Математичка гимназија

МАТУРСКИ РАД

ИЗ РАЧУНАРСТВА И ИНФОРМАТИКЕ

Апликација за одређивање ђака генерације

Ученик: **Филип Шумић**

Ментор: **Милош Арсић**

Београд, мај 2026

Садржај

1 УВОД

У савременом школском окружењу, све је важније да се подаци о ученицима и њиховим постигнућима воде прегледно и тачно. Једна од значајних одлука на крају школовања јесте избор ђака генерације. До сада се ђак генерације одређивао ручно, што је непрактично јер је за тај избор потребна велика количина података о ученицима: општи успех, резултати на такмичењима, друштвене, хуманитарне и уметничке активности и истраживачки радови.

Тема овог матурског рада је израда десктоп апликације, чији је циљ да олакша процес одређивања ђака генерације. Апликација омогућава унос и чување података о ученицима, такмичењима, освојеним наградама и истраживачким радовима. На основу тих података апликација приказује преглед поена и коначну ранг-листу ученика. На тај начин се смањује могућност грешке при ручном рачунању и омогућава јаснији увид у то одакле потичу поени сваког ученика.

Апликација је намењена професорима који учествују у избору ђака генерације. Посебна пажња посвећена је томе да подаци буду организовани по генерацијама, јер се апликација може користити више година. Свака генерација има своје ученике и резултате, док се одређени подаци, као што су такмичења и предмети, могу користити и даље кроз наредне школске године.

У раду ће бити објашњена структура апликације, организација базе података, најважније странице, помоћне класе, корисничке контроле и механизми који омогућавају унос, измену, брисање и приказ података. На крају рада биће написане и смернице за коришћење апликације.

2 КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ И ОСНОВНИ ПОЈМОВИ

За израду апликације коришћено је више технологија које заједно омогућавају прављење десктоп програма, приказ корисничког интерфејса, рад са базом података и касније дистрибуцију апликације. Главне технологије коришћене у овом пројекту су WPF (Windows Presentation Foundation), програмски језик C#, XAML (Extensible Application Markup Language), SQLite база података, библиотека System.Data.SQLite.Core, Git и GitHub.

2.1 WPF

Апликација је направљена помоћу WPF технологије. WPF је технологија намењена изради графичких десктоп апликација за Windows оперативни систем. Заснована је на .NET платформи и омогућава креирање модерних апликација са прозорима, страницама, дугмадима, пољима за унос, табелама, менијима и другим визуелним елементима.

У WPF апликацијама често се раздваја изглед програма од његове логике. Изглед се најчешће описује помоћу језика XAML, док се понашање апликације пише у програмском језику C#. То омогућава бољу организацију кода и лакше одржавање програма.

2.2 Основни WPF појмови

Пошто је апликација направљена као WPF десктоп апликација, важно је објаснити неколико основних појмова који се користе у њеној структури.

Прозор представља основну визуелну целину десктоп апликације. У WPF-у се прозор представља класом Window. Главни прозор ове апликације је MainWindow. Поред главног прозора, у апликацији постоје и мањи помоћни прозори, који врше споредне функције. Неки прозори имају функционалност дијалога, те питају корисника нешто и у методу која их је позвала враћају одговор. Један од таквих прозора је уграђени **MessageBox**, који је пуно коришћен у овој апликацији.

Страница представља део апликације који се приказује унутар главног прозора. У WPF апликацијама се страница представља класом Page. Овакав начин организације омогућава да апликација има један главни прозор, док се садржај мења у зависности од тога коју функционалност корисник изабере.

Контрола је визуелни елемент са којим корисник комуницира. То могу бити дугмад, поља за унос текста, падајуће листе, табеле за приказ података и слично. У WPF-у постоје готове контроле као што су Button, TextBox, ComboBox и DataGrid. Поред тога, могуће је направити и сопствене корисничке контроле.

Догађај (енг. event), представља механизам којим апликација реагује на радњу корисника. На пример, клик на дугме, избор ставке из падајуће листе или промена текста у пољу за унос могу покренути одређени догађај. У WPF апликацијама догађаји су повезани са методама у C# коду, па се на тај начин одређује шта ће се десити када корисник изврши неку радњу.

2.3 C# и XAML

Логика апликације написана је у програмском језику C#. **C#** је модеран, објектно оријентисан програмски језик који се често користи за израду Windows апликација. У овом пројекту C# је коришћен за обраду кликова на дугмад, унос података, проверу исправности унетих вредности, комуникацију са базом података и учитавање података у табеле. C# омогућава организовање кода у класе, методе и интерфејсе.

Класа је програмска целина која групише податке и методе које припадају истој логичкој целини. Класа може бити јавна или приватна. Уколико је класа јавна, све методе и сви делови пројекта могу видети њен садржај, док се приватне класе могу дефинисати само унутар друге класе и виде се само тамо где су дефинисане. Од уобичајених класа може се креирати објекат који је инстанца те класе. Постоје разни типови класа, али нама су битни још само static, класа од које се не може креирати инстанца, већ се у њој само налазе методе и својства и partial, класа која се дефинише кроз више различитих фајлова.

Метода је део кода који извршава одређену радњу. Служи да групише логику која извршава одређене задатке, омогућавајући вишекратно коришћење истог кода без потребе за његовим копирањем.

Интерфејс је посебан тип у C#-у који дефинише које методе нека класа треба да има, али не мора да одређује како ће те методе бити реализоване.

Изглед апликације дефинисан је помоћу језика XAML. **XAML** је декларативни језик који се користи за описивање корисничког интерфејса у WPF апликацијама. У XAML фајловима дефинишу се распоред контрола, њихова величина, боје, стилови, маргине и повезивање са догађајима из C# кода. На пример, изглед страница за додавање ученика, додавање такмичења, приказ ранг-листе и преглед поена описан је у језику XAML, док се логика тих страница налази у одговарајућим C# фајловима.

2.4 База података, SQL и SQLite

База података служи за трајно чување података које апликација користи. Уместо да се подаци чувају само док је програм покренут, база омогућава да они остану сачувани и након затварања апликације.

Подаци у бази организовани су у табеле (ентитете). На пример, табела ученик чува податке о ученицима, табела такмичење податке о такмичењима, а табела предмет податке о предметима.

Свака табела састоји се од редова и колона. Једна колона представља један атрибут, односно једно именовано својство ентитета. Један ред представља једну инстанцу неког ентитета. На пример, једна колона ентитета ученик је име, док један ред представља све податке о ученику Мирку Петровићу.

За рад са базом користи се језик SQL (Structured Query Language). SQL је стандардизовани језик који служи за креирање, читање, измену и брисање података у релационим базама података. У апликацији се користе SQL команде као што су SELECT, INSERT, UPDATE и DELETE. Команда SELECT служи за читање података, INSERT за додавање нових података, UPDATE за измену постојећих, а DELETE за брисање података.

У овом пројекту коришћена је SQLite база података. SQLite је систем за управљање базом података који користи SQL као језик за разговор са базом. То је лаган систем који не захтева посебан сервер, већ се цела база налази у једном фајлу. То је погодно за ову апликацију јер се база може чувати локално уз сам програм.

Поред табела, база садржи и погледе, односно view структуре. Поглед је SQL упит сачуван у бази који приказује податке добијене из једне или више табела.

2.5 Повезивање апликације са базом

За повезивање C# апликације са SQLite базом коришћена је библиотека System.Data.SQLite.Core. Ова библиотека омогућава отварање конекције ка бази, извршавање SQL упита и читање резултата.

У апликацији се преко ове библиотеке извршавају SQL команде за додавање, измену, брисање и приказ података. Такође се користе параметризовани упити, што значи да се вредности које корисник уноси не убацују директно у SQL текст, већ се прослеђују као параметри. Такав приступ чини рад са базом безбеднијим и поузданијим, јер смањује могућност грешке при формирању упита.

2.6 Git u GitHub

За чување верзија при развоју апликације коришћен је Git. Git је систем за контролу верзија који омогућава да се изворни код апликације чува у репозиторијуму и да се прате измене кроз комит (енг. commit) функционалност. На тај начин могуће је видети које су измене направљене током развоја и по потреби се вратити на раније стање пројекта.

Ради лакшег дељења апликације, пројекат је касније пребачен на GitHub. GitHub је онлајн платформа која омогућава чување Git репозиторијума на интернету и приступ пројекту са других рачунара. То у нашем случају омогућава да пројекат потенцијално дораде будуће генерације. Преко GitHub release функционалности могуће је објавити готову верзију апликације у облику zip фајла, тако да корисник не мора тражити програмеру да му пошаље програм, већ га може сам инсталирати са GitHub-а.

На тај начин GitHub служи као место за чување изворног кода, али и као начин за дељење апликације и потенцијално унапређивање у будућности.

