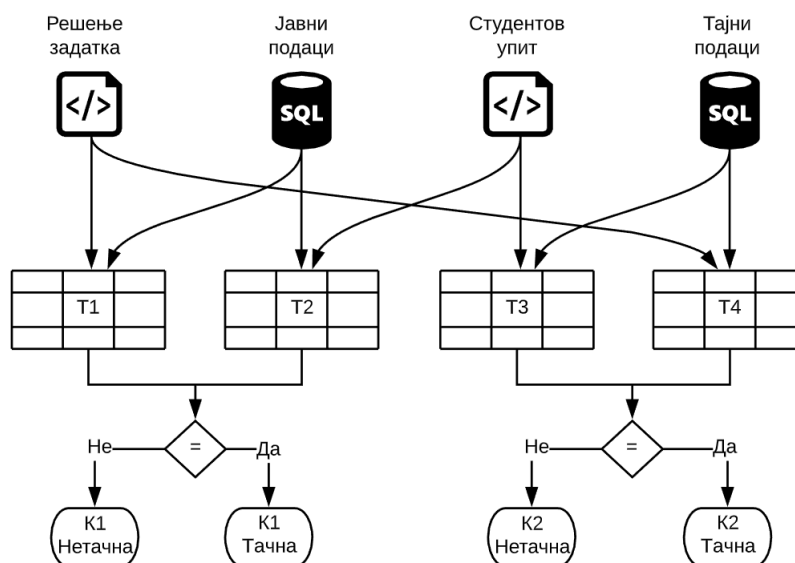
Након што креира испит професор га додељује појединачним студентима. Испит је видљив само оним студентима којима је додељен. Испити се у сваком тренутку налазе у једном од три стања (статуса): заказан, у току или завршен. Иницијално, испит је заказан, његов садржај сакривен и није га могуће полагати. Пребацавањем у стање у току означава се почетак испита, садржај испита постаје видљив и могуће је предати одговор. Пребацавањем у завршно стање означава се крај испита, више није могуће предати одговор и испит је спреман за оцењивање.

2.4.3. Процес евалуације

Оцена испита синтезе састоји се из две компоненте: K1 и K2. Свака компонента на крају процеса добија вредност тачно или нетачно. Обе компоненте проверавају да ли студентово решење даје резултат који је у складу са захтевима задатка. Компонента K1 ово чини за случај када се упит извршава над јавним скупом података, а компонента K2 за случај извршавања над тајним скупом података.

Процес најпре генерише четири табеле: T1, T2, T3 и T4. Табеле T1 и T4 добијају се као резултат извршавања решења задатка дефинисаног од стране професора. Табеле T2 и T3 добијају се као резултат извршавања упита синтетисаног од стране студента. За генерисање табела T1 и T2 користе се јавни подаци. За генерисање табела T3 и T4 користе се тајни подаци.

Вредност компоненте K1 добија се поређењем табела T1 и T2. Вредност компоненте K2 добија се поређењем табела T3 и T4. У оба случаја вредност компоненте добија се поређењем на једнакост. Упоредивање табела подразумева упоређивање њихових формата и њихових садржаја. Под форматом се подразумевају називи колоне и укупан број колоне. Дозвољено је да се редослед колоне разликује. Уколико студентов упит приликом неког од извршавања резултује грешком сматра се да је компонента оцене под којом се извршавао нетачна. Дијаграм на слици 2.4.3.1 илуструје процес евалуације синтетичког знања.



Слика 2.4.3.1 Евалуација синтетичког знања

Процес евалуације покреће се ручно од стране професора, а извршава самостално. Евалуацију је могуће започети тек након што испит пређе у завршно стање. Процес се покреће истовремено за све студенте. Након завршетка евалуације за сваког студента приказује се вредност обе компоненте оцене. Студенти који нису полагали испит односно студенти који су полагали испит али нису синтетисали никакав упит су посебно назначени и над њима се не покреће процес евалуације. Након завршетка испита студент у систему види да ли је испит полагао, да ли је оцењен и оцену исправности појединачних компоненти. Уколико постоји сумња у исправност процеса евалуације професору је на увид доступан упит синтетисан од стране студента.

2.5. Евалуација аналитичког знања

Подразумева процену способности студента да разуме упите које није сам синтетисао и предвиди резултат који тај упит даје када се изврши над неким скупом података. У овом одељку ће најпре бити описане функционалности намењене професору. Затим ће бити описане функционалности намењене студенту. На крају ће бити описан процес евалуације аналитичког знања.

2.5.1. Функционалности намењене студенту

Студенту је на располагању текстуални опис модела базе, текст задатка и упит. Упит је синтетисан од стране другог студента приликом полагања испита синтезе. Представља нетачно решење описаног задатка. Решење, иако нетачно у општем случају, за одређене скупове података може вратити резултат који је у складу са захтевима задатка. Један од таквих скупова јесте јавни скуп података испита синтезе. Из тог разлога, приликом полагања испита анализе једна од ставки која се захтева од студента јесте да наведе гранични случај односно попуњавање базе које ће приликом извршавања упита над њом довести до грешке или до резултата који није у складу са захтевима задатка. За дефинисање овог скупа података студент на располагању има графички кориснички интерфејс помоћу којег је могуће видети и изменити садржај сваке табеле у бази података. Модел базе је идентичан моделу базе описаном у задатку. Почетни подаци идентични су јавном скупу података испита синтезе.

Приликом полагања испита анализе не постоји могућност извршавања упита који се анализира. Самим тим студент ни у једном тренутку нема потврду да ли је пронашао неки од граничних скупова података. И поред тога постоји одређена вероватноћа да се такав скуп пронађе насумичним уношењем података. Тиме би студенти потенцијално дошли до тачног решења испита анализе без било каквог разумевања упита или захтева задатка.

Како би се ово превазишло од студента се захтева да унесе још два скупа података. У овом случају скупови података нису базе података већ табеле. Једна табела представља тачан резултат односно резултат који би се добио извршавањем тачног решења испита синтезе над базом података коју је студент дефинисао као гранични случај. На овај начин се гарантује да је студент успешно анализирао захтеве задатка. Друга табела представља нетачан резултат односно резултат који би се добио извршавањем упита који се анализира над базом података коју је студент дефинисао као гранични случај. На овај начин се гарантује да је студент успешно анализирао упит који представља предмет анализе. За дефинисање садржаја ових табела студент на располагању има графички кориснички интерфејс помоћу којег је могуће видети и изменити садржаје двеју табела. Табеле немају почетни садржај. Формати табела одговарају формату траженог резултата дефинисаног у тексту задатка.

2.5.2. Функционалности намењене професору

Професор креира испит аналитичког знања. Испит анализе креира се на основу одговора једног студента на неки од испита синтезе. Испит синтезе претходно мора бити завршен и оцењен. Одговор студента треба бити такав да јавна компонента оцене буде тачна, док тајна компонента треба да буде нетачна. Као и код испита синтезе, приликом креирања испита професор дефинише јединствено име испита.

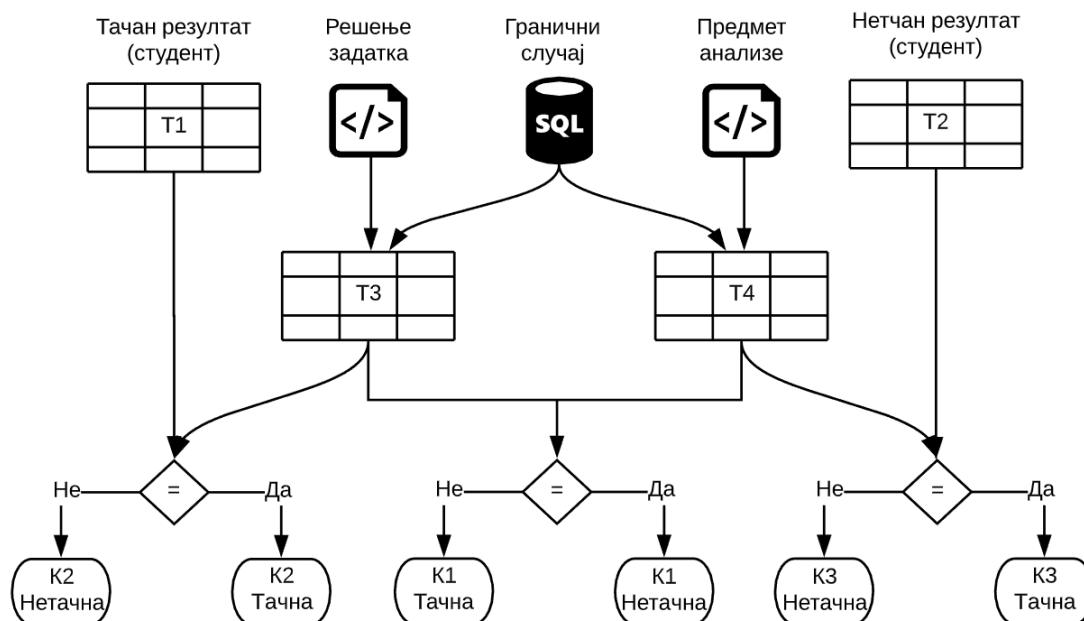
Након креирања испитом анализе се управља на исти начин као и испитом синтезе. Подразумева се додељивање испита студентима као и управљање статусима. Статуси испита анализе идентични су статусима испита синтезе и носе иста значења.

2.5.3. Процес евалуације

Оцена испита анализе састоји се из три компоненте: K1, K2 и K3. Свака компонента на крају процеса добија вредност тачно или нетачно. Компонента K1 проверава да ли се извршавањем упита који је предмет анализе над базом података коју је студент дефинисао као гранични случај долази до резултата који није у складу са захтевима задатка. Компонента K2 проверава да ли табела T1 коју је студент дефинисао као тачан резултат у граничном случају одговара захтевима задатка. Компонента K3 проверава исправност табеле T2 коју је студент дефинисао као излаз предмета анализе у граничном случају.

Процес најпре утврђује излаз тачног решење (табела T3) и стварни излаз предмета анализе (табела T4) у граничном случају. Табела T3 добија се извршавањем решења задатка над граничним случајем. Табела T4 добија се извршавањем предмета анализе над граничним случајем.

Вредност компоненте K1 добија се поређењем табела T3 и T4. Вредност компоненте K2 добија се поређењем табела T1 и T3. Вредност компоненте K3 добија се поређењем табела T2 и T4. У случају компоненте K1 врши се поређење на различитост. У случају компоненти K2 и K3 врши се поређење на једнакост. Као и код испита синтезе, поређење табела подразумева поређење њихових формата и садржаја. Дијаграм на слици 2.5.3.1 илуструје процес евалуације аналитичког знања.



Слика 2.5.3.1 Евалуација аналитичког знања

Процес евалуације покреће се ручно од стране професора, а извршава самостално. Евалуацију је могуће започети тек након што испит пређе у завршно стање. Процес се покреће истовремено за све студенте. Након завршетка евалуације за сваког студента се приказује вредност сваке компоненте оцено. Студенти који нису полагали испит односно студенти који нису бар дефинисали гранични случај посебно су назначени и над њима се не покреће процес евалуације. Након завршетка испита студент у систему види да ли је испит полагао, да ли је оцењен и оцену исправности појединачних компоненти оцено. Уколико постоји сумња у процес евалуације професору су на увид доступни сви подаци које је студент унео током полагања испита.

3. ПРИМЕРИ УПОТРЕБЕ АПЛИКАЦИЈЕ

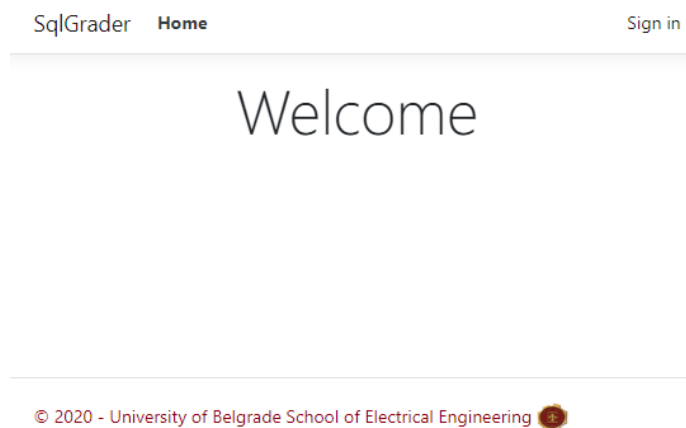
У овом поглављу ће бити изложени начини на које апликација остварује функционалности описане у претходном поглављу. Биће објашњени ентитети система као што су: шаблон, задатак, испит (синтезе и анализе) и резултати испита. Акценат ће бити на прегледу графичког корисничког интерфејса који омогућава управљање овим ентитетима. Након упознавања са садржајем овог поглавља читалац ће бити у стању да самостално користи апликацију.

3.1. Приступ и аутентикација

Приступ апликацији могућ је како аутентикованим тако и неаутентикованим корисницима. У погледу приступа одређеним страницама и функционалностима апликација разликује неаутентикованог корисника, аутентикованог студента и аутентикованог професора. У овом одељку биће објашњене разлике између ових корисника као и поступак давања сагласности корисника.

3.1.1. Неаутентикован корисник

Корисник чији идентитет није претходно утврђен уношењем креденцијала сматра се неаутентикованим корисником. Овакви корисници имају приступ само почетној страни. На слици 3.1.1.1 приказана је почетна страна апликације којој приступа неаутентикован корисник.

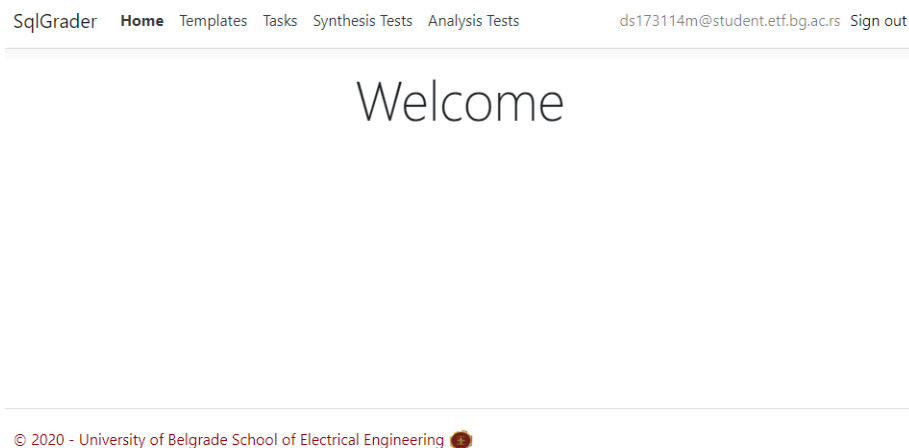


Слика 3.1.1.1 Почетна страна неаутентикованог корисника

Општа структура свих страна у апликацији је иста: заглавље и подножје између којих се налази главни садржај. Кликом на садржај у подножју отвара се нова картица претраживача са адресом сајта Електротехничког факултета. Заглавље садржи назив апликације, секцију за навигацију и секцију за аутентикацију. Тренутна страна означена је подебљаним текстом у секцији за навигацију. У случају неаутентикованог корисника секција за навигацију садржи само почетну страну. Секција за аутентикацију неаутентикованог корисника садржи дугме за аутентикацију.

3.1.2. Аутентикован корисник

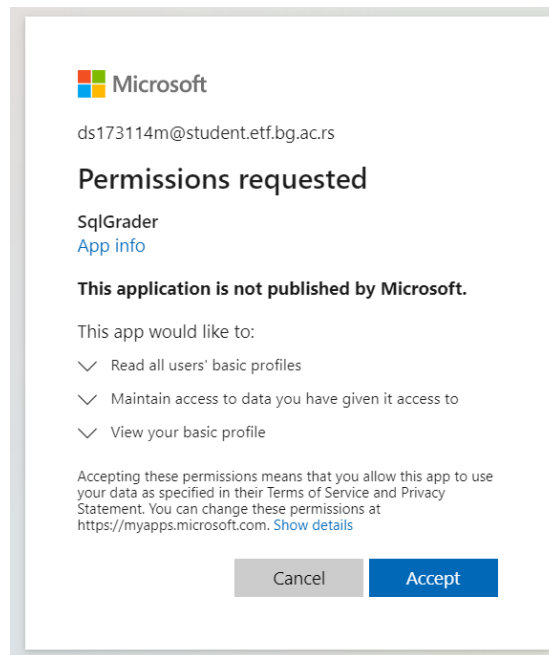
Корисник чији је идентитет претходно утврђен уношењем креденцијала сматра се аутентикованим корисником. Овакви корисници имају приступ свим деловима система као и неаутентиковани корисници. Професори додатно имају приступ страницама за табеларни приказ шаблона, задатака, испита синтезе и испита анализе. Студенти, у односу на неаутентиковане кориснике, додатно имају приступ и страницама за усавршавање знања и полагање испита. У секцији за аутентикацију корисника налази се корисничко име аутентикованог корисника и дугме за одјављивање са ситема. На слици 3.1.2.1 приказана је почетна страна аутентикованог професора.