3 УОЧАВАЊЕ ПОТРЕБА АПЛИКАЦИЈЕ, ПЛАНИРАЊЕ, КАСНИЈЕ ПРОМЕНЕ И АРХИТЕКТУРА БАЗЕ ПОДАТАКА

Први кораци развоја сваке апликације су уочавање потреба апликације и планирање. Пре почетка развоја, потребно је знати шта треба развијати. Процес планирања ове апликације почео је од разумевања критеријума за одређивање ђака генерације (слика 3.2) и планирања архитектуре базе на основу њега. Одмах је било очигледно да ће у бази бити потребни ентитети за ученика и такмичење, јер је потребно за сваког ученика и такмичење знати неколицину података. Такође, било је јасно да ће бити потребан ентитет за везу између ученика и такмичења, јер се више ученика може такмичити на истом такмичењу, а један ученик може учествовати на више такмичења. Оно што је такође претпостављено у првој верзији модела базе је да ће бити потребни и посебни ентитети за истраживачки рад и друштвено, хуманитарно, уметничке активности, као и њихове везе са учеником. Та претпоставка подразумева да се у базу уноси које су тачно активности ученици радили и имена радова које су писали. Са овом архитектуром, настала је првобитна база.

Пред почетак развоја апликације, одлучио сам да желим да апликација памти базу података за све године када је коришћена, те сам увео у базу ентитет генерација. Тиме је омогућено да се ученици поделе по годишту, али и да се бодовање такмичења подели по генерацији. То је важно, јер значи да се резултати од претходних година чувају веродостојно, исти какви су тада израчунати. Када бодовање такмичења не би зависило од генерације, онда би променом бодовања, бивали промењени и резултати претходних година.

Последња промена архитектуре базе настала је након разговора са наставницом упознатом са процесом одређивања ђака генерације. Закључак је био да је лакше за активности ученика памтити само поене за сваку од 4 године школовања у ентитету ученик, јер се при одређивању ђака генерације активности бодују за све ученике, а не обраћа се пуно пажње на њих, те је сигурно практичније уносити само поене, него уносити све активности којима су се ученици бавили. Истраживачки рад остаје посебан ентитет јер се ретко дешава да ученици имају радове, па није проблем уносити их. Тако долазимо до финалне архитектуре базе (слика 3.1). Она садржи следеће табеле:

generacija - Чува школске генерације, на пример 2025/2026. Колона aktivna означава која је генерација тренутно изабрана у апликацији.

ucenik - Чува податке о ученицима: име, презиме, одељење, успех по годинама, поене за активности по годинама и генерацију којој ученик припада.

predmet - Чува предмете такмичења, на пример математика, физика, информатика. Табела омогућава да се предмети додају и бришу кроз апликацију.

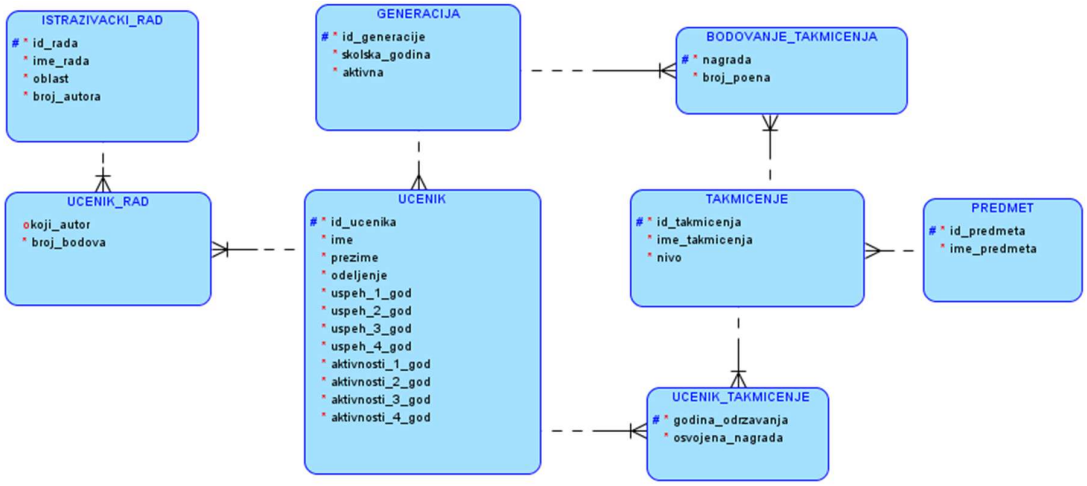
takmicenje - Чува дефиниције такмичења: назив такмичења, предмет и ниво такмичења. Свако такмичење је повезано са табелом predmet. Такмичења су подељена у нивое: домаће, међународно – променљиви поени и међународно – фиксни поени. То је природно јер се у правилнику такмичења различито бодују по ове 3 категорије. (Слика 3.2)

bodovanje_takmicenja - Чува број поена за прву, другу и трећу награду за свако такмичење у одређеној генерацији.

ucenik_takmicenje - Чува резултате ученика на такмичењима. Повезује ученика са такмичењем и памти коју је награду ученик освојио.

istrazivacki_rad - Чува податке о истраживачким радовима: назив рада, област и број аутора.

ucenik_rad - Повезује ученике са истраживачким радовима и чува број поена које је ученик добио за одређени рад.



Слика 3.1 Модел финалне базе података

Уочавање потреба саме апликације зависи највише од базе података и потреба корисника. Прва ствар коју сам уочио је да је генерација централни ентитет у бази и да већина осталих ентитета зависе од ње, те сам одлучио да се она бира на врху екрана и да је тај одабир видљив у свим страницама. Следеће сам приметио да треба омогућити уписивање и брисање података у све табеле базе које дозвољавамо кориснику да мења, да би корисник могао контролисати који ће подаци бити у бази. Следеће потребне функционалности биле су приказ ранг листе, да би се могло одредити ко је ђак генерације и приказ база, у коме се приказују сви основни ентитети, али не и везе између њих. Везе су измештене у посебну функционалност преглед поена, ради груписања свих извора поена ученика и ради лакшег приказа. Следећи корак био је омогућити измену постојећих података, а то је урађено преко омогућавања едита за страницу база и увођења функционалности за промену бодовања такмичења. Промена бодовања такмичења је јако важна јер критеријум није егзактан, те бодовање неких такмичења зависи од одлуке комисије сваке године. Последње, одлучио сам да убацим неке функционалности које олакшавају посао кориснику. Неке од њих су основне (енг. default) вредности бодова за активности при уносу ученика (10) и могућност за каснију измену, и прилагођене корисничке контроле, нпр. претраживи ComboBox који олакшава одабир елемената из дугих листа. О архитектури апликације биће детаљније речи у наредном поглављу.

ПРЕДЛОГ КРИТЕРИЈУМА ЗА ОДРЕЂИВАЊЕ ЂАКА ГЕНЕРАЦИЈЕ

ТАКМИЧЕЊА	**Међународна такмичења**	Прва награда	Друга награда	Трећа награда
	Олимпијада из математике, физике, информатике	28	16	10
	Олимпијада из астрономије, астрофизике, хемије; Јуниорска научна олимпијада	18	11	8
	ЕГМО	16	8	5
	Међународна такмичења (Балканијада, Жаутиковска олимпијада, Олимпијада метропола, Румунски мастер из математике, информатике и физике, Европска физичка олимпијада…)[1]	18-8	11-7	8-4
	Домаћа такмичења	Пласман међу најбољих 8% ученика[2]	Пласман међу најбољих 9-25% ученика[3]	Пласман међу најбољих 25-50% ученика[4]
	СМО, ЈСМО, СИО, ЈСИО, СФО, ЈСФО, СХО	15	9	6
	Државно такмичење из математике, физике, информатике и хемије (у категорији ученика Математичке гимназије ако је има)	10	6	4
	Државно такмичење из осталих предмета	5	2	1
ОПШТИ УСПЕХ	до 40 поена			
ДРУШТВЕНЕ, ХУМАНИТАРНЕ, УМЕТНИЧКЕ АКТИВНОСТИ	до 40 поена (до 10 поена по школској години)			
ОБЈАВЉЕНИ ИСТРАЖИВАЧКИ РАДОВИ (по школској години)	до 10 поена ако је ученик једини аутор до 8 поена ако је ученик један од два коаутора до 6 поена ако је ученик један од три коаутора до 4 поена ако је ученик један од четири или више коаутора			

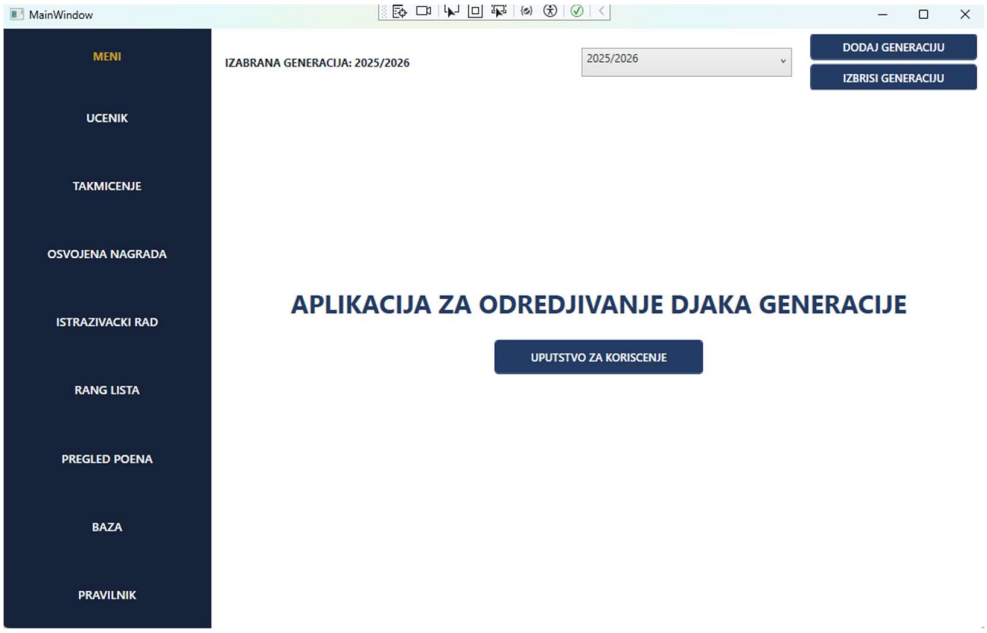
[1] Одлуку о тачном броју поена из датих интервала доносе одговарајућа стручна већа.
[2] Заокружено на први већи цео број.
[3] Заокружено на први већи цео број.
[4] Заокружено на први већи цео број.