Слика 3.1.2.1 Почетна страна аутентикованог професора

3.1.3. Аутентикација и сагласност

Притиском на дугме за аутентикацију корисник бива преусмерен Мајкрософтов портал за аутентикацију. Процес је прилично једноставан и своди се на уношење корисникових креденцијала. Аутентикација не представља део саме апликације и с тога неће бити детаљније описана. Са друге стране давање сагласности, које се такође врши на Мјакрософтовом порталу, директно зависи од саме имплементације. Приликом аутентикације од свих корисника захтева се сагласност за читање информација њиховог корисничког профила као и одржавање претходно датих сагласности. Ово су уједно и све сагласности које се захтевају од студената. Када се ради о професорима, поред већ поменутих, захтева се и сагласност читања основних информација других корисничких профила. Ова сагласност захтева се тек након приступа некој од страница на којима је она неопходна. Пример су странице табеларног прегледа шаблона, задатака и испита на којима је неопходно приказати информације о ауторима. Још један пример је страница за додељивање испита где је неопходно приказати информације о студентима. На слици 3.1.3.1 приказан је захтев за сагласност професора приликом приступа једној од таквих страна.



Слика 3.1.3.1 Давање сагласности професора

Сагласности дате апликацији корисник може укинути коришћењем Мајкрософтове интернет платформе. Потребно је, уз евентуалну претходну аутентикацију, отићи на адресу странице за управљање AAD налогом. Међу апликацијама треба пронаћи апликацију за коју се укида сагласност и уклонити ту апликацију. Приступ апликацији је и даље могућ али уз поновно давање сагласности.

3.2. Примери употребе апликације од стране професора

3.2.1. Шаблони

Означавају ентитете преко којих се дефинишу базе података. Над шаблонима се креирају задаци који се користе у испитима синтезе. Могу се учинити доступним студентима као базе података за усавршавање знања. Сами шаблони креирају се независно од других ентитета. Приликом креирања шаблона потребно је дефинисати: име шаблона, модел базе, текстуални опис модела базе и податке унутар саме базе.

На засебној страни апликације професору је доступан табеларни преглед постојећих шаблона. Табела има шест колона при чему првих пет имају информативну улогу, а последње две управљачку. У зависности од броја записа које је неопходно приказати садржај табеле се приказује на једној или више страна табеле. Маскималан број записа који се приказује у оквиру једне стране табеле налази се је у нумеричком пољу изнад заглавља последње колоне. Подразумевана вредност је пет. Коришћењем стрелица корисник може да повећа или смањи ову вредност у корацима од по пет. Могуће је унети и произвољну вредност. На слици 3.2.1.1 приказан је садржај стране за табеларни преглед шаблона.

Name				5	
Name	Description	Author	Created on	Public	Create
Библиотека	База података јед...	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	30/06/2020 15:18	<input checked="" type="checkbox"/>	Actions
Računovođa	-	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	10/06/2020 12:34	<input checked="" type="checkbox"/>	Actions
Dom zdravlja	Model baze čini 1...	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	10/06/2020 12:09	<input type="checkbox"/>	Actions
Restoran	-	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	09/06/2020 21:14	<input checked="" type="checkbox"/>	Actions
Електронски часопис	-	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	09/06/2020 21:12	<input type="checkbox"/>	Actions

Prev
1
2
3
Next

Слика 3.2.1.1 Табеларни преглед шаблона

Прве две колоне приказују име шаблона и текстуални опис модела базе, респективно. Име шаблона мора имати најмање један а највише 50 карактера. Опис модела базе нема ограничења у погледу броја карактера. У табеларном прегледу приказују се највише првих 20 карактера.

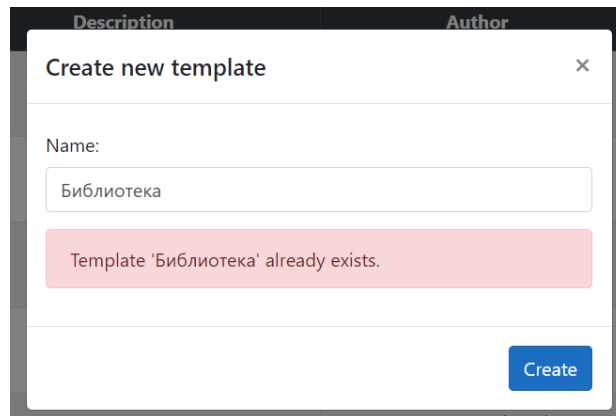
Професори имају приступ шаблонима које су сами креирали као и шаблонима које су креирали други професори. Трећа колона дефинише аутора односно корисника који је креирао шаблон. У првом реду стоје име и презиме аутора. Како име и презиме не морају нужно бити јединствени у другом реду приказано је корисничко име аутора.

Четврта колона приказује датум и време креирања шаблона. У првом реду приказан је датум, а у другом време креирања. Формати датума и времена зависе од уређаја на ком се апликација користи односно од одабране временске зоне и задатих формата.

Претпоследња колона има информативну и управљачку улогу. Садржи једно поље за потврду. Уколико је поље одабрано шаблон је доступан студентима као база података за усавршавање знања. Иницијално, након што се шаблон креира, поље није одабрано. Корисник накнадно може да промени вредност овог поља једном или више пута.

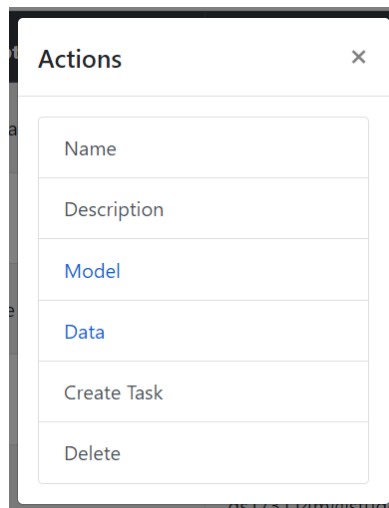
Последња колона не садржи никакав информативни садржај. У заглављу колоне налази се дугме које служи за креирање нових шаблона. У сваком од редова последње колоне налази се дугме које омогућава приступ акцијама које се могу извршавати над шаблоном.

Притиском на дугме за креирање шаблона кориснику се приказује прозор за креирање шаблона. Прозор садржи текстуално поље у које се уноси назив шаблона и дугме за потврду. Затварањем прозора за креирање шаблона корисник одустаје од креирања истог. Није могуће креирати шаблон уколико није попуњен назив или уколико шаблон са истим називом већ постоји. У оба случаја, испод текстуалног поља приказује се одговарајућа порука о грешци. У случају успешног креирања шаблона прозор се затвара, а садржај табеле освежава. На слици 3.2.1.2 приказан је прозор за креирање шаблона у ситуацији када креирање шаблона није успело јер унети назив није јединствен.



Слика 3.2.1.2 Неуспешно креирање шаблона

Притиском на дугме за приступ акцијама приказује прозор са списком акција које се могу извршити над шаблоном. На слици 3.2.1.3 приказан је прозор са списком акција за управљање шаблоном. Одабиром једне од акција означене плавим текстом корисник бива преусмерен на посебну страну за извршавање те акције. Конкретан пример су акције за дефинисање (и измену) модела и података у самој бази, респективно. Одабиром неке од преосталих акција корисник остаје на истој страни, а прозор са списком акција замењује се прозором за извршавање одабране акције.

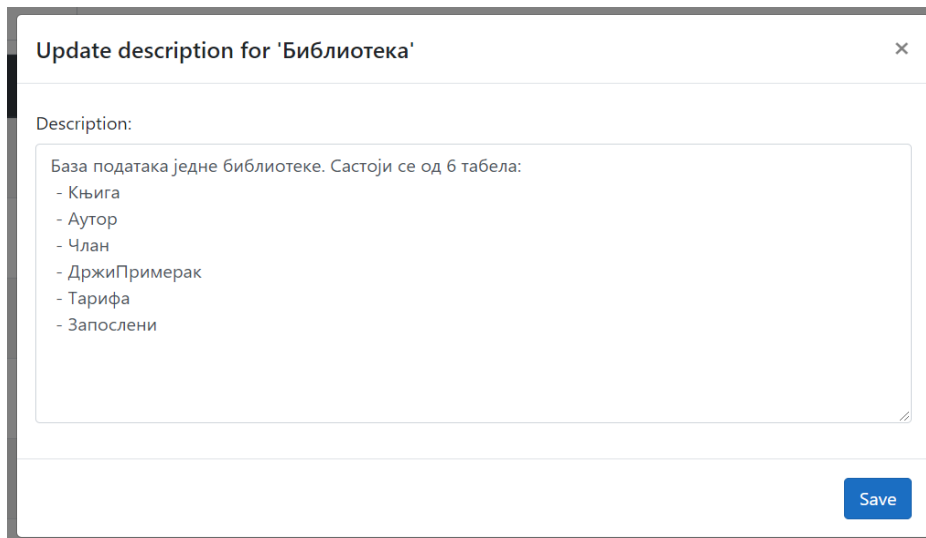


Слика 3.2.1.3 Акције за управљање шаблоном

Прва и претпоследња акција служе за промену назива шаблона и креирање задатака над шаблоном, респективно. Прозори за извршавање ових акција су готово идентични прозору приказаном на слици 3.2.1.2. Једина функционална разлика је у случају промене назива где је текстуално поље иницијално попуњено тренутним називом шаблона. Као и приликом креирања шаблона, нови назив мора бити јединствен. У оба случаја постоје и мање текстуалне разлике: заглавље прозора, лабела изнад поља текстуалног уноса и лабела дугмета за потврду.

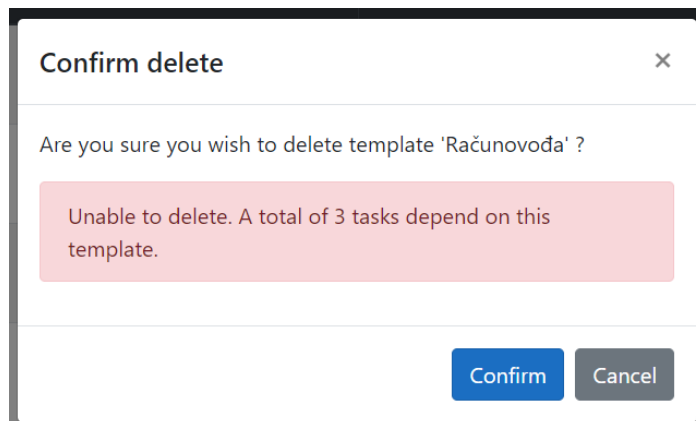
Друга акција служи за дефинисање или измену текстуалног описа шаблона. Прозор за извршавање ове акције је готово идентичан прозору који се користи за измену назива шаблона. Једина функционална разлика је што текстуално поље дозвољава унос у више редова. Поље је иницијално попуњено тренутним описом шаблона. Постоје мање текстуалне

разлике: заглавље прозора и лабела дугмета за потврду. На слици 3.2.1.4 приказан је прозор за дефинисање или измену описа шаблона.



Слика 3.2.1.4 Прозор за дефинисање/измену описа шаблона

Последња акција служи за брисање шаблона. За разлику од претходних акција, прозор за извршавање ове акције не прикупља никакве податке. Прозор служи искључиво да корисник потврди намеру о брисању. На овај начин смањује се вероватноћа да корисник ненамерно обрише неки шаблон. Кориснику су на располагању дугме за потврду и дугме за одустајање од брисања. Затварањем прозора корисник такође одустаје од брисања. Да би операција брисања била успешно извршена не сме постојати ни један задатак креиран над датим шаблоном. У случају успешног брисања прозор се затвара, а табела освежава. У супротном, унутар прозора се приказује порука о грешци. На слици 3.2.1.5 приказан је прозор за потврду брисања шаблона у случају када брисање није успело.



Слика 3.2.1.5 Неуспешно брисања шаблона

Претраживање шаблона могуће је по називу, а сортирање по називу и датуму креирања. Могуће је сортирање у растућем или опадајућем поретку. Филтрирање се врши уношењем текста у текстуално поље изнад заглавља колоне. Уколико је текстуално поље празно приказују се сви шаблони и не врши се филтрирање. Уколико текстуално поље има вредност, приказују се шаблони који у свом називу садрже вредност задату филтером. Мала и велика слова се игноришу приликом филтрирања. Притиском на заглавље колоне врши се сортирање у опадајућем поретку. Наредним притиском на исто заглавље мења се смер

сортирања. Црвеном стрелицом означена је колона и смер по коме су записи тренутно сортирани. Иницијално, табела нема филтере, а записи су сортирани према датуму креирања у опадајућем поретку.

3.2.2. Задаци

Означавају ентитете преко којих се дефинишу проблеми које студенти решавају на испитима синтезе. Сваки задатак креира се над једним од шаблона. Задатак се креира извршавањем одговарајуће акције шаблона. Над истим шаблоном могуће је креирати више задатака. Приликом креирања задатка неопходно је дефинисати: назив задатка, текстуални опис проблема и тачно решење проблема. Подаци којима је попуњена база података шаблона користе се као јавни скуп података испита синтезе. Тајни скуп података иницијално представља копију јавног. Могуће га је изменити након креирања задатка.

На посебној страни апликације професору је доступан табеларни приказ постојећих задатака. Табела има шест колона при чему првих пет имају информативну улогу, а последња управљачку. У зависности од броја записа које је неопходно приказати садржај табеле се приказује на једној или више страна табеле. Маскималан број записа који се приказује у оквиру једне стране табеле налази се је у нумеричком пољу изнад заглавља последње колоне. Подразумевана вредност је пет. Коришћењем стрелица корисник може да повећа или смањи ову вредност у корацима од по пет. Могуће је унети и произвољну вредност. На слици 3.2.2.1 приказан је табеларни преглед задатака.

<input type="text" value="Name"/>	<input type="text" value="Template name"/>	<div>5</div>			
Name	Template	Description	Author	Created on	
Pronađi kuvara	Restoran	-	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	20/07/2020 15:41	Actions
Preuzimanje knjiga	Библиотека	-	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	30/06/2020 15:20	Actions
Sumnjive transakcije	Računovoda	-	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	30/06/2020 15:17	Actions
Izvestaj finansija 1	Računovoda	-	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	30/06/2020 15:16	Actions
Izvestaj finansija 2	Računovoda	Sastaviti upit ko...	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	10/06/2020 12:34	Actions
					<div>Prev</div> <div>1</div> <div>2</div> <div>3</div> <div>Next</div>

Слика 3.2.2.1 Табеларни преглед задатака

Прва и трећа колона приказују назив задатка и текстуални опис проблема синтезе, респективно. Назив задатка мора имати најмање један а највише 50 карактера. Опис проблема нема ограничења у погледу броја карактера. У табеларном прегледу приказују се највише првих 20 карактера. Између ових колона налази се колона која приказује назив шаблона над којим је задатак креиран. Две претпоследње колоне имају исти формат као и код табеларног прегледа шаблона (слика 3.2.1.1) с тим да се у овом случају ради о датуму креирања и аутору задатка, а не шаблона. Последња колона не садржи никакав информативни садржај. Заглавље колоне је празно, а у сваком од редова налази се дугме приступ акцијама задатка.

Притиском на дугме у последњој колони кориснику се приказује прозор са списком акција које се могу извршити над задатком. Дизајн прозора одговара дизајну прозора приказаног на слици Слика 3.2.1.3. У овом случају прозор садржи акције за: промену назива задатка, дефинисање или измену описа задатка, измену тајног скупа података, дефинисање тачног решења задатка, креирање испита синтезе и брисање задатка. Измена тајног скупа података и дефинисање тачног решења задатка су акције које се извршавају на засебним странама апликације.

Прозори за измену назива и креирање испита синтезе су готово идентични прозору приказаном на слици 3.2.1.2. Једина функционална разлика је у случају промене назива где је текстуално поље иницијално попуњено тренутним назива задатка. Као и приликом креирања шаблона, нови назив задатка мора бити јединствен. У оба случаја постоје и мање текстуалне разлике: заглавље прозора, лабела изнад поља текстуалног уноса и лабела дугмета за потврду.

Прозор за дефинисање или измену текстуалног описа проблема синтезе је готово идентичан прозору приказаном на слици 3.2.1.4. Постоје мање текстуалне разлике: заглавље прозора и лабела дугмета за потврду. Прозор за брисање задатка је готово идентични прозору приказаном на слици 3.2.1.5. Да би операција брисања била успешно извршена не сме постојати ни један испит синтезе креиран над датим шаблоном.

Садржај табеле задатака могуће је претражити и сортирати. Претрага је могућа по називу задатка и називу шаблона. Сортирање је могуће по називу задатка, називу шаблона и датуму креирања задатка. Текстуална поља за претрагу налазе се изнад заглавља колона по којима се претражује, а сортирање се врши помоћу стрелица које се налазе уз називе колона.

3.2.3. Испит синтезе

Означавају ентитете помоћу којих се извршава провера синтетичког знања студената. Сваки испит синтезе креира се над једним од задатака синтезе. Испит се креира извршавањем одговарајуће акције задатка. Над истим задатком могуће је креирати више испита синтезе. Приликом креирања испита синтезе непходно је дефинисати назив испита и скуп студената који ће полагаати дати испит.

На посебној страни апликације професору је доступан табеларни приказ постојећих испита синтезе. Табела има седам колона при чему првих шест имају информативну улогу, а последња управљачку. У зависности од броја записа које је неопходно приказати садржај табеле се приказује на једној или више страна табеле. Маскималан број записа који се приказује у оквиру једне стране табеле налази се је у нумеричком пољу изнад заглавља последње колоне. Подразумевана вредност је пет. Коришћењем стрелица корисник може да повећа или смањи ову вредност у корацима од по пет. Могуће је унети и произвољну вредност. На слици 3.2.3.1 приказан је табеларни преглед испита синтезе.

Name

Task name

Template name

All

5

Name	Task	Template	Author	Created on	State	
Kolokvijum 1 grupa 1	Sumnjive transakcije	Računovođa	Стефан Ђорђевић ds173114m@student.etf.b g.ac.rs	30/06/2020 15:18	Scheduled	Actions
Kolokvijum 1 grupa 2	Izvestaj finansija 2	Računovođa	Стефан Ђорђевић ds173114m@student.etf.b g.ac.rs	30/06/2020 15:12	Scheduled	Actions
K2 popravni	Najkraći put	Mapa sveta	Стефан Ђорђевић ds173114m@student.etf.b g.ac.rs	23/06/2020 14:02	In Progress	Actions
Kolokvijum 2 grupa 1	Izvestaj finansija 2	Računovođa	Стефан Ђорђевић ds173114m@student.etf.b g.ac.rs	10/06/2020 18:50	Completed	Actions
Kolokvijum 2 grupa 2	Izvestaj finansija 2	Računovođa	Стефан Ђорђевић ds173114m@student.etf.b g.ac.rs	10/06/2020 18:49	Completed	Actions

Prev

1

2

3

Next

Слика 3.2.3.1 Табеларни преглед испита синтезе

Прве три колоне приказују назив испита синтезе, назив задатка над којим је креиран испит синтезе и назив шаблона над којим је креиран задатак, респективно. Назив испита синтезе мора имати најмање један а највише 50 карактера. У табеларном прегледу приказују се највише првих 20 карактера. Наредне две колоне имају исти формат као и код табеларног прегледа шаблона (слика 3.2.1.1) с тим да се у овом случају ради о датуму креирања и аутору испита синтезе, а не шаблона. Претпоследња колона приказује стање у ком се налази испит синтезе. Последња колона не садржи никакав информативни садржај. Заглавље колоне је празно, а у сваком од редова налази се дугме за приступ акцијама.

Притиском на дугме у последњој колони кориснику се приказује прозор са списком акција које се могу извршити над испитом синтезе. Дизајн прозора одговара дизајну прозора приказаног на слици 3.2.1.3. Садржај прозора односно акције које су на располагању зависе од стања у ком се налази испит синтезе. У сва три стања доступне су акције за промену назива и брисање испита. Акција за прелазак у наредно стање доступна је у свим стањима изузев завршног стања. Акција за доделу испита студентима доступна је искључиво када је испит заказан. Акција за присту резултатима доступна је искључиво завршном стању. Додељивање испита студентима и преглед резултата испита врши се на посебним странама апликације.

Прозор за промену назива испита синтезе је готово идентични прозору приказаном на слици 3.2.1.2. Једина функционална разлика је у текстуалном пољу које је иницијално попуњено тренутним називом испита синтезе. Као и приликом креирања шаблона, нови назив испита синтезе мора бити јединствен. Постоје и мање текстуалне разлике: заглавље прозора, лабела изнад поља текстуалног уноса и лабела дугмета за потврду.

Акција за прелазак у следеће стање омогућава промену стања испита синтезе. Промена стања подразумева прелазак из почетног стања („заказан“) у стање „у току“ или прелазак из стања „у току“ у завршно стање. Прозор за извршавање ове акције је готово идентични прозору приказаном на слици 3.2.1.5. Постоје мање текстуалне разлике: заглавље прозора и текст у главном делу прозора. Прозор за извршавање акције за брисање испита

синтезе је готово идентични прозору приказаном на слици 3.2.1.5. Да би операција брисања била успешно извршена испит синтезе не сме бити додељен ни једном студенту и мора бити у иницијалном стању.

Садржај табеле испита синтезе могуће је претражити и сортирати. Претрага је могућа по називу испита синтезе, називу задатака, називу шаблона и стању испита синтезе. Сортирање је могуће по називу испита синтезе, називу задатака, називу шаблона, стању испита синтезе и датуму креирања испита синтезе. Текстуална поља за претрагу налазе се изнад заглавља колона по којима се претражује, а сортирање се врши помоћу стрелица које се налазе уз називе колона. Изузетак је колона стања где се претрага врши одабиром вредности из падајуће листе уместо уноса у текстуалног поља.

3.2.4. Испит анализе

Означавају ентитете помоћу којих се извршава провера аналитичког знања студената. Сваки испит анализе креира се над одговором студента на неком од испита синтезе. Над истим одговором могуће је креирати више испита анализе. Приликом креирања испита анализе неопходно је дефинисати назив испита и скуп студената који ће полагати тај испит.

На посебној страни апликације професору је доступан табеларни приказ постојећих испита анализе. Табела има осам колона при чему првих седам имају информативну улогу, а последња управљачку. У зависности од броја записа које је неопходно приказати садржај табеле приказује се на једној или више страна табеле. Маскималан број записа који се приказује у оквиру једне стране табеле налази се је у нумеричком пољу изнад заглавља последње колоне. Подразумевана вредност је пет. Коришћењем стрелица корисник може да повећа или смањи ову вредност у корацима од по пет. Могуће је унети и произвољну вредност. На слици 3.2.4.1 приказан је табеларни преглед испита анализе.

Name

Task name

Template name

All

5

Name	Synthesis Test	Task	Template	Author	Created on	State	
Kolokvijum 3 grupa 1	st4 Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	Najkraći put	Mapa sveta	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	30/06/2020 16:28	Scheduled	Actions
Kolokvijum 3 grupa 2	st4 Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	Najkraći put	Mapa sveta	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	23/06/2020 17:35	InProgress	Actions
Kolokvijum 3 grupa 3	st4 Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	Najkraći put	Mapa sveta	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	23/06/2020 15:07	Scheduled	Actions
Kolokvijum 3 grupa 4	st4 Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	Najkraći put	Mapa sveta	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	23/06/2020 13:16	Completed	Actions
K3 popravni	st4 Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	Najkraći put	Mapa sveta	Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	29/05/2020 23:27	Completed	Actions

Prev

1

Next

Слика 3.2.4.1 Табеларни преглед испита анализе

Прва, друга и четврта колона приказују назив испита анализе, назив задатка синтезе и назив шаблона над којим је креиран задатак, респективно. Назив испита анализе мора имати

најмање један а највише 50 карактера. У табеларном прегледу приказују се највише првих 20 карактера. Друга колона садржи назив испита синтезе над чијим одговором је креиран испит анализе. У истој колони налази се име, презиме и корисничко име студента који је предао дати одговор. Пета и шеста колона имају исти формат као и код табеларног прегледа шаблона (слика 3.1.1.1) с тим да се у овом случају ради о датуму креирања и аутору испита анализе, а не шаблона. Претпоследња колона приказује стање у ком се налази испит анализе. Последња колона не садржи никакав информативни садржај. Заглавље колоне је празно, а у сваком од редова налази се дугме за приступ акцијама.

Притиском на дугме у последњој колони кориснику се приказује прозор са списком акција које се могу извршити над испитом анализе. Дизајн прозора одговара дизајну прозора приказаног на слици Слика 3.2.1.3. Садржај прозора односно акције које су на располагању зависе од стања у ком се налази испит анализе. У сва три стања доступне су акције за промену назива и брисање испита. Акција за прелазак у наредно стање доступна је у свим стањима изузев завршног стања. Акција за доделу испита студентима доступна је искључиво када је испит заказан. Акција за присту резултатима доступна је искључиво завршном стању. Додељивање испита студентима и преглед резултата испита врши се на посебним странама апликације.

Прозор за промену назива испита анализе је готово идентични прозору приказаном на слици 3.2.1.2. Једина функционална разлика је у текстуалном пољу које је иницијално попуњено тренутним називом испита анализе. Као и приликом креирања шаблона, ново име испита анализе мора бити јединствено. Постоје и мање текстуалне разлике: заглавље прозора, лабела изнад поља текстуалног уноса и лабела дугмета за потврду.

Акција за прелазак у следеће стање омогућава промену стања испита анализе. Промена стања подразумева прелазак из почетног стања („заказан“) у стање „у току“ или прелазак из стања „у току“ у завршно стање. Прозор за извршавање ове акције је готово идентични прозору приказаном на слици 3.2.1.5. Постоје мање текстуалне разлике: заглавље прозора и текст у главном делу прозора. Прозор за извршавање акције за брисање испита анализе је готово идентични прозору приказаном на слици 3.2.1.5. Да би операција брисања била успешно извршена испит анализе не сме бити додељен ни једном студенту и мора бити у иницијалном стању.

Садржај табеле испита анализе могуће је претражити и сортирати. Претрага је могућа по називу испита анализе, називу задатака, називу шаблона и стању испита анализе. Сортирање је могуће по називу испита анализе, називу задатака, називу шаблона, стању испита анализе и датуму креирања испита анализе. Текстуална поља за претрагу налазе се изнад заглавља колоне по којима се претражује, а сортирање се врши помоћу стрелица које се налазе уз називе колоне. Изузетак је колона стања где се претрага врши одабиром вредности из падајуће листе уместо уноса у текстуалног поља.

3.2.5. Моделирање базе података

Страница за дефинисање и измену модела базе података користи се приликом дефинисања шаблона. Одабиром акције за дефинисање (и измену) модела на неком од шаблона корисник бива преусмерен на ову страницу. Садржај странице приказан је на слици 3.2.5.1.

```

1 create table tabela_2
2 (
3   k1 int primary key,
4   k2 nvarchar(50) default 'N/A',
5   k3 float not null
6 )
7 create table tabela_3
8 (
9   k1 int primary key,
10  k2 nvarchar(50) default 'N/A',
11  k3 float not null
12 )
13 create table tabela_4
14 (
15  k1 int primary key

```

Execute

Choose a table from the list:

dbo.tabela_1

columns:

- k1:
 - type: "int"
 - isNullable: false
 - defaultValue: null
 - maxLength: null
- k2:
 - type: "nvarchar"
 - isNullable: true
 - defaultValue: "('N/A')"
 - maxLength: 50

Слика 3.2.5.1 Дефинисање модела базе података

У доњем делу странице налази се падајућа листа чији су елементи представљају постојеће табеле унутар базе података. Вредност сваког од елемената састоји се од назива схеме табеле и назива саме табеле одвојених тачком. Испод падајуће листе налази се компонента за приказ структуре табеле. Компонента приказује структуру табеле која је одабрана у падајућој листи. Компонента има структуру стабла. Чворови првог нивоа представљају називе колона. Сваки од ових чворова има четири потомка који приказују: тип колоне, да ли колона може бити без вредности, подразумевану вредност и максималну дужину података. Потомке чворова првог нивоа могуће је сакрити притиском на стрелицу уз назив колоне. Притиском на стрелицу уз корени чвор могуће је сакрити комплетан садржај ове компоненте. Сакривање делова компоненте за циљ има побољшање прегледности у ситуацијама када табела има велики број колона.

У горњем делу странице налази се текстуално поље за унос *SQL* упита. Одмах испод текстуалног поља налази се дугме за извршавање унетог упита. Замисао је да се на овај начин дефинишу табеле, релације међу табелама и ограничења (на нивоу колоне, табеле или базе података). Могуће је и извршавати упите који врше промену података с тим да је ово знатно лакше остварити коришћењем посебне странице за измену података. Након успешног извршавања упита ажурира се садржај падајуће листе. У случају да је приликом извршавања упита дошло до грешке порука сервера базе података се приказује у прозору обавештења претраживача.

3.2.6. Попуњавање садржаја базе података

Страница за попуњавање садржаја базе података користи се приликом дефинисања садржаја базе података шаблона и базе података задатка. Одабиром одговарајуће на неком од

шаблона или задатака корисник бива преусмерен на ову страницу. Садржај странице приказан је на слици 3.2.6.1.

Choose a table from the list:

dbo.tabela_2

	kol1	kol2
-	1	1
-	2	5
-	3	3
-	4	19
+		-5

Слика 3.2.6.1 Попуњавање садржаја базе података

У горњем делу странице налази се падајућа листа чији су елементи представљају постојеће табеле унутар базе података. Вредност сваког од елемената састоји се од назива схеме табеле и назива саме табеле одвојених тачком. Испод падајуће налази се компонента за приказ и измену садржаја једне табеле унутар базе података. Компонента се односи на табелу која је тренутно одабрана у падајућој листи. Променом одабраног елемента падајуће листе ажурира се компонента за приказ и измену садржаја табеле. Заглавље прве колоне је празно. Заглавља преосталих колона садрже називе колона табеле у бази података. Измена садржаја табеле подразумева три операције: додавање записа, измена постојећих записа и брисање појединачних записа из табеле.


За додавање новог записа користи се последњи ред компоненте. У првој колони налази се дугме за потврду уноса новог записа. У преосталим колонама налазе се поља за текстуални унос. Ова поља иницијално немају вредност и служе како би корисник унео вредности ћелија новог записа табеле. Притиском на дугме за потврду уноса у табелу базе података се уноси нови запис и ажурира се садржај компоненте. Приликом дефинисања вредности новог записа корисник не мора унети вредности за колоне које у табели базе података имају дефинисану подразумевану вредност. Колоне чија се вредност аутоматски генерише од стране сервера базе података садрже текстуална поља којима је онемогућена измена садржаја. На слици 3.2.6.1 приказан је унос новог записа који за колону „kol2“ дефинише вредност „-5“ док се вредност колоне „kol1“ генерише аутоматски од стране сервера базе података.


Средишњи редови компоненте (између последњег реда и заглавља) представљају тренутне записе у табели базе података. У првој колони налази се дугме за брисање записа. Притиском на ово дугме брише се одговарајући запис из базе података и компонента се ажурира. На слици 3.2.6.1 приказана је садржај табеле која тренутно има четири записа.

Измена постојећих записа врши се изменом вредности појединачних ћелија. На кориснику је да промени садржај одговарајућег поља за текстуални унос. У овој ситуацији нема засебног дугмета за потврду измене већ се измена у бази врши након што текстуално поље изгуби фокус. Након извршене измене компонента се ажурира.

3.2.7. Додељивање испита студентима

Страница за додељивање испита студентима користи се како би се остварила веза између једног испита и једног или више студената. Додељивање студената испиту синтезе врши се на исти начин као и додељивање студената испиту анализе. Одабиром акције за додељивање испита студентима на неком од испита синтезе или испита анализе корисник бива преусмерен на ову страницу. Садржај странице приказан је на слици 3.2.7.1.