Слика 3.2 Критеријум за одређивање ђака генерација

4 АРХИТЕКТУРА АПЛИКАЦИЈЕ

Апликација је организована као WPF десктоп апликација са једним главним прозором и већим бројем страница које се учитавају унутар њега. Оваква организација омогућава да корисник има сталан главни мени са леве стране, док се садржај у централном делу прозора мења у зависности од изабране функционалности. На тај начин апликација изгледа као јединствен програм, иако је њена логика подељена на више мањих целина.

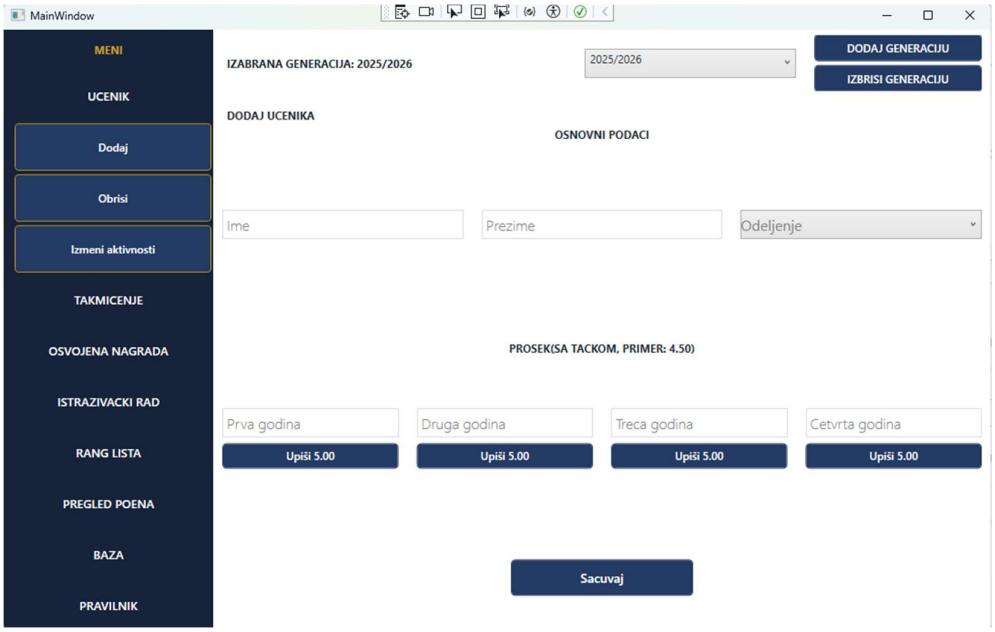
Главни прозор апликације је MainWindow (слика 4.1). Он представља основу целог програма и садржи леви мени, горњу контролу за избор генерације и простор у коме се приказују различите странице. Леви мени је подељен на више секција: UCENIK, TAKMICENJE, OSVOJENA NAGRADA, ISTRAZIVACKI RAD, RANG LISTA, PREGLED POENA, BAZA и PRAVILNIK. Неке секције менија могу да се отварају и затварају, јер у себи садрже више сродних опција. На пример, секција UCENIK садржи опције за додавање, брисање и измену активности ученика (ово се види на левој страни слике 4.2), док секција такмичење садржи опције за додавање, брисање, измену бодовања и страницу предмет.



Слика 4.1 Почетни екран апликације

За приказ страница користи се MainFrame. То је WPF контрола која омогућава да се у једном делу прозора учитавају различите Page странице. Када корисник кликне на неко дугме у менију, не отвара се нови главни прозор, већ се у MainFrame учитава одговарајућа страница. На пример, кликом на UCENIK > Dodaj учитава се страница DodajUcenika (слика 4.2), док се кликом на RANG LISTA учитава страница RangLista.

Централна метода за навигацију кроз апликацију је NavigateToPage(…) (Слика 4.3). Ова метода прима страницу коју треба приказати и информацију о томе да ли треба приказати горњи избор генерације (ако се не прецизира узима да треба). Већина страница зависи од тренутно изабране генерације, па се код њих приказује контрола за избор генерације. Са друге стране, странице BAZA и PRAVILNIK нису везане за једну генерацију, па се приликом њиховог отварања избор генерације сакрива. Тако MainWindow контролише не само која страница се приказује, већ и како изгледа распоред главног дела прозора.



Слика 4.2 Страница за Додавање ученика

Отварање и затварање секција у левом менију реализовано је методом ToggleMenuSection(...). Она мења видљивост подменија и висину реда у коме се он налази. Ако је секција већ отворена, она се затвара; ако је затворена, поново се

приказује. Ово омогућава да мени остане прегледан и да корисник види само оне групе опција које су му тренутно потребне.

```
19 references
private void NavigateToPage(Page page, bool prikaziGeneraciju = true)
{
    if (!MozeDaSeNapustiTrenutnaStranica())
    {
        return;
    }

    PostaviVidljivostGeneracije(prikaziGeneraciju);
    MainFrame.Navigate(page);
}

2 references
private bool MozeDaSeNapustiTrenutnaStranica()
{
    if (MainFrame.Content is IStranicaSaIzmenama stranicaSaIzmenama &&
        stranicaSaIzmenama.ImaNesacuvaneIzmene())
    {
        return stranicaSaIzmenama.PotvrdiNapustanjeStranice();
    }

    return true;
}
```

Слика 4.3 Метода за прелазак на другу страницу и помоћна метода

Важан део архитектуре је и контрола GeneracijaOdabir. Она се налази у горњем делу апликације и служи за избор активне генерације. Генерација је битна зато што се ученици, награде, ранг-листа и преглед поена односе на одређену школску годину. Када корисник изабере другу генерацију, апликација ажурира глобалну промењљиву GlobalnePromenljive.IzabranaGeneracija и поново учитава тренутну страницу ако је она зависна од генерације. На тај начин се обезбеђује да корисник увек ради са подацима који припадају тренутно изабраној генерацији. Ова контрола такође омогућава и додавање генерација у базу и брисање генерација из базе.

Апликација је подељена на више врста компоненти. Странице, односно Page класе, користе се за главне функционалности као што су додавање ученика, брисање такмичења, измена бодовања, приказ ранг-листе и рад са базом. Посебни прозори, односно Window класе, користе се за мање издвојене радње, као што су додавање генерације, брисање генерације, додавање предмета или приказ упутства. Корисничке контроле, односно UserControl класе, користе се за делове интерфејса који се понављају на више места и/или се не могу реализовати коришћењем интегрисаних контрола.

Посебна пажња посвећена је спречавању губитка несачуваних измена. За то је направљен интерфејс IStranicaSaIzmenama (Слика 4.4). Странице које могу да имају несачуване измене имплементирају овај интерфејс и дефинишу методу ImaNesacuvaneIzmene(). Када корисник покуша да пређе на другу страницу или да промени генерацију, MainWindow проверава да ли тренутна страница има несачуване измене (позивом методе ImaNesacuvaneIzmene(), ово се види у другој методи на слици 4.3). Ако их има, кориснику се приказује упозорење и он може да одлучи да ли жели да напусти страницу или да остане на њој (метода PotvrdiNapustanjeStranice()).

Оваква архитектура омогућава да апликација буде прегледна и лакша за одржавање. Главни прозор управља навигацијом и распоредом, док су појединачне функционалности издвојене у посебне странице. Заједничка логика, као што су учитавање података из базе или избор активне генерације, издвојена је у посебне класе и контроле. На тај начин се избегава непотребно понављање кода и сваки део апликације има јасну улогу.

```
6 references
public interface IStranicaSaIzmenama
{
    7 references
    bool ImaNesacuvaneIzmene();

    3 references
    bool PotvrdiNapustanjeStranice()
    {
        MessageBoxResult rezultat = MessageBox.Show(
            "Imas nesacuvane izmene. Da li zelis da napustis stranicu?",
            "Nesacuvane izmene",
            MessageBoxButton.YesNo,
            MessageBoxImage.Warning);

        return rezultat == MessageBoxResult.Yes;
    }
}
```

Слика 4.4 Интерфејс за контролу несачуваних измена

5 КЛАСЕ И КОНТРОЛЕ

У апликацији су, поред страница и прозора, направљене и помоћне класе и корисничке контроле. Њихова улога је да се код боље организује и да се избегне понављање истих делова програма.