 Find students on Azure

10 

First name	Last name	Email	Marked
Александар	Аћимовић	aa060148d@student.etf.bg.ac.rs	<input type="checkbox"/>
Aleksandar	Arsenović	aa070267d@student.etf.bg.ac.rs	<input type="checkbox"/>
Aleksandar	Aleksić	aa073253m@student.etf.bg.ac.rs	<input type="checkbox"/>
Ana	Anastasijević	aa080123d@student.etf.bg.ac.rs	<input checked="" type="checkbox"/>
Александар	Абу Сampa	aa080252d@student.etf.bg.ac.rs	<input checked="" type="checkbox"/>
Алексеј	Аврамовић	aa085020p@student.etf.bg.ac.rs	<input type="checkbox"/>
Aleksandar	Arandelović	aa090433d@student.etf.bg.ac.rs	<input checked="" type="checkbox"/>
Aleksandar	Adašević	aa093200m@student.etf.bg.ac.rs	<input type="checkbox"/>
Aleksandar	Antanasković	aa093209m@student.etf.bg.ac.rs	<input type="checkbox"/>
Антонина	Алексић	aa110035d@student.etf.bg.ac.rs	<input type="checkbox"/>

1

2

3

4

5

6

7

8

9

10

Assigned students (3)

First name	Last name	Email	
Aleksandar	Antanasković	aa093209m@student.etf.bg.ac.rs	<input type="button" value="Remove"/>
Стефан	Ђорђевић	ds173114m@student.etf.bg.ac.rs	<input type="button" value="Remove"/>
Aleksandar	Arsenović	aa070267d@student.etf.bg.ac.rs	<input type="button" value="Remove"/>

Слика 3.2.7.1 Додељивање испита студентима

Садржај странице приказане на слици 3.2.7.1. подељен је хоризонталном линијом на две логичке целине. Горњи део странице омогућава претрагу студената и додељивање испита студентима. Доњи део странице омогућава преглед студената којима је испит додељен и раскидање везе између студента и испита.

У горњем делу странице дат је табеларни приказ свих корисника. Табела има четири колоне. Прве три колоне имају информативну улогу - приказују име, презиме и корисничко име студента, респективно. Последња колона садржи поља за потврду. Информације о студентима којима је испит већ додељен у овој табели приказане су сивом бојом, а поље за потврду је неозначено и није могуће променити његову вредност. Због великог броја записа садржај табеле се приказује у неколико страница табеле. Кориснички интерфејс испод табеле, са десне стране омогућава навигацију страницама табеле. Претрага је могућа на основу имена, презимена и корисничког имена студента. Испод табеле, са леве стране, налази се дугме за додељивање студената испиту. Приликом додељивања најпре се пољима за потврду означе жељени студенти. Затим се притиском на поменуто дугме извршава додела означених студената. Могуће је одабрати студенте са различитих страна табеле. Како би се смањила могућност грешке број одабраних студената приказује се као део лабеле дугмета за доделу. Лабела се ажурира променом вредности неког од поља за потврду.

Треба напоменути да иницијално нису приказане све странице табеле већ се након сваке отворене странице открива да ли постоји следећа. У примеру са слике 3.1.7.1 тренутно је отворена страница девет, а укупан број страница може бити десет или више. Отварањем

странице десет појавиће се дугме за приступ страници 11 (под условом да страница 11 постоји). У табелама описаним раније у овом поглављу (шаблони, задаци и испити) иницијално су приказане све странице и могуће је отворити било коју страницу било којим редоследом. Приликом претраге вредност унета у поље за претрагу сматра се почетком тражене вредности. Ово се такође разликује од начина претраге у табелама описаним раније у овом поглављу где се унета вредност сматра делом тражене вредности. Иако поменута ограничења не спречавају корисника да обави додељивање испита одговарајућим студентима јасно је да у одређеној мери умањују једноставност приступа подацима. Разлози због којих ова ограничења није могуће избећи су техничке природе и биће додатно појашњени у одељку 4.3.











Доњи део странице студенте такође приказује у табеларној форми. Имајући у виду да је број студената додељених испиту релативно мали у поређењу са укупним бројем корисника ова табела не пружа могућност претраге и целокупни садржај приказује на једној страни. Сама табела је доста слична табели у горњем делу странице - прве три колоне на истовестан начин приказују име, презиме и корисничко име студента. Последња колона садржи дугме којим се раскида веза између испита и студента.

Треба уочити да садржаји двеју табела нису међусобно независни. Табела у горњем делу странице приказује и студенте којима је испит додељен и оне којима није. Табела у доњем делу странице искључиво приказује студенте којима је испит додељен. Из тог разлога се садржаји обе табеле ажурирају истовремено и то након додељивања испита студентима и након раскидања везе између испита и студента.

3.2.8. Резултати испита

Странице за приказ резултата испита користе се ради прегледа резултата оцењених испита и покретања процеса евалуације код неоцењених. Одабиром акције за приказ резултата на неком од испита корисник бива преусмерен на ову страницу. Садржај странице чине дугме за покретање процеса евалуације и табела. Табела садржи по један запис за сваког студента коме је испит додељен. У зависности од тога да ли се ради о резултатима испита синтезе или анализе разликује се и садржај поменуте табеле. На слици 3.2.8.1 приказан је преглед резултата испита синтезе у моменту када је евалуација резултата у току.

[Start evaluation](#)

Student	Public Data Set	Secret Data Set	Submitted paper	Analysis test
Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs			View	Create
Петар Петровић pp113121m@student.etf.bg.ac.rs		 Failed	View	Create
Ana Ančić aa183321m@student.etf.bg.ac.rs			View	Create
Miloš Milošević mm205221m@student.etf.bg.ac.rs			View	Create
Дуња Дуњић dd192553m@student.etf.bg.ac.rs			View	Create

Слика 3.2.8.1 Евалуација испита синтезе

Табела за преглед резултата испита синтезе садржи пет колона. Прве три колоне имају информативну, а последње две управљачку улогу. У првој колони приказано је име, презиме и корисничко име студента. Наредне две колоне садрже резултат процеса евалуације испита синтезе и анализе, респективно. Уколико је процес евалуације у току ове колоне садрже информацију о тренутној фази евалуације. Следећа колона садржи дугме за преглед студентовог решења. Притиском на ово дугме отвара се страница за полагање испита синтезе са подацима за одабраног студента. Када овој страници приступа професор онемогућено је мењање самог одговора студента. Садржај ове странице објашњен је у одељку 3.3.4. Последња колона садржи дугме за креирање испита анализе. Притиском на ово дугме отвара се прозор за дијалог сличан прозору приказаном на слици 3.2.1.2. Након уношења назива испита анализе прозор се затвара и корисник бива преусмере на страницу за табеларни приказ испита анализе.

Фазе евалуације приказане су симболично. Посматрајући слику 3.2.8.1 одозго, први корисник није полагао испит. Рад другог корисника је оцењен. Евалуација са јавним подацима дала је тачан резултат, док је евалуација са тајним подацима дала нетачан резултат. Код трећег корисника евалуација са јавним подацима је у току, а са тајним у реду за чекање. Евалуација са тајним подацима последњег корисника још није започета и треба да уђе у ред за чекање. Постављањем показивача преко било ког симбола приказује се његово значење у текстуалној форми. Фазе евалуације су исте како код испита синтезе тако и код испита анализе.

Када ради о испиту анализе табела садржи шест колона. Првих пет колона имају информативну, а последње две управљачку улогу. У првој колони приказано је име, презиме и корисничко име студента. У другој колони налази се информација о фази извршавања процеса којим се генеришу табеле Т3 и Т4 са слике 2.5.3.1. Наредне три колоне садрже информацију о фази извршавања процеса којим се генеришу компоненте оцене К1, К2 и К3 са слике 2.6.3.1. Следећа колона садржи дугме за преглед студентовог решења. Притиском на ово дугме отвара се страница за полагање испита анализе са подацима за одабраног студента. Када овој страници приступа професор онемогућено је мењање одговора студента. На слици 3.2.8.2 приказан је преглед резултата испита анализе у моменту када је евалуација резултата у току.

[Start evaluation](#)

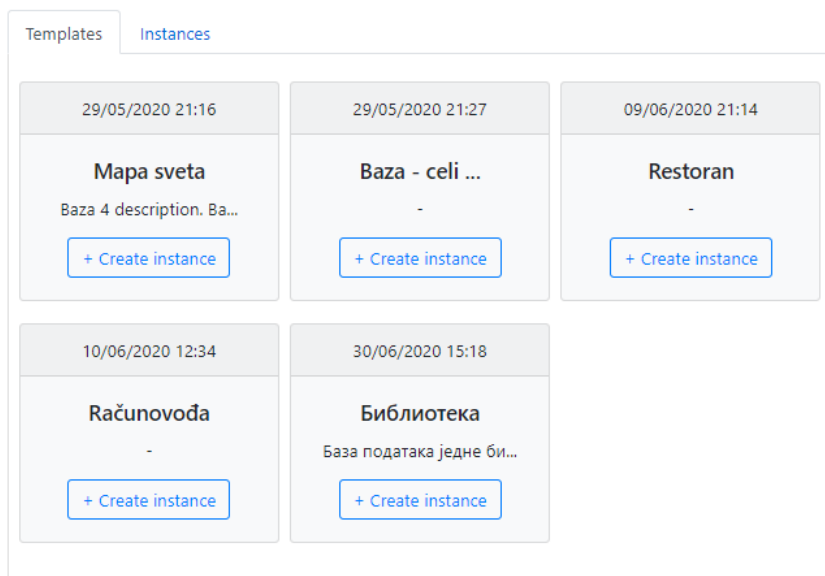
Student	Prepare Outputs	Failing Input	Query Output	Correct Output	Submitted paper
Стефан Ђорђевић ds173114m@student.etf.bg.ac.rs	✔	✘	✘	✔	View
Petar Petrović pp183214m@student.etf.bg.ac.rs	✘	✘	✘	✘	View
Adam Adamović pp183214m@student.etf.bg.ac.rs	✔	✔	✘	✘	View
Milan Milanović ds185522m@student.etf.bg.ac.rs	✔	🔄	🕒	🕒	View
Jovana Jovanović jj1832664m@student.etf.bg.ac.rs	🔄	○	○	○	View

Слика 3.2.8.2 Евалуација испита анализе

3.3. Примери употребе апликације од стране студента

3.3.1 Преглед вежби

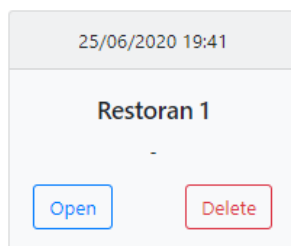
Студенти приступају процесу усавршавања знања преко странице за вежбе. Садржај ове странице подељен је у две картице. У првој су приказани јавни шаблони креирани од стране професора. У другој картици приказане су инстанце база података које је креирао тренутно аутентикован студент на основу једног од шаблона. Картица са инстанцама је подразумевано активна. На слици 3.3.1.1 приказан је садржај странице за вежбе у случају када је активна картица са шаблонима.



Слика 3.3.1.1 Страница за вежбе - шаблони

Сваки шаблон у картици са шаблонима приказан је засебним елементом. У заглављу елемента налази се датум и време креирања шаблона. У главном делу налази се назив шаблона, опис модела базе података и дугме за креирање инстанце. Приказује се првих 15 карактера назива шаблона и првих 25 карактера описа. Корисник може видети целокупан садржај постављањем показивача преко наслова или описа. Притиском на дугме за креирање инстанце отвара се прозор сличан прозору приказаном на слици 3.2.1.2. Корисник уноси назив инстанце и потврђује њено креирање. Назив инстанце мора бити јединствен на нивоу корисника. На основу једног шаблона могуће је креирати једну или више инстанци.

Организација података у картици са инстанцама је јако слична оној у картици са шаблонима. Разликују се појединачни елементи којима су представљене инстанце. На слици 3.3.1.2 приказан је елемент који представља једну инстанцу.



Слика 3.3.1.2 Елемент прегледа инстанци

Датум и време креирања као и назив сада се односе на инстанцу. Дугме за креирање инстанце замењено је паром дугмади за приступ инстанци и њено брисање. Притиском на дугме за приступ инстанци корисник бива преусмерен на посебну страну. Притиском на дугме за брисање отвара се прозор за потврду брисања инстанце. Прозор је сличан прозору за потврду брисања шаблона приказаном на слици 3.2.1.5.

3.3.2 Преглед испита

Студенти приступају процесу евалуације знања преко стране за испите. Садржај ове стране подељен је у три картице. У првој су приказани испити који су заказани, у другој испити који су у току и у трећој испити који су завршени. Картица са испитима у току је подразумевано активна. На слици 3.3.2.1 дати су елементи којима се приказује заказан испит синтезе, испит анализе у току, пропуштен испит синтезе и оцењен испит анализе, респективно.

22/05/2020 17:00	22/05/2020 17:00	Missed	Completed
Kolokvijum 1 grupa 2	Kolokvijum 3 grupa 2	Kolokvijum 1 2020	K3 popravni
Synthesis	Analysis		
Take test	Take test	Public Secret	Input Query Student
		✕ ✕	✓ ✓ ✕

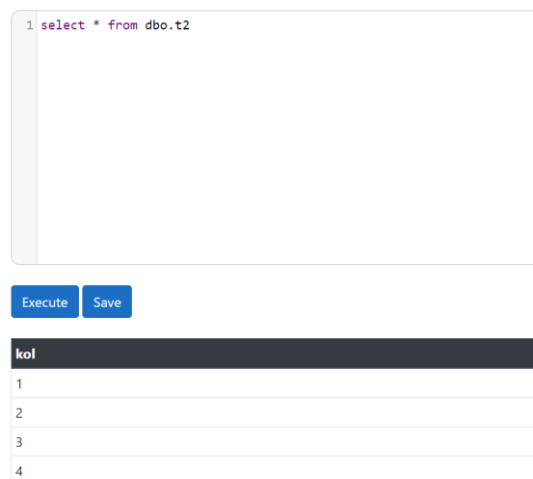
Слика 3.3.2.1 Елементи прегледа испита

Заказани испити и испити у току у свом заглављу имају датум и време креирања. У главном делу налази се назив испита, врста испита и дугме за полагање испита. Дугме за полагање је онемогућено код заказаних испита. Притиском на дугме за полагање испита који је у току корисник бива преусмерен на одговарајућу страну за полагање испита.

Завршени испити у свом заглављу имају информацију о томе да ли је студент изашао на испит или га је пропустио. У главном делу налази се назив испита и табела која у једном реду приказује појединачне компоненте оцено. Број колона у табели зависи од броја компоненти оцено односно од врсте испита.

3.3.3 Инстанца

Садржај стране за рад са инстанцом подељен је у три картице. У првој се налази текстуални опис модела базе. У другој се налази кориснички интерфејс за извршавање упита над инстанцом. Садржај ове картице приказан је на слици 3.3.3.1. У горњој половини налази се кориснички интерфејс за уношење и извршавање упита. Поред дугмета за извршавање налази се додатно дугме за чување унетог упита. У доњој половини налази се табела која представља резултат извршеног упита. Структура и садржај табеле ажурира се након сваког извршавања упита. У последњој картици налази се кориснички интерфејс за измену података у инстанци, а начин његове употребе описан је у одељку 3.2.6. Картица за извршавање упита је подразумевано активна.



Слика 3.3.3.1 Извршавање упита над инстанцом

3.3.4 Испит синтезе

Садржај странице за полагање испита синтезе подељен је у три картице. Прва картица садржи текстуални опис модела базе и задатка. Следећа картица садржи кориснички интерфејс за преглед садржаја базе података. Сличан је интерфејсу за попуњавање базе описаном у одељку 3.2.6. Онемогућено је додавање или измена постојећих записа – последњи ред и прва колона са слике 3.2.6.1 су избачене, а измена текстуалних поља онемогућена. Последња картица садржи кориснички интерфејс за извршавање упита и чување одговора. Сличан је интерфејсу приказаном на слици 3.3.3.1 с тим да није могуће предати упит уколико формат његовог излаза не одговара формату траженог резултата.

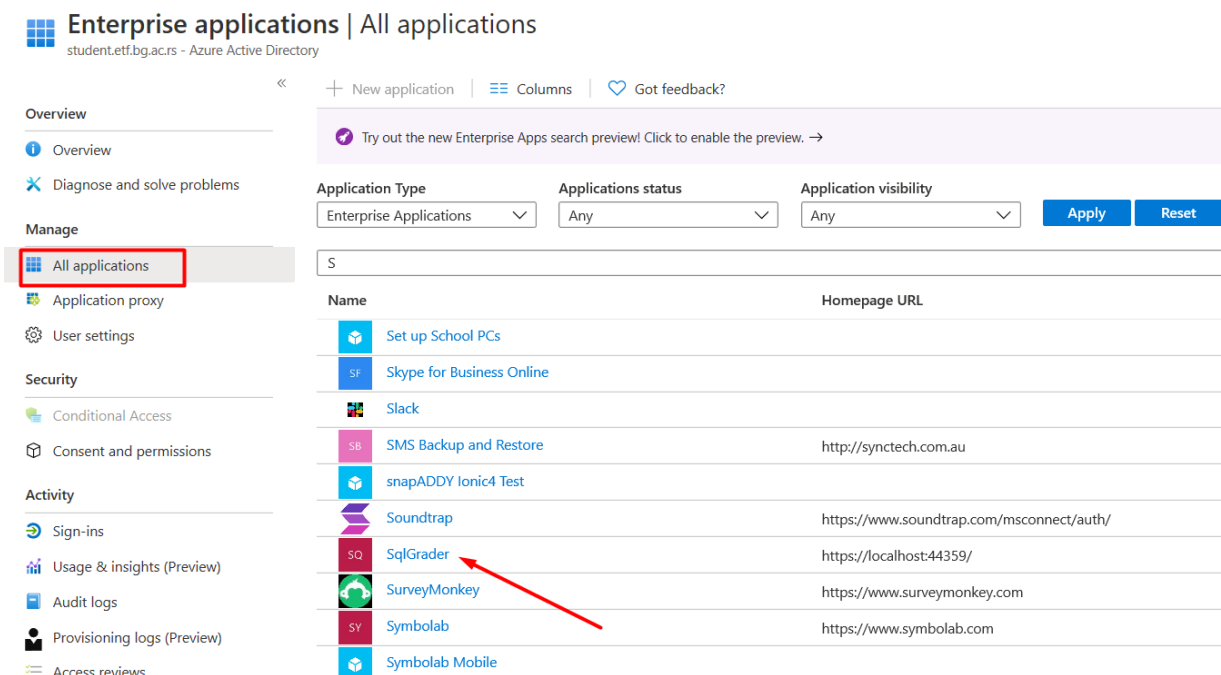
3.3.5 Испит анализе

Садржај странице за полагање испита анализе подељен је у четири картице. Прва картица садржи текстуални опис модела базе, задатка синтезе и упит који представља предмет анализе. Следећа картица садржи кориснички интерфејс за попуњавање садржаја базе података. Идентичан је интерфејсу приказаном на слици 3.2.6.1. Овим студент задаје попуњавање за које предмет анализе треба врати нетачан резултат. Последње две картице садрже исти кориснички интерфејс с тим да не постоји падајућа листа за одабир табеле. У претпоследњој картици студент дефинише садржај табеле који треба да одговара излазу предмета анализе над задатим попуњавањем базе података. У последњој картици студент дефинише садржај табеле који треба да одговара излазу тачног решења задатка синтезе над задатим попуњавањем базе података.

3.4. Примери употребе апликације од стране администратора

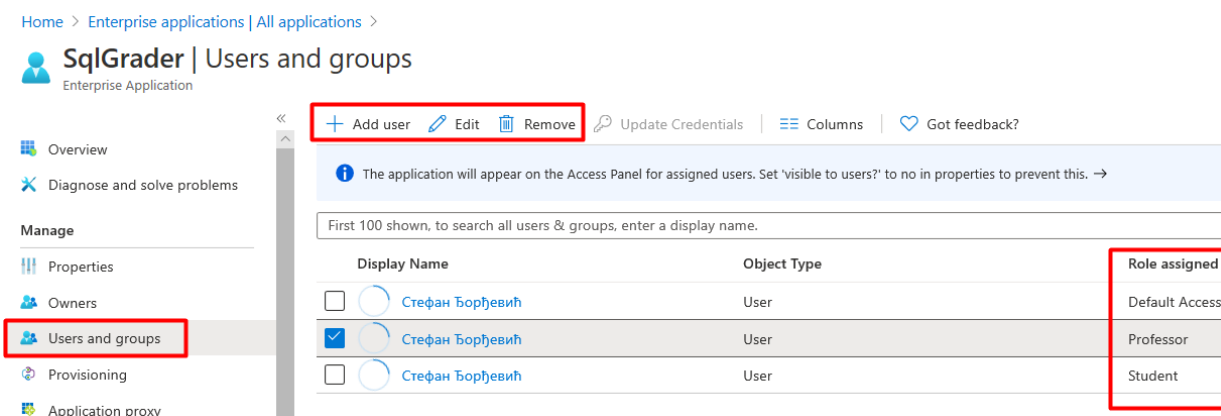
Управљање корисницима апликације и њиховим улогама врши се преко Мајкрософтове интернет апликације *Azure Portal* [11]. Са становишта Мајкрософта власник апликације је тај који управљање њеним корисницима. Власник апликације је онај корисник који је регистровао апликацију. Са становишта апликације која је предмет овог рада исти корисник сматра се администратором система. Најпре је потребно је приступити апликацији *Azure Portal* евентуалну аутентикацију. Затим је потребно приступити сервису *Enterprise*

applications. Унутар секције *All applications* приказане на слици 3.4.1 доступне су све апликације регистроване на домену међу којима је и апликација која је предмет овог рада.



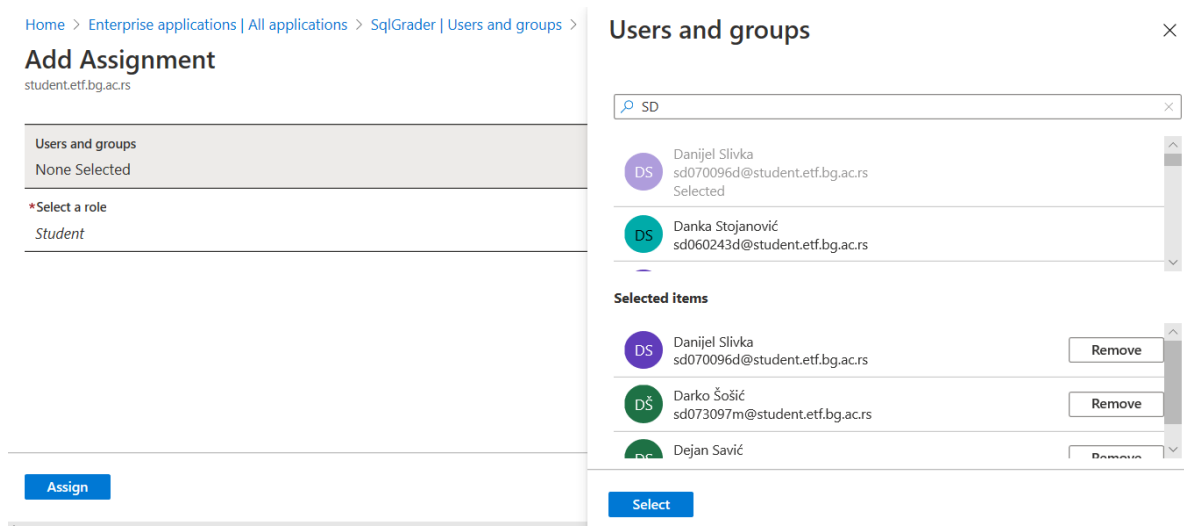
Слика 3.4.1 Секција *All applications* апликације *Azure Portal* [11]

Након приступа апликацији неопходно је приступити секцији *Users and groups* приказаној на слици 3.4.2. Унутар ове секције приказани су сви корисници апликације и њихове улоге са становишта регистроване апликације. Постојеће кориснике могуће је претраживати, уклањати или им мењати улоге. Могуће је и додавати нове кориснике.



Слика 3.4.2 Секција *Users and groups* апликације регистроване на AAD домену [11]

Приликом додавања нових корисника неопходно је одабрати један или више Мајкрософт налога и доделити им једну од улога које дефинише регистрована апликација. На слици 3.4.3 приказано је додељивање неколико студената апликацији.



Слика 3.4.3 Додељивање улога корисницима кроз апликацију *Azure Portal* [11]

4. ТЕХНИЧКИ АСПЕКТИ ИМПЛЕМЕНТАЦИЈЕ

У овом поглављу ће бити изнети технички аспекти имплементације апликације која је предмет овог рада. Биће описане стандардне технологије које су коришћене приликом имплементације. Такође биће наведени проблеми специфични за ову апликацију као и њихова решења. Након упознавања са садржајем овог поглавља читалац који познаје наведене технологије ће бити у стању да врши измене и надограђује постојећу имплементацију.

4.1. Технологије и алати

У овом одељку најпре ће бити изложене технологије које се односе на клијентски део кода. Затим ће бити изложене технологије које се тичу серверског дела кода и складиштења података. На крају ће бити изложен кратак преглед алата који су коришћени приликом израде апликације.

4.1.1. Клијентски део апликације

Под клијентским делом апликације подразумева се изворни код који се извршава у интернет претраживачу корисника. Улога овог дела апликације је приказ и управљање графичким корисничким интерфејсом као и размена података са серверским делом апликације. Крајњи код који се извршава у претраживачу је комбинација кода написаног у језицима *HTML* (*Hypertext Markup Language*), *CSS* (*Cascading Style Sheets*) и *JS* (*Java Script*). *HTML* и *CSS* спадају у скриптне језике и претраживач их обрађује приликом учитавања интернет странице. *HTML* је стандардни језик за приказивање графичког корисничког интерфејса у интернет претраживачима. Основни чиниоци кода су *HTML* елементи. Свака страна има три основна елемента (заглавље, тело и подножје). Параметри елемената називају се атрибути. Елементи имају структуру стабла. Један елемент садржи један или више других. *CSS* код описује изглед *HTML* елемената који се приказују. Исти изворни код може се примењивати на више *HTML* страна чиме се обезбеђује униформан дизајн корисничког интерфејса. *Java Script* представља програмски језик који се претежно користи за извршавање у интернет претраживачима. Има и ширу примену па се тако може користити за програмирање серверског кода (*Node.js*), а користе га и поједине базе података (*MongoDB* и *Apache CouchDB*). Извршавање кода може бити покренуто учитавањем странице, интеракцијом корисника са корисничким интерфејсом или одређеним тренутком у времену.

У комбинацији са изворним *JS* кодом коришћена је библиотека *jQuery* [12]. Ово је једна од најраспрострањенијих *JS* библиотека. Према подацима из јуна 2020. године преко 77.8% интернет сајтова користе *Java Script* при чему 75.8% свих интернет сајтова користе библиотеку *jQuery* [13]. Дизајниран је да олакша манипулацију *HTML* елементима, обраду догађаја, управљање *CSS* анимацијама и извршавање *AJAX* (*Asynchronous JavaScript and XML*) позива. Приликом израде ове апликације коришћена је верзија 3.3.1. библиотеке *jQuery*.

Велики део клијентског дела апликације ослања се на библиотеку *Bootstrap* [13]. Ова библиотека обједињује *HTML*, *CSS* и *JS* изворни код у унапред припремљене компоненте

корисничког интерфејса. Библиотеку одликује уграђена скалабилност која олакшава приказивање корисничког интерфејса на екранима различитих величина. Приликом израде ове апликације коришћена је верзија 4.3.1. библиотеке *Bootstrap*.

Једна од ретких компоненти корисничког интерфејса која није део поменутих библиотека је компонента за извршавање *SQL* упита. Представља део библиотеке *CodeMirror* [15]. Појединачне функционалности имплементирани су засебним датотекама па није потребно учитавање целе библиотеке. Компонента подржава различите програмске језике као и велики број опција за конфигурисање. Када се ради о *SQL* језику могуће је дефинисати називе табела и колона. Те информације библиотека користи како би побољшала аутоматско довршавање уноса. Приликом израде ове апликације коришћена је верзија 3.3.1. библиотеке *CodeMirror*.

4.1.2. Серверски део апликације

Под серверским делом апликације подразумева се изворни код који се извршава на серверу на ком је апликација смештена. У случају ове апликације код се не извршава директно над оперативним системом сервера већ на радном окружењу *.NET Core* [16]. Ово окружење је бесплатно и прати принципе отвореног кода. Највећим делом је развијено од стране Мајкрософта. Одликује га платформска независност па је могуће извршавање на *Windows*, *Linux* или *macOS* оперативним системима. Његов претходник, окружење *.NET Framework*, било је резервисано искључиво за *Windows* оперативне системе. Окружење подржава извршавање кода написаног у програмским језицима *C#*, *F#* и *VB.NET*. Серверски код ове апликације написан је у верзији 8 језика *C#*. Циљно окружење је *.NET Core* верзије 3.1.

4.1.3. Складиштење података

За већину интернет апликација неопходно је складиштити податке који су настали током извршавање. Најчешће је неопходна нека врста перзистентног складиштења где не долази до губитка података између различитих покретања апликације. Један од најраспрострањенијих видова оваквог складиштења јесу управо релационе базе података које су уједно и предмет саме апликације. За сервер базе података одабран је Мајкрософтов *SQL Server*.

За дефинисање модела и интеракцију са својом базом података апликација користи Мајкрософтову библиотеку *EF (Entity Framework)* [17]. Ова библиотека је део радног окружења *.NET Core*. Представља објектно-релациони мапер којим се *C#* класе пресликавају у табеле релационе базе података и обрнуто. У овој апликацији коришћен је *Code First* приступ где се на основу дефинисаних класа генерише база података. Класе односно ентитети који дефинишу модел се командом *add-migration* радног окружења преводе у *EF* миграције. Миграције у овом контексту представљају датотеке *C#* изворног кода које описују измене над моделом релационе базе. Саме миграције могуће је мењати након њиховог генерисања па чак уметати и делове *SQL* изворног кода (у виду ниски). На овај начин могуће је вршити и миграције које управљају самим подацима унутар базе. Командом *update-database* миграције се преводе у *SQL* скрипте и извршавају над базом података. Свака миграција извршава се тачно једном. Да би ово било обезбеђено *EF* генерише додатну табелу пода називом *__EFMigrationsHistory* у којој се чува историјат извршених миграција. Преко инстанци ентитета могуће је читати или мењати садржај базе директно из *C#* кода. Рад са ентитетима додатно олакшава употреба *LINQ (Language Integrated Query)* [18] синтаксе програмског језика *C#*. Њеном употребом практично се губи разлика између рада са класама

које представљају табеле релационе базе и класа које представљају објекте у радној меморији.

4.1.4. Алати

Основни алат који је коришћен за писање и тестирање изворног кода је Мајкрософтово развојно окружење *Visual Studio 2019* [19]. Коришћена је верзија 16.6.4 у издању *Professional*. За управљање изворним кодом коришћена је интернет платформа за верзионисање *GitHub* [19]. Ова платформа се најчешће се користи када постоји потреба да више аутора ради на истом изворном коду. Чак и када на изворном коду ради један аутор платформа доноси значајне предности као што је чување резервне копије и историјата свих измена. Изворни код апликације која је предмет овог рада доступан је на посебном репозиторијуму „Master-Rad” [21] у оквиру платформе *GitHub*. Током развоја кода са платформом је могуће комуницирати преко команде линије или користећи бројне имплементације графичког корисничког интерфејса. Једна таква имплементација уграђена је у развојно окружење *Visual Studio*. За управљање сервером база података коришћена је Мајкросфтова апликација *SQL Server Management Studio* верзије 17.9.1 [22]. За израду дијаграма приказаних на сликама 2.4.3.1, 2.5.3.1 и 4.5.1 коришћена је интернет апликација *Lucidchart* [23]. Коначно, за израду овог документа коришћена је Мајкросфтова апликација *Word* [24] верзије 1908.