5.1 Класе

Једна од најважнијих помоћних класа је DatabaseService. Њена улога је да обезбеди повезивање апликације са SQLite базом података. Уместо да се путања до базе и отварање конекције пишу на свакој страници посебно, ова логика је издвојена у једну класу. Такође, ова класа не користи апсолутну путању која важи само на једном рачунару, већ проналази базу релативно у односу на покренуту апликацију или пројекат. Важно је и то што апликација не прави празну базу ако фајл базе не постоји, већ јасно јавља грешку.

Класа GlobalnePromenljive користи се за чување променљивих које сви делови кода треба да виде. Најважнија вредност у овој класи је IzabranaGeneracija, која представља тренутно изабрану школску генерацију. Пошто велики број страница зависи од генерације, ова вредност омогућава да све странице користе исти податак. У овој класи налази се и метода UcitajIdIzabraneGeneracije(), која из базе проналази идентификатор тренутно изабране генерације. На тај начин странице не морају саме да пишу исти SQL упит сваки пут када им је потребан id_generacije.

Класа UcitavanjePodataka направљена је да би се избегло понављање истих упита за учитавање података, пошто се у апликацији на више места учитавају ученици, такмичења, предмети и истраживачки радови. Све методе ове класе врше SELECT упите над базом. Методе ове класе су: UcitajUcenike(), UcitajUcenikeZaPrikaz(), UcitajTakmicenja(), UcitajTakmicenjaZaPrikaz(), UcitajPredmete() и UcitajRadove(). Методе које учитавају за приказ користе се за убацивање података у DataGrid, па учитавају све податке као посебне променљиве, да би их приказале табеларно. За разлику од њих, остале методе се углавном користе за попуњавање опција ComboBox контрола, те је практичније податке учитати у формату кључ, приказана вредност, где је кључ оно чему програм приступа, а приказана вредност оно што се приказује у ComboBox-у. Метода UcitajUcenike() приказана је на слици 5.1.2.

Поред ових класа, у пројекту постоје и једноставне модел класе које служе за приказ података у контролама или табелама. Примери таквих класа су RangListaItem, IzborStavka, UcenikPrikaz, TakmicenjePrikaz и RadIzborStavka. Оне не садрже сложену логику. Већина ових класа су заправо под класе у класи UcitavanjePodataka и користе се за груписање података који се приказују кориснику. На пример, IzborStavka садржи идентификатор и назив ставке, па се лако користи у ComboBox контролама. Класа IzborStavka приказана је на слици 5.1.1.

```
15 references
public class IzborStavka
{
    6 references
    public int Id { get; set; }
    7 references
    public string Naziv { get; set; } = "";
}
```

Слика 5.1.1 Класа за ComboBox податке

```
5 references
public static List<IzborStavka> UcitajUcenike(bool UbaciOpcijuSviUcenici = false)
{
    List<IzborStavka> ucenici = new();

    if (UbaciOpcijuSviUcenici)
    {
        ucenici.Add(new IzborStavka { Id = 0, Naziv = "SVI UCENICI" });
    }

    using SQLiteConnection conn = DatabaseService.Connection();
    conn.Open();

    SQLiteCommand komanda = conn.CreateCommand();
    komanda.CommandText = @"SELECT u.id_ucenika, u.ime || ' ' || u.prezime || ' (' || u.odeljenje || ')' AS naziv
                            FROM ucenik u
                            JOIN generacija g ON g.id_generacije = u.id_generacije
                            WHERE g.skolska_godina = @skolska_godina
                            ORDER BY u.prezime, u.ime";
    komanda.Parameters.AddWithValue("@skolska_godina", GlobalnePromenljive.IzabranaGeneracija);

    using SQLiteDataReader citac = komanda.ExecuteReader();
    while (citac.Read())
    {
        ucenici.Add(new IzborStavka
        {
            Id = citac.GetInt32(0),
            Naziv = citac.GetString(1)
        });
    }

    return ucenici;
}
```

Слика 5.1.2 Метода за учитавање ученика из класе UcitavanjePodataka

5.2 Корисничке контроле

У апликацији су направљене и сопствене корисничке контроле.

Контрола PlaceHolderTextBox представља поље за унос текста са placeholder текстом. Placeholder је текст који се приказује док корисник није ништа унео, а крије када корисник крене да куца и служи као објашњење шта треба уписати у то поље. Ова контрола се користи на страницама за унос података, као што су додавање ученика, додавање такмичења и додавање рада. Њена предност је у томе што форма изгледа прегледније, јер корисник одмах види шта треба да унесе.

Контрола SearchComboBox направљена је као проширена верзија обичног ComboBox-а. Она омогућава да корисник не мора ручно да тражи ставку у дугој листи, већ може да куца део назива и тако филтрира резултате. Ово је посебно корисно код избора ученика, такмичења, предмета или истраживачких радова, јер тих података може бити много. Контрола има методе као што су Clear(), SelectFirst(), SelectItem(…), FilterItems(…) и OpenPopup(). Оне омогућавају брисање избора, избор прве ставке, ручно постављање изабране ставке, филтрирање листе и отварање падајућег прозора. Најважнија од ових метода је метода за претрагу, FilterItems(…). Битно је поменути и да је ова контрола направљена тако да има веома слична својства као контрола ComboBox, те поседује, на пример, својства SelectedValuePath и DisplayMemberPath која омогућавају да се за изабрани објекат у ComboBox-у једна вредност приказује на екрану (на пример име), а друга у програму (на пример ид). Ово је јако практично јер омогућује да се у коду и бази лако ради са ид-ем, док се кориснику приказују читљиви подаци. Поред тога, ова контрола имплементира и идеју контроле PlaceHolderTextBox, односно преко поља за унос текста има placeholder.

Већ је споменута контрола GeneracijaOdabir која се користи за избор активне генерације. Уз помоћ ње је избор генерације издвојен у посебну контролу, па главни прозор не мора директно да садржи сву логику за рад са генерацијама.

```
2 references
private void FilterItems(string searchText)
{
    string search = searchText.Trim();
    List<object> filteredItems = new(); *

    foreach (object item in allItems)
    {
        string displayText = GetDisplayText(item);
        if (search == "" || displayText.Contains(search, StringComparison.CurrentCultureIgnoreCase))
        {
            filteredItems.Add(item);
        }
    }

    lbItems.ItemsSource = filteredItems;
    lbItems.DisplayMemberPath = DisplayMemberPath;
}
```

Слика 5.2.1 Функција за филтрирање (претрагу) ставки у контроли SearchComboBox

6 СТРАНИЦЕ И ПОМОЋНИ ПРОЗОРИ

У овој апликацији коришћен је велик број страница и прозора. Странице су коришћене за главне функционалне целине апликације, док су прозори коришћени за поједине помоћне функционалности.

Странице се приказују унутар главног прозора, у контроли MainFrame, у зависности од тога коју опцију корисник изабере у менију. Уместо да свака функционалност отвара нови прозор, већина садржаја приказује се као посебна Page страница. На тај начин апликација остаје прегледна, а корисник се лако креће кроз различите делове програма.

Странице у апликацији могу се поделити у неколико група: странице за додавање података, странице за брисање података, странице за измену података, странице за приказ резултата и помоћне странице.

6.1 Странице за додавање података

Странице за додавање података користе се када корисник у базу уноси нове податке. У ову групу спадају DodajUcenika, DodajTakmicenje, DodajNagradu, DodajRad и DodajRadUceniku.

Ове странице имају сличан начин рада. Корисник попуњава поља на форми, након чега кликће на дугме Sacuvaj. Затим се у C# коду проверава да ли су унети подаци исправни. На пример, проверава се да ли су обавезна поља попуњена, да ли су бројеви у дозвољеном опсегу и да ли су изабрани потребни подаци из падајућих листи. Ако је унос исправан, подаци се уписују у базу помоћу параметризованог SQL INSERT упита. Ако унос није исправан, кориснику се приказује порука о грешци. Изглед једне овакве странице приказан је на слици 6.1.2.

Најважнија метода на овим страницама најчешће је btnSacuvaj_Click. Она се покреће када корисник кликне на дугме за чување. У њој се налази главни ток рада: читање података из форме, провера исправности, припрема SQL команде и упис у базу. Ова метода за једну од страница приказана је на слици 6.1.1.

Страница DodajUcenika омогућава додавање ученика у тренутно изабрану генерацију. Корисник уноси име, презиме, одељење и успех по годинама. Поред поља за успех

постоје и дугмад која аутоматски уписују вредност 5.00, како би се убрзао унос за ученике са овим успехом. Поени за активности се при додавању ученика подразумевано постављају на максималну вредност.

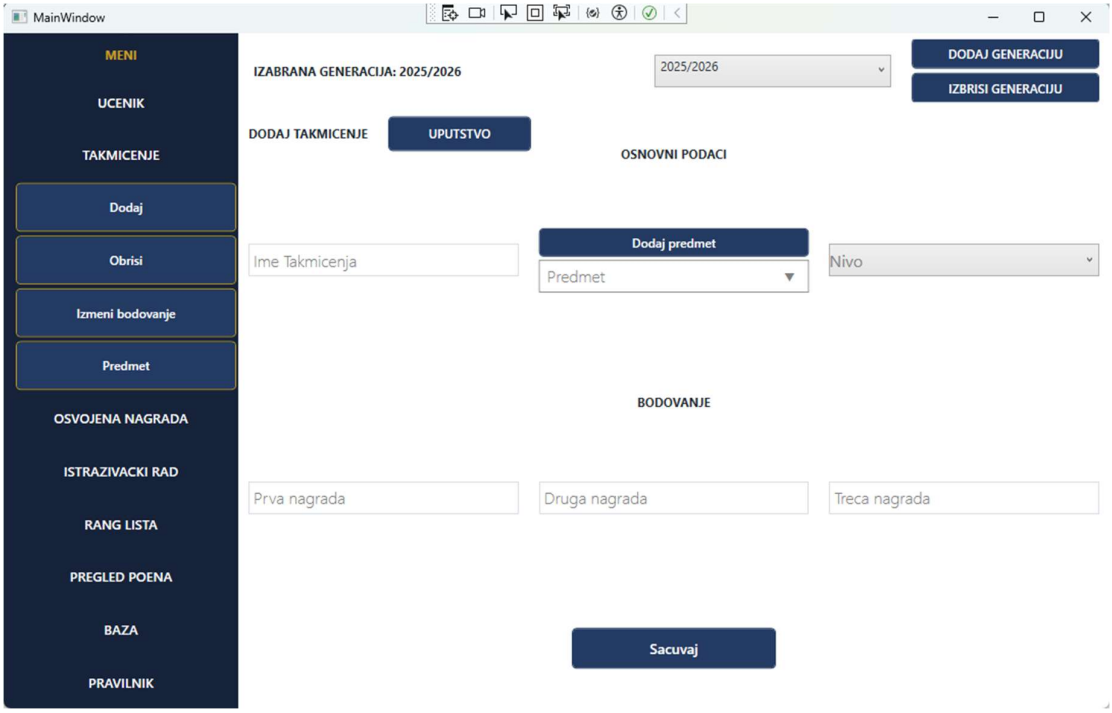
Страница DodajTakmicenje служи за додавање новог такмичења. Корисник уноси назив такмичења, бира предмет, бира ниво такмичења и уноси бодовање за прву, другу и трећу награду. Предмет се бира преко контроле SearchComboBox, што омогућава лакше проналажење предмета. Ако предмет не постоји, може се додати преко посебног прозора. При чувању такмичења уписују се подаци о такмичењу, али и бодовање за тренутно изабрану генерацију. Изглед странице DodajTakmicenje приказан је на слици 6.1.2.

Страница DodajNagradu омогућава додавање награда које су ученици освојили на такмичењима. Корисник бира ученика, такмичење, годину одржавања и освојену награду. Ученик и такмичење се оба бирају преко корисничке контроле SearchComboBox. Ова страница повезује ученика и такмичење преко табеле ucenik_takmicenje.

Страница DodajRad служи за додавање дефиниције истраживачког рада. Уносе се назив рада, област и број аутора. Страница DodajRadUceniku затим омогућава да се неки рад повеже са учеником. При томе се бира ученик, бира рад, приказује се број аутора рада и уноси се број поена који ученик добија.

```
1 reference
private void btnSacuvaj_Click(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrWhiteSpace(GlobalnePromenljive.IzabranaGeneracija))
    {
        MessageBox.Show("Prvo dodaj generaciju.");
        return;
    }

    if (cbUcenik.SelectedValue == null)
    {
        MessageBox.Show("Izaberi ucenika.", "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }
    if (cbTakmicenje.SelectedValue == null)
    {
        MessageBox.Show("Izaberi takmicenje.", "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }
    if (!int.TryParse(tbGodina.Text, out int godina) || godina < 1900 || godina > 2100)
    {
        MessageBox.Show("Unesi validnu godinu odrzavanja.", "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }
    if (cbNagrada.SelectedItem is not ComboBoxItem selectedNagrada || !int.TryParse(selectedNagrada.Content.ToString(), out int nagrada))
    {
        MessageBox.Show("Izaberi osvojenu nagradu.", "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    try
    {
        using SQLiteConnection conn = DatabaseService.Connection();
        conn.Open();

        SQLiteCommand komanda = conn.CreateCommand();
        komanda.CommandText = @"INSERT INTO ucenik_takmicenje (id_ucenika, id_takmicenja, godina_odrzavanja, osvojena_nagrada)
                                VALUES (@id_ucenika, @id_takmicenja, @godina_odrzavanja, @osvojena_nagrada)";
        komanda.Parameters.AddWithValue("@id_ucenika", cbUcenik.SelectedValue);
        komanda.Parameters.AddWithValue("@id_takmicenja", cbTakmicenje.SelectedValue);
        komanda.Parameters.AddWithValue("@godina_odrzavanja", godina);
        komanda.Parameters.AddWithValue("@osvojena_nagrada", nagrada);
        komanda.ExecuteNonQuery();

        MessageBox.Show("Nagrada je dodata.", "Uspesno", MessageBoxButton.OK, MessageBoxImage.Information);
        cbUcenik.Clear();
        cbTakmicenje.Clear();
        tbGodina.Text = "";
        cbNagrada.SelectedItem = null;
    }
    catch (SQLiteException ex) when (ex.ResultCode == SQLiteErrorCode.Constraint)
    {
        MessageBox.Show("Ovaj rezultat vec postoji ili podaci ne zadovoljavaju pravila baze.", "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

Слика 6.1.1 Метода чувања података у страници DodajNagradu



Слика 6.1.2 Изглед странице DodajTakmicenje

6.2 Странице за брисање података

Странице за брисање података користе се када корисник жели да уклони постојећи запис или везу из базе. У ову групу спадају ObrisiUcenika, ObrisiTakmicenje, ObrisiNagradu, ObrisiRad и ObrisiRadUceniku.

За ове странице је карактеристично да се подаци најчешће приказују у DataGrid контроли. Зато ове контроле користе методе учитавања за приказ из класе UcitavanjePodataka. Корисник изабере ред који жели да обрише, а затим кликне на дугме за брисање. Пре самог брисања апликација приказује поруку за потврду, како би се избегло случајно брисање података. Ако корисник потврди брисање, извршава се SQL DELETE команда, а затим се табела поново учитава.

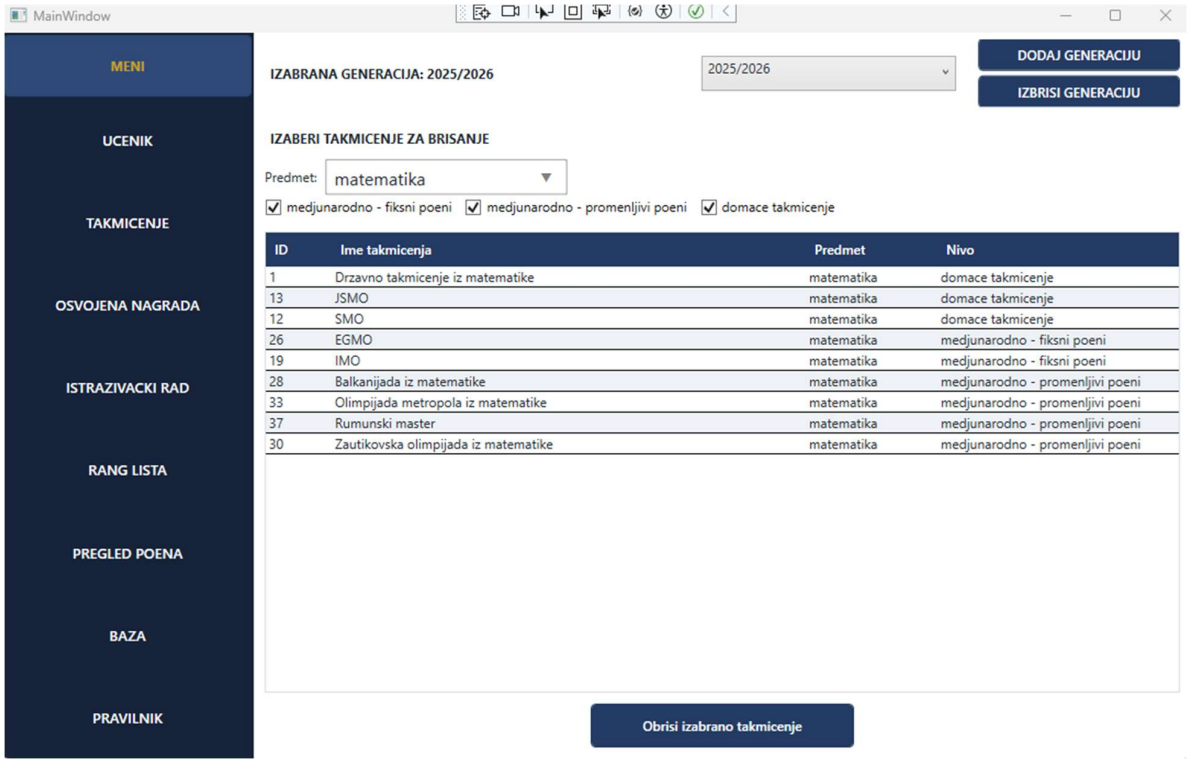
Страница ObrisiUcenika брише изабраног ученика. При брисању ученика потребно је обрисати и податке који су повезани са њим, као што су резултати на такмичењима и истраживачки радови ученика.

Страница ObrisiTakmicenje омогућава брисање такмичења. Она има филтере по предмету и нивоу такмичења, што је корисно јер у бази може постојати велики број такмичења. При брисању такмичења бришу се и повезани резултати ученика и

бодовање тог такмичења. Изглед ове странице приказан је на слици 6.2.1, док је метода за брисање такмичења приказана на слици 6.2.2.

Страница ObrisiNagradu служи за брисање освојене награде ученика. Она приказује резултате ученика на такмичењима и омогућава филтрирање по ученику и нивоу такмичења. Брисањем се уклања само конкретна веза између ученика и такмичења, а не сам ученик или такмичење.

Страница ObrisiRad брише дефиницију истраживачког рада, док ObrisiRadUceniku брише само везу између ученика и рада. Ово је важно јер један рад може постојати у бази као посебан податак, а ученици могу бити повезани са њим преко табеле ucenik_rad.



Слика 6.2.1 Изглед странице за брисање такмичења