4.2. Клијент-сервер комуникација

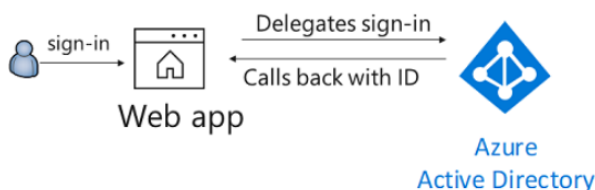
Основна поставка решења прати *MVC (Model View Controller)* шаблон. Улазна тачка су методе класа које представљају контролере. Ове методе се још називају и акције контролера. Контролери прихватају захтеве од интернет претраживача. Затим се извршава одређени део логике апликације који најчешће подразумева обраду података, интеракцију са базом података или комуникацију са екстерним сервисом. Резултат рада логике је модел података које треба приказати на страници апликације. Модел се прослеђује датотекама које представљају *View* компоненту *MVC* шаблона. Ове датотеке имају екстензију *cshtml* и у њима се налази комбинација *C#* и *HTML* кода. *HTML* типично садржи референце на друге датотеке у којима се налази *CSS* и *JS* код. Извршавањем *C#* кода генерише одговарајући *HTML*, *CSS* и *JS* кодом. Овакав садржај враћа се претраживачу и представља једну интернет страницу апликације. Треба напоменути да се *CSS* и *JavaScript* код не прослеђује у изворном облику. Код се обједињује у две датотеке (једна за *CSS*, а друга за *JavaScript*). Име датотеке аутоматски се сугерише претраживачу о којој верзији датотеке се ради односно да ли претраживач треба да инвалидира део своје кеш меморије. Обједињени код се додатно компримује како би се смањила његова величина и тиме убрзао пренос преко мреже.

У зависности од захтева претраживача варира и количина података у моделу. У ситуацијама као што је табеларни приказ великог броја записа може се десити да учитавање странице траје дуго или да до њега уопште не дође. Решење је у асинхроним позивима ка серверу коришћењем *AJAX (Asynchronous JavaScript and XML)* технологије. Приликом учитавања стране са сервера стижу само неопходни подаци и приказују се непроменљиви елементи стране. Након учитавања основног садржаја стране или као реакција на акцију корисника извршава се асинхрони позив ка серверу. Сервер након одређеног времена претраживачу шаље податке у *JSON (JavaScript Object Notation)* формату. *JS* код у претраживачу обрађује ове податке и динамички генерише део странице.

4.3. Интеграција са *Azure Active Directory*

Како би се остварила рестрикција приступа подацима и функционалностима на основу идентитета и улоге корисника апликација је интегрисана са Мајкрософтовом *AAD* платформом. Ова платформа омогућава управљање идентитетима и правима приступа независно од система коме се приступа. Овакво решење погодно је јер се очекује да корисници система који је предмет овог рада већ поседују *AAD* налоге. Корисници *AAD* платформе подељени су у организације. Претпоставка је да корисници апликације која је предмет овог рада имају налоге унутар једне *AAD* организације. Поред аутентикације апликацији је неопходна и могућност делегирање права приступа од стране корисника. Делегирана права апликација користи како би у име аутентикованог корисника приступила Мајкрософтовоом сервису под називом *Graph API* [25]. Преко овог сервиса апликација има могућност претраживања корисника унутар *AAD* организације.

Решења за различите типске проблеме који се тичу интеграције платформе *AAD* и других апликације постоје у виду узорака на *GitHub* репозиторијуму *Azure-Samples* [26]. Један од тих узорака [27] се бави управо проблемом аутентикације. На слици 4.3.1 илустрован је сценарио који се остварује коришћењем овог узорака. Слика приказује корисника који жели да се аутентикuje на интернет апликацију. Апликација делегира процес аутентикације *AAD* платформи. Корисник се аутентикuje и бива преусмерен назад на основну апликацију. Као резултат аутентикације апликација добија идентификатор корисника (аутентикациони токен). Користећи овај токен корисник остварује вишеструки приступ апликацији без потребе да поново уноси креденцијале. Унутар самог токена налазе се основне информације о кориснику попут корисничког имена, јединственог идентификатора, имена за приказ, улога које корисник има и периода важења токена. Ове информације су кодиране, али нису шифроване те су стога јавно доступне. Како би се обезбедио интегритет информација део токена резервисан је за дигитални потпис у виду *HMACSHA256* кода. Прва четири слова назива кода означавају да се ради о коду за аутентикацију порука на бази хеш функције (*Hash-based Message Authentication Code*). Остатак назива дефинише хеш функцију којом се врши генерисање кода и верификација поруке. У овом случају ради се о *SHA256* хеш функцији. Верификацију токена врши апликација којој се токеном приступа и то самостално (без *AAD* аутентикационог сервера).

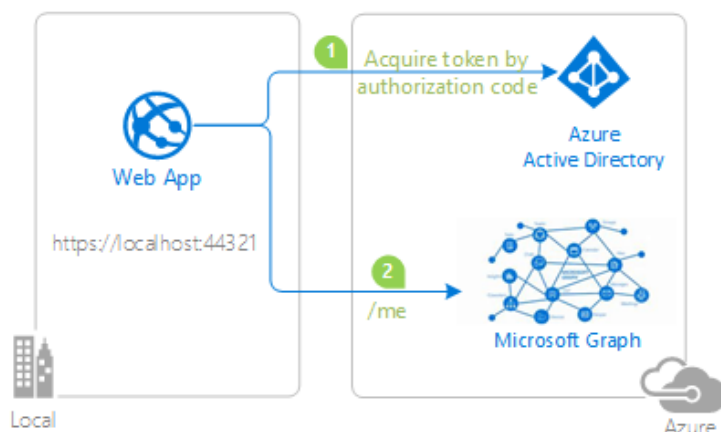


Слика 4.3.1 Дијаграм тока аутентикације [27]

Како би се остварила аутентикација, поред употребе имплементације из узорака, неопходно је регистровати апликацију унутар *AAD* организације. Регистрација се врши преко сервиса под називом *App registrations* који је део Мајкрософтовог *Azure* портала. Детаљно објашњење поступка постоји унутар датотеке *README.md* која је део репозиторијума поменутог узорака. Процес подразумева креирање нове апликације и дефинисање интернет адреса за пријављивање на систем и одјављивање са истог. Приликом креирања апликације генерише се њен идентификатор, тајни кључ и јавни кључ. Рестрикција приступа на основу улоге корисника није покривена поменутим узорком. Дефинисање улога такође се врши се преко сервиса *App registrations*. Једна од секција регистроване апликације омогућава приступ

њеном манифесту. Манифест апликације је датотека која садржи конфигурацију регистроване апликације у JSON формату. За дефинисање улога потребно је доделити вредност кључу „*appRoles*“ који се налази на првом нивоу датотеке. Вредност кључа треба да буде низ где сваки елемент низа представља једну корисничку улогу. Неке од информација које сваки од елемената низа дефинише за улогу су назив, опис, јединствени идентификатор улоге и да ли је улога омогућена. Манифест апликације која је предмет овог рада дефинише две корисничке улоге које представљају студенте и професоре.

Поменути узорак омогућава апликацији да оствари аутентикацију, али не и делегирање права приступа. У ту сврху употребљен је други узорак [28] који демонстрира комуникацију са *Graph API* сервисом. Овај узорак представља надоградњу претходног. За делегирања права приступа узорак користи механизам под називом *OpenID Connect*. Овај механизам представља проширење аутентикационог протокола *OAuth 2.0*. Постоје три могуће варијанте употребе овог механизма: имплицитна, коришћење ауторизационог кода и хибридна. Поменути узорак користи ауторизациони код како би остварио делегирање права приступа. Корисник даје сагласност за делегирање својих права апликацији. Резултат давања сагласности је једнократни ауторизациони код. На слици 4.3.2 приказан је ток механизма након добијеног ауторизационог кода. Интернет апликација шаље ауторизациони код *AAD* платформи. Платформа верификује код и шаље одговор у виду приступног токена. Интернет апликација користи приступни токен да у име корисника приступи трећем сервису (*Graph API*).



Слика 4.3.2 Делегиран приступ Мајкрософтовом *Graph API* сервису [28]

Како би се узорак интегрисао у апликацију неопходно у апликацији референцирати пројекте узорка. Конфигурациону датотеку апликације треба попунити интернет адресама за пријављивање и одјављивање са апликације, идентификатором и тајним кључем апликације као и називом и идентификатором организације. Процес аутентикације и делегирање приступа имплементиран је у посебном слоју апликације који се назива мидлвер. Код из овог слоја извршава се пре и након обраде *HTTP* захтева од стране контролера. Овај слој кода конфигурише се у методи *ConfigureServices* која је део класе *Startup*. Део ове методе који се односи на аутентикацију и делегирања права приступа приказан је на слици 4.4.3. Наредба која почиње на линији 52 укључује уграђени мидлвер за аутентикацију. Најпре се као механизам аутентикације дефинише *OpenIDConnect*, а затим се прослеђују вредности из конфигурационе датотеке које се везују за кључ „*AzureAd*“. Наредба која почиње на линији 57 конфигурише код за делегацију права приступа. Најпре се дефинише читање корисничког профила као подразумевано право које се захтева од свих аутентикованих корисника. На

крају наредбе дефинише се меморијски кеш као врста кеш меморије за чување приступних токена.

```
52 services.AddAuthentication(OpenIdConnectDefaults.AuthenticationScheme)
53     .AddSignIn("AzureAd", Configuration,
54         options => Configuration.Bind(Constants.AADConfigSection, options));
55
56 var initialScopes = new string[] { Constants.ScopeUserRead };
57 services.AddWebAppCallsProtectedWebApi(Configuration, initialScopes)
58     .AddInMemoryTokenCaches();
```

Слика 4.3.3 Конфигурација мидлвера за аутентикацију и делегацију приступа

За приступ већини метода контролера неопходно је да корисник буде аутентикован. Изузетак су методе које воде до јавних страница као што је почетна страна апликације. Из тог разлога дефинисана је глобална полиса у виду филтера која захтева да корисник буде аутентикован приликом приступа свим методама контролера. Наредба којом се дефинише ова полиса такође се налази у методи *ConfigureServices* класе *Startup* и приказана је на слици 3.4.4. Полису је могуће надјачати локално навођењем атрибута *AllowAnonymous* изнад методе или класе контролера.

```
88 services.AddControllersWithViews(options =>
89 {
90     var policyBuilder = new AuthorizationPolicyBuilder();
91     var policy = policyBuilder.RequireAuthenticatedUser()
92         .Build();
93     options.Filters.Add(new AuthorizeFilter(policy));
94 })
```

Слика 4.3.4 Конфигурација глобалног захтева аутентикације

Регулација права приступа на основу улоге аутентикованог корисника постиже се коришћењем атрибута *Authorize* на методама или класама контролера. Атрибут као аргумент прихвата једну или више улога. Релација између више улога наведених као аргумент једног атрибута је логичко ИЛИ. Другим речима, довољно је да корисник поседује једну од наведених улога како би приступио ресурсу. Над једном класом или методом могуће је навести и више од једног *Authorize* атрибута. Релација између више ауторизационих атрибута је логичко И.

Постоји неколико регуларних ситуација у којима се може десити да апликација није у стању да употреби претходно добијени приступни токен. Речено је да се приступни токени кеширају и да се за њихово кеширање користи меморијски кеш. Оваква кеш меморија није перзистентна. Уколико из неког разлога дође до прекида рада апликације кеш меморија ће приликом наредног покретања апликације бити празна. Како је ова меморија ограничена једном добијен приступни токен не мора бити у кеш меморији ни када апликација ради без прекида. Могуће је и да приступни токен просто истекне па је потребно поново затражити сагласност корисника. У узорку који употребљава ова апликација предвиђено је да се користи атрибут *AuthorizeForScopes* како би се ове ситуације разрешиле. Сам атрибут прихвата права приступа које је неопходно да корисник делегира како би се метода извршила. Уколико приликом приступа екстерном сервису дође до подизања изузетка извршава се имплементација овог атрибута. Најпре се утврђује врста изузетка односно да ли је проблем могуће отклонити поновним давањем сагласности од стране корисника. Уколико јесте, корисник се преусмерава на *AAD* портал како би се потребна сагласност обезбедила.

Решење коришћењем атрибута *AuthorizeForScopes* је коректно, али се заснива на претпоставци да се изворни код који приступа екстерном сервису налази у методи контролера који врши рутирање. Из таквих метода је могуће извршити преусмеравање интернет претраживача и добити сагласност корисника. Међутим исто решење није применљиво када се изворни код који приступа екстерном сервису налази у методи контролера која се позива асинхроно (извршавањем *AJAX* позива након што је страница већ учитана). Потреба за таквим сценариом јавља се на страници за додељивање испита студентима. Приликом претраге студената апликација приступа екстерном сервису *Graph API*.

Једно потенцијално решење било би већ поменуто коришћење атрибута *AuthorizeForScopes* уз освежавање целокупне стране након сваке претраге. Ово решење би функционисало уз одређене недостатке. У већини случајева приступни токен би био валидан, а освежавање беспотребно. Освежавање целокупне стране је непожељно и због чињенице да се мења само део садржаја. На крају, освежавање целокупне стране не доприноси добром корисничком искуству.

Друго могуће решење је покушати да се унутар асинхроне методе контролера идентификује стање у коме је неопходно преусмерити корисника. Затим, дефинисати посебан резултат који претраживачу сугерише да треба преусмерити корисника и извршити преусмерење из *JS* кода у претраживачу. Најправилније начин за пренос ове информације је преко статусног кода *HTTP* протокола. Иако би отклонио све проблеме претходног решења овај приступ сам по себи не би функционисао. Циљна интернет адреса приликом преусмеравања не налази се на истом домену као и интернет адреса апликације. Покушај преусмеравања у овом случају не би био дозвољен јер би се нарушила политика истог порекла (*Same Origin Policy* [28]).

Проблем је превазиђен комбинацијом два размотрена решења. На акцији контролера који учитава страницу потребно је применити атрибуте *AuthorizeForScopes* и *ImplicitAuthoriseForScopesTrigger*. На акцији контролера која се позива асинхроно потребно је применити атрибут *AjaxMsGraphProxy*. Претпоставимо да се страна успешно учитала, али је након тога токен постао недоступан. Асинхрони позив методи која врши претрагу резултује подизањем изузетка. Овај изузетак обрађује имплементација атрибута *AjaxMsGraphProxy* која клијентском коду прослеђује информацију да је неопходно добити сагласност од корисника. Клијентски код уместо покушаја преусмерења врши освежавање странице. Овим се не нарушава политика истог порекла. Пре поновног извршавања акције контролера који учитава страницу извршава се имплементација атрибута *ImplicitAuthoriseForScopesTrigger*. Имплементација проверава доступност токена што такође резултује изузетком. Коначно, последњи изузетак обрађује имплементација атрибута *AuthorizeForScopes*. Атрибут *AuthorizeForScopes* је део имплементације готовог узорка док су атрибути *ImplicitAuthoriseForScopesTrigger* и *AjaxMsGraphProxy* имплементирани као део изворног кода апликације која је предмет овог рада.

За претрагу студената приликом додељивања испита апликација користи верзију 1.0 *Graph API* сервиса. Конкретна метода сервиса назива се *Users* и спада у групу *HTTP GET* метода. Приликом сваког позива жељени број записа у резултату (величина стране) шаље се као параметар. У зависности од критеријума претраге унетих од стране професора као додатни параметри шаљу се филтери имена, презимена и корисничког имена. Између филтера поставља се релација логичко И. За разлику од неких других, метода *Users* не подржава филтрирање на основу садржаности вредности већ само на основу поклапања

почетног садржаја. Из овог ограничења потиче и различитост у критеријумима претраге студената у односу на претрагу других ентитета попут шаблона, задатака и испита. Резултат позива је колекција студената чија максимална величина одговара задатој величини стране. Уколико је број записа који одговара резултату претраге већи од задате величине стране у резултату се налази и интернет адреса за приступ наредној страни резултата. Формат резултата наредних страна не разликује се од прве. Последња страна не садржи интернет адресу за приступ наредној. Резултат методе не даје увид у укупан број страна након и дозвољава искључиво узастопно отварање нових страна. Из овог ограничења потиче и различитост у начину управљања страницама приликом претраге студената у односу на претрагу других ентитета попут шаблона, задатака и испита.