```
1 reference
private void btnObrisi_Click(object sender, RoutedEventArgs e)
{
    if (dgTakmicenja.SelectedItem is not TakmicenjePrikaz takmicenje)
    {
        MessageBox.Show("Izaberi takmicenje za brisanje.", "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    MessageBoxResult odgovor = MessageBox.Show($"Da li zelis da obrises takmicenje: {takmicenje.ImeTakmicenja}?",
        "Potvrda brisanja", MessageBoxButton.YesNo, MessageBoxImage.Warning);
    if (odgovor != MessageBoxResult.Yes)
    {
        return;
    }

    using SQLiteConnection conn = DatabaseService.Connection();
    conn.Open();
    using SQLiteTransaction transakcija = conn.BeginTransaction();

    string[] queries =
    {
        "DELETE FROM ucenik_takmicenje WHERE id_takmicenja = @id_takmicenja",
        "DELETE FROM bodovanje_takmicenja WHERE id_takmicenja = @id_takmicenja",
        "DELETE FROM takmicenje WHERE id_takmicenja = @id_takmicenja"
    };

    foreach (string query in queries)
    {
        SQLiteCommand komanda = conn.CreateCommand();
        komanda.Transaction = transakcija;
        komanda.CommandText = query;
        komanda.Parameters.AddWithValue("@id_takmicenja", takmicenje.Id);
        komanda.ExecuteNonQuery();
    }

    transakcija.Commit();
    MessageBox.Show("Takmicenje je obrisano.", "Uspesno", MessageBoxButton.OK, MessageBoxImage.Information);
    UcitajTakmicenja();
}
```

Слика 6.2.2 Метода за брисање такмичења

6.3 Странице за измену података

Странице за измену података омогућавају промену већ постојећих вредности у бази. Странице у овој групи су IzmeniAktivnostUceniku, IzmeniBodovanjeTakmicenja и Baza.

Страница IzmeniAktivnostiUceniku омогућава измену поена за активности ученика по годинама. Корисник најпре бира ученика преко SearchComboBox контроле, након чега се приказују четири поља за унос активности по годинама. Вредности морају бити у дозвољеном опсегу од 1 до 10. Након чувања, подаци се уписују у колоне aktivnosti_1_god, aktivnosti_2_god, aktivnosti_3_god и aktivnosti_4_god у табели ucenik. Метода ове странице за упис промењених података приказана је на слици 6.3.1.

Страница IzmeniBodovanjeTakmicenja користи се за измену бодовања такмичења за тренутно изабрану генерацију. На страници се приказује табела у којој сваки ред представља једно такмичење, а корисник може да мења поене за прву, другу и трећу награду. Страница има филтере по предмету и нивоу такмичења. Важне методе

су UcitajBodovanje(), која учитава податке, FiltrirajStavku(..), која примењује филтере, ValidirajSveStavke(), која проверава исправност унетих поена, и SacuvajNagradu(…), која уписује измене у базу. Метода ValidirajSveStavke() приказана је на слици 6.3.2.

Страница Baza представља контролисани editor главних табела базе. Она омогућава преглед и измену података у табелама generacija, ucenik, takmicenje, predmet и istrazivacki_rad. Неке колоне су закључане за измену, као што су ID колоне, док се остале могу мењати. За чување измена користе се SQLiteDataAdapter и SQLiteCommandBuilder. Пре чувања се врши валидација података како би се спречио унос неисправних вредности.

```
1 reference
private void btnSacuvaj_Click(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrWhiteSpace(GlobalnePromenljive.IzabranaGeneracija))
    {
        MessageBox.Show("Prvo dodaj generaciju.");
        return;
    }

    if (cbUcenik.SelectedValue is not int idUcenika)
    {
        MessageBox.Show("Izaberi ucenika.", "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    if (!ValidanBrojPoena(tbAktivnosti1.Text, out double aktivnosti1)
        || !ValidanBrojPoena(tbAktivnosti2.Text, out double aktivnosti2)
        || !ValidanBrojPoena(tbAktivnosti3.Text, out double aktivnosti3)
        || !ValidanBrojPoena(tbAktivnosti4.Text, out double aktivnosti4))
    {
        MessageBox.Show("Aktivnosti za svaku godinu moraju biti broj od 1 do 10.", "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return;
    }

    using SQLiteConnection conn = DatabaseService.Connection();
    conn.Open();

    SQLiteCommand komanda = conn.CreateCommand();
    komanda.CommandText = @"UPDATE ucenik
                            SET aktivnosti_1_god = @aktivnosti_1_god,
                                aktivnosti_2_god = @aktivnosti_2_god,
                                aktivnosti_3_god = @aktivnosti_3_god,
                                aktivnosti_4_god = @aktivnosti_4_god
                            WHERE id_ucenika = @id_ucenika";
    komanda.Parameters.AddWithValue("@aktivnosti_1_god", aktivnosti1);
    komanda.Parameters.AddWithValue("@aktivnosti_2_god", aktivnosti2);
    komanda.Parameters.AddWithValue("@aktivnosti_3_god", aktivnosti3);
    komanda.Parameters.AddWithValue("@aktivnosti_4_god", aktivnosti4);
    komanda.Parameters.AddWithValue("@id_ucenika", idUcenika);
    komanda.ExecuteNonQuery();

    MessageBox.Show("Aktivnosti su izmenjene.", "Uspesno", MessageBoxButton.OK, MessageBoxImage.Information);
    ResetujFormu();
}

4 references
private bool ValidanBrojPoena(string tekst, out double brojPoena)
{
    return double.TryParse(tekst, out brojPoena) && brojPoena >= 1 && brojPoena <= 10;
}
```

Слика 6.3.1 Упис промењених података при мењању активности ученика

```
1 reference
private bool ValidirajSveStavke()
{
    foreach (BodovanjeTakmicenjaStavka stavka in stavke)
    {
        if (!ValidirajPoene(stavka.PrvaNagrada, stavka.ImeTakmicenja, "prvu nagradu", stavka.Nivo, 8, 18) ||
            !ValidirajPoene(stavka.DrugaNagrada, stavka.ImeTakmicenja, "drugu nagradu", stavka.Nivo, 7, 11) ||
            !ValidirajPoene(stavka.TrecaNagrada, stavka.ImeTakmicenja, "trecu nagradu", stavka.Nivo, 4, 8))
        {
            return false;
        }
    }

    return true;
}

3 references
private static bool ValidirajPoene(string vrednost, string imeTakmicenja, string nagrada, string nivo, int minimum, int maksimum)
{
    if (!int.TryParse(vrednost, NumberStyles.None, CultureInfo.InvariantCulture, out int brojPoena) || brojPoena < 0)
    {
        MessageBox.Show($"Unesi ceo broj poena veci ili jednak 0 za {nagrada}: {imeTakmicenja}.",
            "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return false;
    }

    if (nivo == NivoMedjunarodnoPromenljiviPoeni && (brojPoena < minimum || brojPoena > maksimum))
    {
        MessageBox.Show($"Za takmicenje {imeTakmicenja}, poeni za {nagrada} moraju biti od {minimum} do {maksimum}.",
            "Greska", MessageBoxButton.OK, MessageBoxImage.Error);
        return false;
    }