На одређеним странама у апликацији приказују се детаљи више корисника. Табеларни приказ шаблона садржи детаље о ауторима. Табеларни приказ резултата испита садржи детаље о студентима. Апликација не чува ове детаље већ само *AAD* идентификаторе корисника. За приступ детаљима групе корисника такође се користим метода *Users* сервиса *Graph API*. Идентификатори корисника шаљу се као параметри методе који врше филтрирање по критеријуму једнакости. Између филтера поставља се релација логичко ИЛИ. Како се параметри *HTTP GET* метода шаљу као део саме интернет адресе неопходно је водити рачуна о њеној укупној дужини. Да би се избегло прекорачење, информације о већем броју студената добијају се из неколико позива. Ради побољшања перформанси коришћена је парадигма асинхроног програмирања. Захтеви сервису се шаљу један за другим при чему се између захтева не чека на резултат претходног. Након што су сви захтеви послати апликација ослобађа тренутну нит и спремна је да обрађује друге захтеве претраживача. Извршавање нити наставља се тек када сервис одговори на све захтеве. Детаљи о корисницима који се приказују не представљају информације које се често мењају па и као такви погодни су за смештање у кеш меморију. На овај начин мањује се број неопходних позива ка сервису. Коришћена је уграђена имплементација механизма кеширања која користи радну меморију апликације. Параметре кеш механизма могуће је конфигурисати унутар секције *UserDetailCache* конфигурационе датотеке апликације. Доступни параметри дефинишу максимални капацитет кеш меморије, величину појединачних записа, време валидности записа без претходног приступа и укупно време валидности записа.

4.4. Базе података апликације

Како би перзистентно чувала податке који настају током извршавања апликација користи једну релациону базу података. Ова база података ће у наставку бити означена као примарна. Примарна база података креира се пре првог извршавања апликације. Различите функционалности апликације захтевају креирање додатних база. Ове базе података у наставку ће бити означене као динамичке. Динамичке базе података креирају се у току извршавања апликације. Изузетак је база података *Output_Tables*. Поменута база креира се пре првог извршавања, али се табеле унутар ње креирају динамички.

4.4.1. Примарна база података

Креирање примарне базе података као и интеракција апликације са њом врши се преко Мајкрософтове библиотеке *Entity Framework*. Принципи рада и предности ове библиотеке изложени су у одељку 4.3.1. У овом одељку ће бити изложен релациони модел примарне базе података.

Табеле *EFMigrationsHistory*, *ExceptionLog* и *RequestLog* издвајају се по томе што нису део ни једне конкретне функционалности апликације. Табела *EFMigrationsHistory* је објашњена у одељку 4.3.1. Табела *ExceptionLog* служи за чување информација о евентуалним изузецима до којих може доћи приликом извршавања апликације. Информације се чувају само за необрађене изузетке односно за изузетке које имплементација не предвиђа. За сваки необрађени изузетак у табелу се додаје један запис. Запис садржи информације о самом изузетку, време настанка записа као и идентификатор тренутно аутентикованог корисника. Уколико је до изузетка дошло приликом обраде *HTTP* захтева у табелу *RequestLog* се додаје додатни запис. Запис садржи информације о циљној интернет адреси захтева, коришћеној *HTTP* методи, времену настанка захтева, садржају тела, садржају заглавља итд.

Преостале табеле су у служби испуњавања функционалности апликације. Колоне које су заједничке овим табелама садрже информације о тренутку настанка записа, тренутку последње измене записа, аутору записа и кориснику који је извршио последњу измену над записом. Такође поседују и колону у којој се налази временски печат сваког записа. Његова вредност мења се приликом сваке промене записа. Ове вредности преносе се све до клијентског дела апликације. Клијентска страна ове вредности прослеђује приликом сваког захтева за измену или брисање записа. Када дође до покушаја измене или брисања неког записа уколико се временски печат захтева не слаже са временским печатом записа доћи ће до изузетка. Овим се спречавају стања утркивања између различитих корисника који покушавају да обришу или измене исти запис. Већина табела има и самогенеришући јединствени целобројни идентификатор који представља њихов примарни кључ. Изузетак су везне табеле које имају композитне примарне кључеве.

Табела *Template* садржи по један запис за сваки шаблон који настане у апликацији. Запис садржи назив шаблона, текстуални опис релационог модела шаблона, назив динамичке базе података која представља дати шаблон и информацију да ли је тај шаблон доступан студентима као шаблон за усавршавање знања. Не постоје зависности у односу на друге табеле.

Табела *Task* садржи по један запис за сваки задатак који настане у апликацији. Запис садржи назив задатка, текстуални опис задатка, назив динамичке базе која представља јавни скуп података задатка и скрипту која представља решење задатка. Колоном *TemplateId* дефинише се релација један према више са табелом *Template*.

Табела *SynthesisTest* садржи по један запис за сваки испит синтезе који настане у апликацији. Запис садржи назив и стање испита. Колоном *TaskId* дефинише се релација један према више са табелом *Task*.

Табела *SynthesisTestStudent* садржи по један запис за сваки пар испит-студент где је испит синтезе додељен студенту. Запис садржи информацију да ли је студент полагао испит и *SQL* скрипту коју је студент предао као одговор. Колоне *StudentId* и *SynthesisTestId* дефинишу релацију више према више између табела *AzureSQLUserMap* и *SynthesisTest*. Ове колоне уједно чине и композитни примарни кључ табеле *SynthesisTestStudent*.

Табела *AnalysisTest* садржи по један запис за сваки испит анализе који настане у апликацији. Запис садржи назив и стање испита. Колонама *StudentId* и *SynthesisTestId* дефинише се релација један према више са табелом *SynthesisTestStudent*.

Табела *AnalysisTestStudent* садржи по један запис за сваки пар испит-студент где је испит анализе додељен студенту. Запис садржи информацију да ли је студент полагао испит, назив базе података у којој се налази попуњавање које представља гранични случај као и

називе табела у бази *Output_Tables* које представљају излазе тачног решења и предмета анализе за дато попуњавање. Колоне *StudentId* и *AnalysisTestId* дефинишу релацију више према више између табела *AzureSQLUserMap* и *AnalysisTest*. Ове колоне уједно чине и композитни примарни кључ табеле *AnalysisTestStudent*.

Табела *AzureSQLUserMap* садржи по један запис за сваког студента који користи систем. Записи се креирају приликом прве доделе испита или приликом креирања инстанце за усавршавање знања. Садржи AAD идентификатор корисника и креденцијале корисника за сервер базе података.

Табела *SynthesisEvaluation* садржи по један запис за сваку компоненту оцене испита синтезе. Записи садрже информацију о фази процеса евалуације као и о компоненти оцене на коју се запис односи. Колонама *STS_StudentId* и *STS_SynthesisTestId* дефинише се релација један према више са табелом *SynthesisTestStudent*.

Табела *AnalysisEvaluation* садржи по један запис за сваку компоненту оцене испита анализе. Записи садрже информацију о фази процеса евалуације као и о компоненти оцене на коју се запис односи. Колонама *ATS_StudentId* и *ATS_AnalysisTestId* дефинише се релација један према више са табелом *AnalysisTestStudent*.

Табела *ExerciseInstance* садржи по један запис за сваку инстанцу која је креирана у апликацији. Записи садрже назив инстанце, назив базе података која представља инстанцу и скрипту коју је студент сачувао. Колоном *StudentId* дефинише се релација један према више са табелом *AzureSQLUserMap*. Колоном *TemplateId* дефинише се релација један према више са табелом *Template*.

Табела *SolutionColumn* садржи по један запис за сваку колону решења неког задатка. Запис садржи назив колоне и назив типа у бази података. Колоном *TaskId* дефинише се веза један према више са табелом *Task*.

4.4.2. Динамичке базе података и права приступа

За управљање динамичким базама података користи се уграђена C# класа *SqlCommand*. Ова класа омогућава извршавање произвољног упита задатог у виду ниске. На овај начин врши се креирање нових база или мењање модела постојећих у току извршавања апликације. Иста класа користи се и за извршавање упита које уносе студенти. Посебан део логике у апликацији води рачуна о креденцијалима под којима се упити извршавају у различитим ситуацијама. Додељивањем одговарајућих права приступа креденцијалима ограничава се приступ студената серверу базе података.

На нивоу сервера базе података апликација користи два администраторска налога. Први налог (A1) има право читања и писања. Други налог (A2) има само право читања. Приликом креирања сваког шаблона креира се једна база података. За извршавање упита којим професор дефинише релациони модел шаблона користи се налог A1. Приликом креирања задатка долази до клонирања базе података шаблона. За извршавање упита којим професор дефинише решење задатка користи се налог A2, а упит се извршава над клонираном базом. Коришћењем налога A2 спречава се да упит који професор дефинише као решење врши измене над подацима у бази.

Приликом додељивања испита студентима проверава се да ли сви студенти имају запис у табели *AzureSQLUserMap*. За оне који немају у табелу се уноси нови запис са насумично генерисаним креденцијалима. Исти логика извршава се када студент креира

инстанцу за усавршавање знања. У том случају провера се врши за студента који креира инстанцу.

Приликом додељивања испита синтезе студенту креира се налог за студента на нивоу базе података шаблона. Налог се додељује право читања базе. За креирање налога користе се студентови креденцијали из табеле *AzureSQLUserMap*. Исти креденцијали користе се за извршавање упита током полагања испита синтезе.

Приликом додељивања испита анализе студенту клонира се база података шаблона. На нивоу клониране базе креира се налог за студента. Налог добија права читања и писања на целој бази. У бази *Output_Tables* креирају се две табеле чија структура одговара формату тачног решења испита синтезе. Уколико студенту претходно нису додељивани испити анализе, на нивоу базе *Output_Tables*, креира се налог за студента. Налог се додају права читања и писања над креираним табелама. За креирање ових налога користе се студентови креденцијали из табеле *AzureSQLUserMap*. Исти креденцијали користе се за приступ клонираној бази шаблона и табелама у бази *Output_Tables* током полагања испита анализе.

Приликом креирања инстанце за усавршавање знања клонира се база података шаблона. На нивоу клониране базе креира се налог за студента. Налог добија права читања и писања на целој бази. За креирање налога користе се студентови креденцијали из табеле *AzureSQLUserMap*. Исти креденцијали користе се за извршавање упита током усавршавања знања.

Мајкросфотов сервер базе података пружа два модела аутентикације. Традиционални модел подразумева постојање налога на *Master* бази и мапирање тог налога на неке од осталих база. Приликом приступа одређеној бази података, корисник се најпре аутентичује на *Master* базу, а затим се утврђује да ли постоји одговарајуће мапирање на тражену базу. Код рестриктивног модела корисник има засебне налоге на појединачним базама. Приликом приступа одређеној бази корисник се аутентичује директно на тражену базу независно од *Master* базе. За потребе креирања налога студената на нивоу појединачних база коришћен је ограничен модел јер је рестриктивнији и не укључује приступ *Master* бази. [30]

4.5. Архитектура

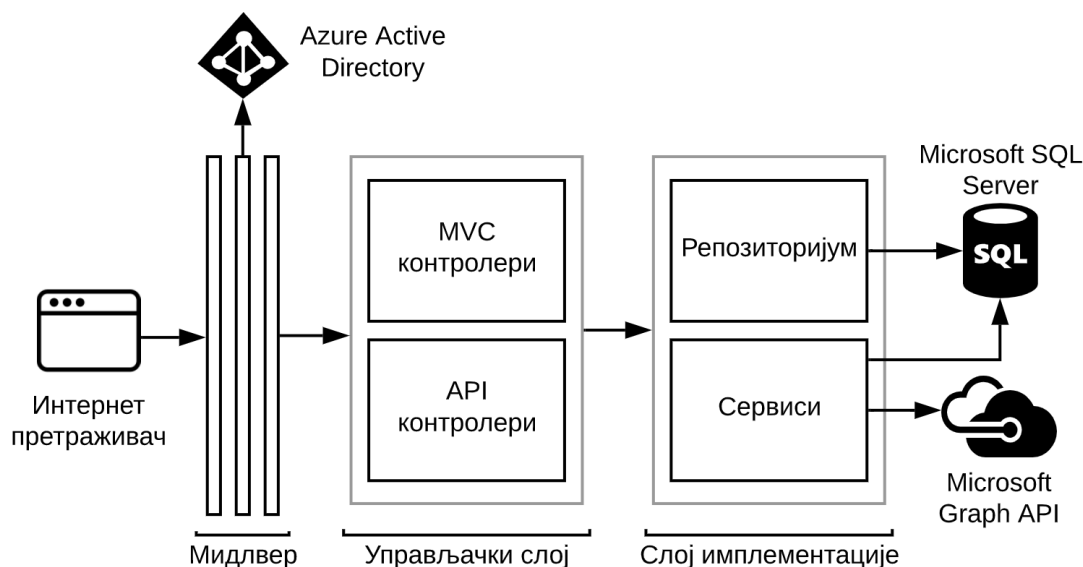
У овом одељку ће бити изложен образац по коме је структуриран изворни код апликације. Ради се о вишеслојној архитектури где су слојеви дефинисани улогом и међусобним зависностима изворног кода. Апликација комуницира са неколико тачака интеграције. Тачкама интеграције сматрају се интернет претраживач, AAD, сервер база података и *Graph API*. На слици 4.5.1 дат је шематски приказ структуре изворног кода и тачака интеграције. Могу се уочити три слоја изворног кода: мидлвер, управљачки слој и слој имплементације. Стрелице означавају зависности и усмерене су ка зависним елементима.

Мидлвер је слој који се у циклусу обраде *HTTP* захтева налази између интернет претраживача и управљачког слоја. Захтеви који долазе од стране претраживача најпре пролазе кроз појединачне сегменте мидлвера у задатом редоследу. Затим бивају обрађени од стране управљачког слоја. Резултат обраде се на крају враћа кроз исте сегменте мидлвера, али сада у обрнутом редоследу. Сегменти мидлвера дефинишу се у методи *Configure* класе *Startup*. Редослед у ком су сегменти наведени у изворном коду представља редослед улазне обраде *HTTP* захтева. Први сегмент врши рутирање захтева ка одговарајућем контролеру. Наредна два сегмента баве се аутентикацијом корисника и провером ауторизације. Ови

сегменти имају директну зависност ка *AAD* сервису. Најближи управљачком слоју налази се сегмент који бележи необрађене изузетке.

Управљачки слој подељен је на две целине. У једној се налазе *MVC* контролери који обрађују захтеве за учитавање одређене стране. У другој се налазе *API* (*Application Programming Interface*) контролери који обрађују асинхроне захтеве које претраживач шаље након учитавања стране. Сви контролери имају директну зависност ка имплементационом слоју. Улога контролера је валидација, трансформација и прослеђивање података деловима имплементационог слоја. Подаци могу бити делови *HTTP* захтева или међурезултати који долазе са имплементационог слоја.

Имплементациони слој подељен је на две функционалне целине: репозиторијум и сервис. У репозиторијуму се налазе класе које користећи *EntityFramework* читају или мењају садржај примарне базе података. Из зависност према примарној бази произилази зависност према серверу база података. Сервис садржи део имплементације која управља динамичким базама одакле потиче зависност према серверу база података. У сервису се налази и интеграција са *Graph API* интерфејсом, логика за евалуацију знања као и имплементације комуникације са интернет претраживачем посредством библиотеке *SignalR*. Оглашавање порука преко библиотеке *SignalR* независно је од тога да ли постоји претраживач који тренутно ослушкује поруке. Из тог разлога не постоји зависност сервиса ка интернет претраживачу.



Слика 4.5.1 Архитектура изворног кода

Основни образац софтверског дизајна који је коришћен приликом израде ове апликације јесте убацивање зависности (*Dependency Injection*). Убацивање зависности је један од начина да се постигне инверзија контроле између главне и зависне класе. Већина апликација написана је тако имплементације класа директно референцирају имплементације других класа. У оваквој ситуацији смер зависности у време компилације одговара смеру зависности у време извршавања. Референцирањем интерфејса уместо имплементација мења се смер зависности у време извршавања. Оваква имплементација погодна је јер изворни код чини модуларним. Модуларност дозвољава да се у потпуности промени имплементација неке класе без измена у остатку система. Једини услов је да имплементација и даље задовољава изложени интерфејс. Ово изворни код чини знатно лакшим за одржавање и надограђивање. Модуларност је такође погодна јер олакшава изоловано тестирање делова

имплементације коришћењем аутоматских тестова (*Unit Tests*). Коначно, убацивање зависности омогућава централизовану контролу животног века инстанци класа. Платформа *.NET Core* има уграђену подршку за овај образац софтверског дизајна. Предвиђена су три животна века: *Transient*, *Scoped* и *Singleton*. У методи *ConfigureServices* класе *Startup* дефинише се животни век за сваки пар интерфејс-имплементација [31].

Животни век *Transient* подразумева инстанцирање класе приликом сваког референцирања у току извршавања. Типично се користи за класе чије извршавање не зависи од стања апликације. Овакав опсег искоришћен је за класу *Evaluator* у којој је имплементирано утврђивање једнакости табела.