    return true;
}
```

Слика 6.3.2 Метода за проверу исправности измењених података при мењању бодовања такмичења

6.4 Странице за приказ резултата и података

Странице за приказ резултата и података не служе првенствено за унос података, већ за читање и анализу онога што је већ уписано у базу. Најважније странице у овој групи су RangLista, PregledPoena, Pravilnik, Pocetna и Baza.

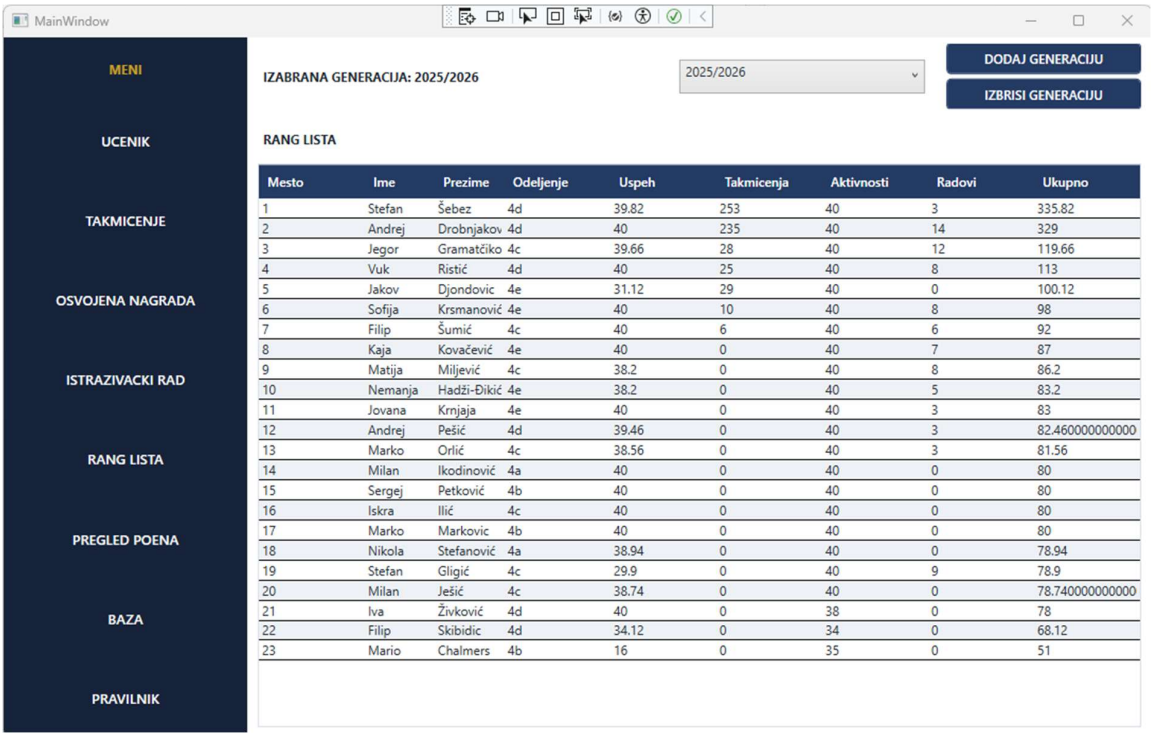
Страница RangLista приказује коначну ранг-листу ученика. Подаци се читају из погледа rang_lista, који у бази израчунава укупан број поена за сваког ученика. На овај начин апликација не мора ручно да рачуна све категорије поена у C# коду, већ само приказује резултат који је припремљен у бази. Изглед странице RangLista приказан је на слици 6.4.1.

Страница PregledPoena приказује детаљан преглед извора поена. Она омогућава да се види одакле потичу поени ученика: од такмичења, активности или истраживачких радова. Страница има филтер по ученику и филтере по категоријама, па корисник може одабрати да прикаже само оне податке који га тренутно интересују.

Страница Pravilnik приказује слику правилника, односно критеријума по којем се додељују поени. Она није везана за изабрану генерацију, па се при њеном отварању сакрива горњи избор генерације.

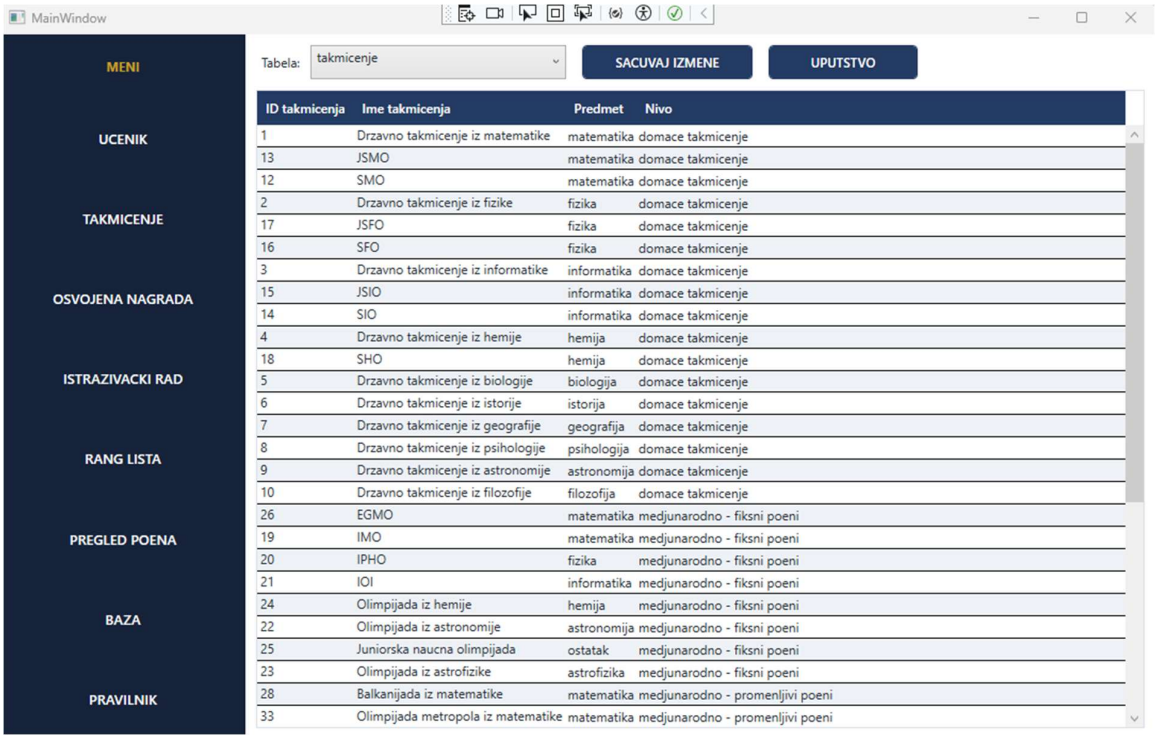
Страница Pocetna приказује почетни екран апликације. На њој се налази назив апликације и дугме за отварање упутства за коришћење. Кликом на дугме MENI у левом делу апликације корисник се враћа на ову страницу.

Страница Baza описана је у претходној подобласти, а њен изглед приказан је на слици 6.4.2.



Mesto	Ime	Prezime	Odeljenje	Uspeh	Takmicenja	Aktivnosti	Radovi	Ukupno
1	Stefan	Šebez	4d	39.82	253	40	3	335.82
2	Andrej	Drobnjakov	4d	40	235	40	14	329
3	Jegor	Gramatčiko	4c	39.66	28	40	12	119.66
4	Vuk	Ristić	4d	40	25	40	8	113
5	Jakov	Djondovic	4e	31.12	29	40	0	100.12
6	Sofija	Krsmanović	4e	40	10	40	8	98
7	Filip	Šumić	4c	40	6	40	6	92
8	Kaja	Kovačević	4e	40	0	40	7	87
9	Matija	Miljević	4c	38.2	0	40	8	86.2
10	Nemanja	Hadži-Đikić	4e	38.2	0	40	5	83.2
11	Jovana	Krnjaja	4e	40	0	40	3	83
12	Andrej	Pešić	4d	39.46	0	40	3	82.460000000000
13	Marko	Orlić	4c	38.56	0	40	3	81.56
14	Milan	Ikodinović	4a	40	0	40	0	80
15	Sergej	Petković	4b	40	0	40	0	80
16	Iskra	Ilić	4c	40	0	40	0	80
17	Marko	Markovic	4b	40	0	40	0	80
18	Nikola	Stefanović	4a	38.94	0	40	0	78.94
19	Stefan	Gligić	4c	29.9	0	40	9	78.9
20	Milan	Ješić	4c	38.74	0	40	0	78.740000000000
21	Iva	Živković	4d	40	0	38	0	78
22	Filip	Skibidic	4d	34.12	0	34	0	68.12
23	Mario	Chalmers	4b	16	0	35	0	51

Слика 6.4.1 Приказ ранг листе за базу са измишљеним подацима



Слика 6.4.2 Приказ странице Baza за табелу такмичење

6.5 Помоћне странице и прозори

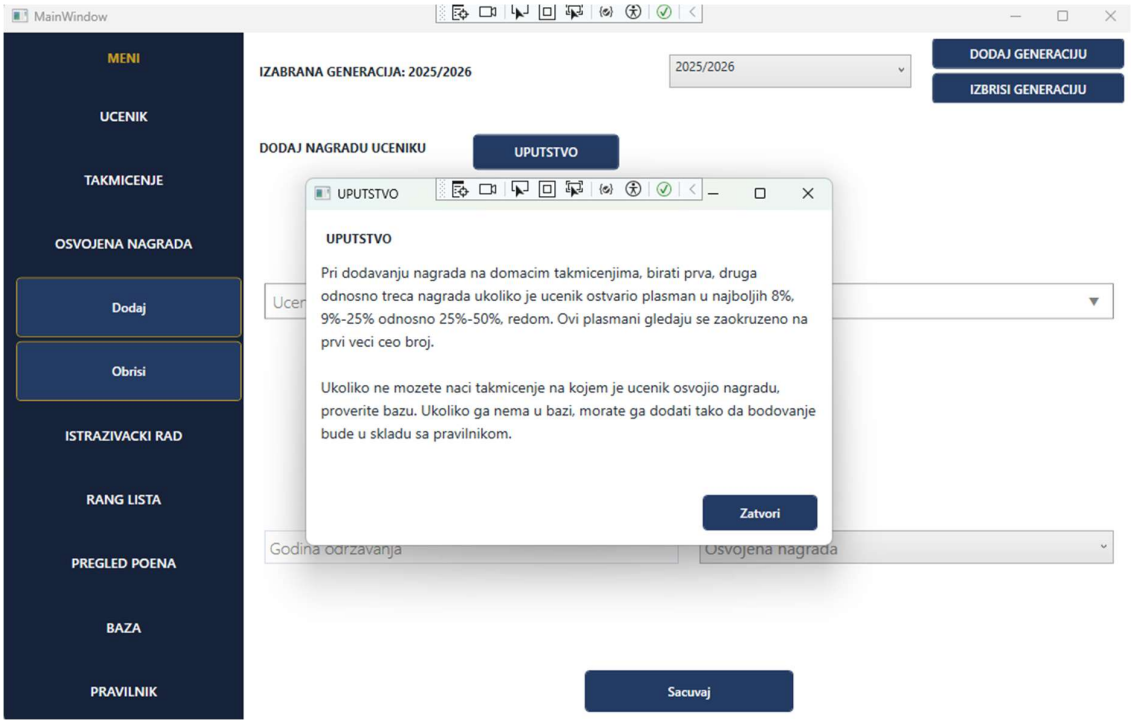
Поред главних страница, у апликацији постоје и помоћне странице и прозори. Они се користе за мање, али важне функционалности.

Страница Predmet служи за управљање предметима. Са ње корисник може да отвори прозор за додавање предмета или прозор за брисање предмета. Прозор DodajPredmetWindow омогућава унос новог предмета, док ObrisiPredmetWindow приказује предмете и број такмичења која их користе. Корисник бира предмет који жели да обрише. Ако неки предмет користи бар једно такмичење, брисање тог предмета није дозвољено.

Прозор DodajGeneracijuWindow користи се за додавање нове генерације. При додавању нове генерације апликација копира бодовање такмичења из претходне генерације која има дефинисано бодовање. То олакшава рад, јер корисник не мора ручно да уноси бодовање за сваку нову генерацију.

Прозор ObrisiGeneracijuWindow користи се за брисање генерације. Пошто брисање генерације може да уклони велики број повезаних података, овај прозор приказује број ученика у генерацији и тражи потврду пре брисања.

Прозор UputstvoWindow користи се за приказ дугачких упутстава. Уместо да се дугачак текст стално приказује на страници, корисник може да кликне на дугме UPUTSTVO и добије посебан прозор са објашњењем. Прозор за упутство на страници DodajNagradu приказан је на слици 6.5.1.



Слика 6.5.1 Помоћни прозор за упутство на страници за додавање награде

7 СМЕРНИЦЕ ЗА КОРИШЋЕЊЕ

Овај део рада намењен је корисницима апликације и независан је од остатка. У овом делу рада објашњене су основне функционалности апликације и предложен је начин коришћења. Смернице за коришћење могу се пронаћи на неколико места и унутар саме апликације кликом на дугмад са натписом *UPUTSTVO*.

При покретању апликације приказује се почетни екран са скраћеном верзијом упутства за коришћење. Са леве стране екрана је мени са избором страница које представљају главне функционалности.

Једна од страница у апликацији, PRAVILNIK, приказује критеријуме за бодовање. Препоручено је да корисник пре коришћења прочита ове критеријуме уколико није упознат са њима.

На врху апликације налази се избор генерације. Пре уноса података потребно је изабрати постојећу генерацију или додати нову генерацију кликом на дугме DODAJ GENERACIJU. Генерација представља школску годину за коју се врши бодовање ученика. Није препоручљиво додавати нове генерације унапред, јер се при додавању генерација копира бодовање такмичења из већ постојећих генерација. Ако додамо будућу генерацију и након тога за неку ранију додамо ново такмичење, будућа генерација неће имати податак о бодовању тог такмичења. Зато нову генерацију треба додавати тек када се крене са одређивањем ђака генерације за ту генерацију.