Животни век *Scoped* подразумева инстанцирање приликом првог референцирања на нивоу опсега извршавања. У контексту интернет апликација најчешћи опсег извршавања је циклус обраде *HTTP* захтева. Контролери користе овај опсег како би инстанцирали класе имплементационог слоја.

Животни век *Singleton* подразумева инстанцирање приликом првог референцирања у току животног циклуса апликације. Иста инстанца користи се током остатка извршавања. Ово је уједно и имплементација истоименог обрасца софтверског дизајна. У овој апликацији коришћен је за инстанцирање класа које имплементирају механизам кеш меморије.

4.6. Евалуација испита

Било да се ради о испиту синтезе или о испиту анализе процес евалуације студентовог одговора своди се на операције извршавање упита и упоређивање резултата. Време извршавања ових операција зависи од величине скупа података, комплексности самих упита и величине резултата који се упоређују. Очекује се да испит типично полаже неколико десетина студената. Процес евалуације читавог испита треба с тога сматрати процесом који се временски дуго извршава. Као такав овај процес није погодан за извршавање у опсегу обраде *HTTP* захтева. Дуга обрада *HTTP* захтева доприноси негативном корисничком искуству јер је корисник спречен да напусти страницу док претраживач чека резултат. У крајњем случају оваква ситуација може довести до грешке на страни претраживача услед истека времена за добијање одговора од сервера.

Неопходно је скратити време обраде *HTTP* захтева. Ово је могуће постићи измештањем процеса евалуације у засебан контекст. Обрада *HTTP* захтева треба да подразумева искључиво пријем захтева за извршавање процеса евалуације. Како би се скратило укупно време евалуације пожељно је процес поделити у независне целине и извршавати их у паралели. Апликација поделу процеса врши поделом захтева за евалуацију. Делови захтева се затим смештају у ред за чекање чиме се завршава опсег обраде *HTTP* захтева. Радно окружење преузима извршавање појединачних захтева из реда за чекање у независним контекстима. Апликација користи имплементацију реда за чекање унутар радне меморије коју пружа библиотека *Coravel* [32].

Као очигледна независна јединица обраде намеће се евалуација рада једног студента. Овакве јединице могуће је додатно поделити тако да свака компонента оцене студента представља независну јединицу обраде. Таква подела је уједно и оптимална када се ради о евалуацији испита синтезе. Код испита анализе процес је могуће додатно оптимизовати. Као што је приказано на слици 2.5.3.1, табеле T1 и T2 учествују у генерисању по две компоненте оцене. Додатна оптимизација процеса подразумева генерисање ових табела само једном у оквиру посебне јединице обраде. Тако генерисане табеле користе преостале три јединице

обrade које генеришу појединачне компоненте оцене. Треба напоменути да је овом оптимизацијом уведена зависност унутар евалуације једног студента. Тек након генерисања табела T1 и T2 компоненте оцена се могу независно генерисати. Извршавање кода ван циклуса обраде *HTTP* захтева представља отежавајућу околност за уметање зависности. Основни проблем је непостојање контекста под којим би се креирале инстанце класа. Како би се овај проблем превазишао у коду је било неопходно експлицитно декларисати контексте извршавања и разрешити зависности. Метода *CreateScope* интерфејса *IserviceScopeFactory* омогућава експлицитно креирање контекста. Контекст садржи поље типа *IserviceProvider* чија генеричка метода *GetService* омогућава експлицитно разрешавање зависности. На слици 4.6.1 дат је део имплементације методе која врши евалуацију одговора испита синтезе. На линији 114 експлицитно се креира један опсег извршавања. Опсег се креира у оквиру *using* наредбе чиме се дефинише део кода који обухвата. На линији 118 дефинисани опсег користи се за добијање инстанце репозиторијума за приступ подацима испита синтезе. На линији 175 исти опсег користи се за добијање инстанце сервиса за чување информација о необрађеним изузецима. У приложеном коду имплементација је обавијена *try-catch* блоком како би се забележили евентуални необрађени изузеци. У општем случају овај блок је сувишан јер се та имплементација налази у мидлверу. Конкретна имплементација извршава се ван обраде *HTTP* захтева па не долази до извршавања кода у мидлверу. Из тог разлога неопходно је експлицитно убележити евентуални изутетак.

```

114  using (var scope = serviceScopeFactory.CreateScope())
115  {
116      try
117      {
118          var synthesisRepository = scope.ServiceProvider.GetService<ISynthesisRepository>();
119          var sts = synthesisRepository.GetEvaluationData(testId, studentId);
120
121          Record_Start
122
123
124          Prepare_Data
125
126
127          Evaluate
128
129
130          Signal_End_And_Save_Result
131      }
132      catch (Exception ex)
133      {
134          var logRepo = scope.ServiceProvider.GetService<ILogRepository>();
135          logRepo.Log(ex);
136      }
137  }

```

Слика 4.6.1 Експлицитно дефинисање опсега и разрешавање зависности

Евалуација је измештена у засебан ток обраде који се одвија у позадини. Корисник више није везан за дату интернет страницу током трајања обраде. Како се одговор на захтев за покретање евалуације добија независно од процеса евалуације не може да садржи крајњи резултат евалуације. Један начин да корисник добије информацију о резултату процеса јесте да изврши освежавање странице. Ово очигледно не доприноси добром корисничком искуству. Поред тога, код дугих обрада података као што је евалуација испита, пожељно је обавештавати корисника о напретку процеса, а не само о крајњем резултату. Једно могуће решење било би стална провера информација на серверу коришћењем упосленог чекања. Овакво решење захтева додатни мрежни саобраћај, а додатним захтевима се непотребно повећава оптерећење сервера. Карактеристика модерних интернет апликација јесте да у

реалном времену осликавају стање система на страницама уз минималне режијске трошкове. Важно је уочити да је у овој ситуацији сервер иницијатор комуникације. Овакав вид комуникације разликује се од сценарија описаних у одељку 4.2 у којима је интернет претраживач увек иницијатор комуникације. Мајкрософт је развио библиотеку под називом *SignalR* [33] која нуди готову имплементацију оваквог вида комуникације. Библиотека се састоји из дела који се извршава на серверу и дела који се извршава у претраживачу. Представља апстракцију над различитим транспортним протоколима неопходним за одвијање комуникације између сервера и клијента у реалном времену. Иницијална веза се остварује коришћењем *HTTP* протокола. Уколико је то могуће врши се прелаз на *WebSocket* конекцију. Оваква конекција је најпогоднија јер најефикасније користи меморију сервера, има мало кашњење и омогућава истовремену комуникацију у оба смера. Са друге стране најзахтевнија је у погледу неопходног оперативног система и радног окружења. Апликација сада има механизам којим у реалном времену може да обавештава професора о напретку процеса евалуације испита. Изворни код за обраду независних јединица након сваке фазе информацију о напретку бележи у базу података. Користећи *SignalR* исту информацију о напретку оглашава интернет претраживачима.

5. ЗАКЉУЧАК

У оквиру овог мастер рада урађена је анализа процеса оцењивања знања полазника на примеру курса који се бави изучавањем релационих база података. Уочено је да са аспекта наставника оцењивање захтева креирање испита и анализу исправности одговора полазника. Уочено је и да је неопходно анализирати способности полазника да синтетишу и анализирају упите. Као резултат поменуте анализе дефинисани су захтеви на основу којих имплементирано софтверско решење у виду интернет апликације.

Апликација полазницима омогућава усавршавање теоријских знања извршавањем упита, полагање испита синтезе и полагање испита анализе. Наставницима апликација олакшава креирање испита и анализу исправности одговора. Код креирања испита синтезе циљ је био олакшати процес принципом реупотребљивости. На основу једне базе података могуће је креирати више задатака синтезе. Исти задатак синтезе могуће је користити за више испита. Креирање испита анализе сведено је на наставников одабир погодног одговора полазника на задатак синтезе. Анализу исправности одговора обавља апликација и тај процес је потпуности је аутоматизован.

На овај начин постигнуто је максимално растерећење наставника у процесу оцењивања знања полазника што је уједно и био примарни циљ овог рада. Поред тога процес учења и оцењивања прилагођен је како би што приближније осликавао проблеме са којима се полазници могу сусретати у пракси. Интеграцијом са екстерним сервисом за аутентикацију показано је да се овакав систем може прилагодити за употребу у организацијама које се примарно ослањају на софтвер одређеног произвођача.

Приликом израде апликације коришћени су принципи софтверског дизајна који олакшавају одржавање и евентуално надограђивање у складу са потребама корисника. Један могући смер надоградње био би омогућавање додатних типова испита попут провере теоријског знања. Такође корисна била би могућност анализе података ради уочавања честих грешака или области које су полазницима теже за разумевање.

На крају посебно треба истаћи да сва поменута запажања у овом закључку нису уско везана за изучавање релационих база података. У конкретном случају циљно окружење је Мајкрософтов *SQL Server*. Генерализацијом циљног окружења апликација би могла имати примену у изучавању других језика за управљање базама података као и различитих скриптних или програмских језика.

ЛИТЕРАТУРА

- [1] „SQL Server technical documentation“ www.docs.microsoft.com/sql/sql-server, последњи пут приступано јула 2020.
- [2] SQLBolt, SQLBolt www.sqlbolt.com, последњи пут приступано јула 2020.
- [3] SQLZoo, SQLZoo www.sqlzoo.net, последњи пут приступано јула 2020.
- [4] Pluralsight LLC, Pluralsight www.pluralsight.com, последњи пут приступано јула 2020.
- [5] Udemy Incorporated, Udemy www.udemy.com, последњи пут приступано јула 2020.
- [6] Exam Builder , Exam Builder www.exambuilder.com, последњи пут приступано јула 2020.
- [7] TALVIEW, TALVIEW www.talview.com, последњи пут приступано јула 2020.
- [8] Martin Dougiamas, Moodle www.moodle.org, последњи пут приступано јула 2020.
- [9] Code runner, Code runner www.coderunner.org.nz, последњи пут приступано јула 2020.
- [10] „Azure Active Directory fundamentals documentation“ www.docs.microsoft.com/en-us/azure/active-directory/fundamentals, последњи пут приступано јула 2020.
- [11] Microsoft Corporation, Azure Portal www.portal.azure.com, последњи пут приступано јула 2020.
- [12] jQuery Foundation, jQuery www.jquery.com, последњи пут приступано јула 2020.
- [13] Q-Success, „Usage statistics and market share of jQuery for websites“ www.w3techs.com/technologies/details/js-jquery#:~:text=jQuery%20is%20used%20by%2097.5,an%20extensive%20jQuery%20market%20report, последњи пут приступано јуна 2020.
- [14] Bootstrap ,Bootstrap www.getbootstrap.com, последњи пут приступано јула 2020.
- [15] CodeMirror, CodeMirror www.codemirror.net, последњи пут приступано јула 2020.
- [16] „.NET Core documentation“ www.docs.microsoft.com/dotnet/core, последњи пут приступано јула 2020.
- [17] „Entity Framework Core“ www.docs.microsoft.com/ef/core, последњи пут приступано јула 2020.
- [18] „Language Integrated Query (LINQ)“, www.docs.microsoft.com/dotnet/csharp/programming-guide/concepts/linq, последњи пут приступано јула 2020.

- [19] Microsoft Corporation, Visual Studio www.visualstudio.microsoft.com/vs, последњи пут приступано јула 2020.
- [20] Microsoft Corporation, GitHub www.github.com, последњи пут приступано јула 2020.
- [21] Стефан Ђорђевић, Master-Rad www.github.com/stef94dj/Master-Rad, последњи пут приступано јула 2020.
- [22] „What is SQL Server Management Studio (SSMS)?“, www.docs.microsoft.com/sql/ssms, последњи пут приступано јула 2020.
- [23] Lucid Software Incorporated, Lucidchart www.lucidchart.com, последњи пут приступано јула 2020.
- [24] Microsoft Corporation, Word www.microsoft.com/microsoft-365/word, последњи пут приступано јула 2020.
- [25] “Microsoft Graph REST API v1.0 reference”, www.docs.microsoft.com/graph/api/overview?toc=.%2Fref%2Ftoc.json&view=graph-rest-1.0, последњи пут приступано јула 2020.
- [26] Azure-Samples www.github.com/Azure-Samples, последњи пут приступано јула 2020.
- [27] Jean-Marc Prieur „An ASP.NET Core Web app signing-in users with the Microsoft identity platform in your organization“ www.github.com/Azure-Samples/active-directory-aspnetcore-webapp-openidconnect-v2/tree/master/1-WebApp-OIDC/1-1-MyOrg, последњи пут приступано јула 2020.
- [28] Jean-Marc Prieur „Using the Microsoft identity platform to call the Microsoft Graph API from an An ASP.NET Core 2.x Web App, on behalf of a user signing-in using their work and school or Microsoft personal account“ www.github.com/Azure-Samples/active-directory-aspnetcore-webapp-openidconnect-v2/tree/master/2-WebApp-graph-user/2-1-Call-MSGraph, последњи пут приступано јула 2020.
- [29] „Same-origin policy“ www.developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy, последњи пут приступано јула 2020.
- [30] „Authentication in SQL Server“, www.docs.microsoft.com/dotnet/framework/data/adonet/sql/authentication-in-sql-server, последњи пут приступано јула 2020.
- [31] Steve Smith, Scott Addie, Brandon Dahler, „Dependency injection in ASP.NET Core“ www.docs.microsoft.com/aspnet/core/fundamentals/dependency-injection, последњи пут приступано јула 2020.
- [32] James Hickey, „Queuing“ www.docs.coravel.net/Queuing , последњи пут приступано јула 2020.
- [33] „Tutorial: Get started with ASP.NET Core SignalR“ www.docs.microsoft.com/aspnet/core/tutorials/signalr, последњи пут приступано јула 2020.

СПИСАК СКРАЋЕНИЦА

AAD	Azure Active Directory
API	Application Programming Interface
AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
EF	Entity Framework
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
JS	Java Script
JSON	Java Script Object Notation
LINQ	Language Integrated Query
MVC	Model View Controller
SQL	Structured Query Language

СПИСАК СЛИКА

Слика 2.4.3.1 Евалуација синтетичког знања	5
Слика 2.5.3.1 Евалуација аналитичког знања	7
Слика 3.1.1.1 Почетна страна неаутентикованог корисника	9
Слика 3.1.2.1 Почетна страна аутентикованог професора	10
Слика 3.1.3.1 Давање сагласности професора	11
Слика 3.2.1.1 Табеларни преглед шаблона	12
Слика 3.2.1.2 Неуспешно креирање шаблона	13
Слика 3.2.1.3 Акције за управљање шаблоном	13
Слика 3.2.1.4 Прозор за дефинисање/измену описа шаблона	14
Слика 3.2.1.5 Неуспешно брисања шаблона	14
Слика 3.2.2.1 Табеларни преглед задатака	15
Слика 3.2.3.1 Табеларни преглед испита синтезе	17
Слика 3.2.4.1 Табеларни преглед испита анализе	18
Слика 3.2.5.1 Дефинисање модела базе података	20
Слика 3.2.6.1 Попуњавање садржаја базе података	21
Слика 3.2.7.1 Додељивање испита студентима	22
Слика 3.2.8.1 Евалуација испита синтезе	23
Слика 3.2.8.2 Евалуација испита анализе	24
Слика 3.3.1.1 Страница за вежбе - шаблони	25
Слика 3.3.1.2 Елемент прегледа инстанци	25
Слика 3.3.2.1 Елементи прегледа испита	26
Слика 3.3.3.1 Извршавање упита над инстанцом	27
Слика 3.4.1 Секција <i>All applications</i> апликације <i>Azure Portal</i>	28
Слика 3.4.2 Секција <i>Users and groups</i> апликације регистроване на AAD домену	28
Слика 3.4.3 Додељивање улога корисницима кроз апликацију <i>Azure Portal</i>	29
Слика 4.3.1 Дијаграм тока аутентикације	33
Слика 4.3.2 Делегиран приступ Мајкрософтовом <i>Graph API</i> сервису	34
Слика 4.3.3 Конфигурација мидлвера за аутентикацију и делегацију приступа	35
Слика 4.3.4 Конфигурација глобалног захтева аутентикације	35
Слика 4.5.1 Архитектура изворног кода	41
Слика 4.6.1 Експлицитно дефинисање опсега и разрешавање зависности	43