Након избора генерације потребно је унети ученике. Ученици се додају избором опције UCENIK > Dodaj у левом менију. У форми за додавање ученика уносе се име, презиме, одељење и успех по годинама школовања. Испод сваког места за унос просека постоји дугме које аутоматски на то место уноси просек 5.00. Након попуњавања података потребно је кликнути на дугме Sacuvaj. Ако су подаци исправно унети, ученик се уписује у базу података. По уносу, ученику се аутоматски уписује максималан број поена на активности.

Уколико ученик не треба да има максималан број поена на активности, потребно је изабрати опцију UCENIK > Izmeni aktivnosti. На овој страници бира се ученик и уносе се поени за активности по годинама. Након измене потребно је кликнути на дугме Sacuvaj, како би измене биле уписане у базу.

Освојене награде ученика додају се избором опције OSVOJENA NAGRADA > Dodaj. На овој страници потребно је изабрати ученика, такмичење, годину одржавања и освојену награду. Након уноса података потребно је кликнути на дугме Sacuvaj. Награде је могуће додати само за такмичења која постоје у бази. Ако у бази нема такмичења на ком је ученик освојио награду, потребно га је додати на испод описани начин.

Такмичења се додају избором опције TAKMICENJE > Dodaj. При додавању такмичења уноси се назив такмичења, бира се предмет, бира се ниво такмичења и уноси се бодовање за прву, другу и трећу награду. Уколико предмет не постоји у понуђеној листи, потребно је додати нови предмет преко дугмета Dodaj predmet. Треба обратити пажњу да се домаћа такмичења уносе са бројем поена који је прописан по правилнику.

Истраживачки радови уносе се кроз секцију ISTRAZIVACKI RAD. Најпре је потребно додати сам рад избором опције ISTRAZIVACKI RAD > Dodaj rad. Уносе се назив рада, област и број аутора. Након тога рад се повезује са учеником избором опције ISTRAZIVACKI RAD > Dodaj rad uceniku. На тој страници бирају се ученик и рад, а затим се уноси број поена који ученик добија за тај рад.

Уколико је потребно изменити бодовање такмичења, користи се опција TAKMICENJE > Izmeni bodovanje. На овој страници приказује се табела са такмичењима и бодовима за прву, другу и трећу награду. Вредности се мењају директно у табели. Након измене потребно је кликнути на дугме SACUVAJ IZMENE. Ако се страница напусти пре чувања, апликација приказује упозорење о несачуваним изменама.

За брисање података користе се одговарајуће опције у менију. Ученици се бришу преко UCENIK > Obrisi, такмичења преко TAKMICENJE > Obrisi, а освојене награде преко OSVOJENA NAGRADA > Obrisi. Предмети се бришу преко TAKMICENJE > PREDMET>Obrisi Predmet. Истраживачки радови и везе између ученика и радова бришу се кроз секцију ISTRAZIVACKI RAD. При брисању података апликација приказује поруку за потврду, како би се избегло случајно брисање.

Резултати се проверавају преко страница RANG LISTA и PREGLED POENA. Страница RANG LISTA приказује коначну ранг-листу ученика за изабрану генерацију. Страница PREGLED POENA приказује детаљан преглед извора поена,

односно приказује на основу чега је ученик добио поене. Ове две странице треба користити за проверу тачности унетих података и коначног резултата.

Страница BAZA служи за контролисан преглед и измену главних података у бази. На овој страници могу се прегледати и мењати подаци из основних табела, понуђених у ComboBoxu на врху странице. Нека поља могу се изменити двокликом на одговарајућу ћелију, док нека поља није дозвољено мењати. Након измене потребно је кликнути на дугме SACUVAJ IZMENE, јер у супротном измене неће бити уписане у базу.

Препоручени редослед коришћења апликације је следећи: прво погледати правилник, затим изабрати или додати генерацију, додати ученике, потом унети освојене награде, додавајући успут такмичења која фале, додати истраживачке радове и повезати их са ученицима, проверити/изменити бодовање такмичења, а затим прегледати странице RANG LISTA и PREGLED POENA.

На крају је потребно поново проћи кроз податке и проверити да ли су сви правилно унети. То је најлакше урадити кроз странице PREGLED POENA и BAZA. У апликацији је имплементиран велики број мера за спречавање уноса недозвољених података, али су и даље могуће грешке, поготову словне или граматичке.

8 ЗАКЉУЧАК

Израдом ове апликације остварен је главни циљ рада: направљен је програм који процес избора ђака генерације чини организованијим, прегледнијим и мање зависним од ручне обраде података.

Као резултат рада добијена је функционална WPF десктоп апликација са локалном SQLite базом података. Апликација омогућава избор активне генерације, унос и измену података, преглед извора поена, приказ правилника и формирање коначне ранг-листе. Апликација омогућава поуздан рад са подацима, јер су у програму уведене провере унетих вредности, потврде пре брисања и упозорења у случају несачуваних измена.

Током рада показало се да су за израду оваквог типа апликације најважнији кораци планирање структуре базе података и планирање структуре апликације. То је зато што структура базе одређује скуп потенцијалних функционалности које се могу направити, док добра структура апликације омогућава поделу кода на целине и значајно олакшава имплементацију.

Упркос томе, током израде апликације више пута су се јавиле потребе за изменом почетног плана. На пример, неки подаци су касније другачије организовани у бази, додате су нове функционалности и побољшан је начин приказа и измене података. Ово показује да је развој апликације постепен процес и да се решење често унапређује када се боље уоче стварне потребе корисника. Добро планирање не спречава могућност за касније промене, већ само смањује број промена које су потребне.

Иако апликација испуњава основни циљ, постоје могућности за њено даље унапређење. Једно од најважнијих побољшања било би увођење сервера на коме би се налазила база података. Тренутно се база чува локално у фајлу, што је једноставно за употребу, али отежава усклађивање података и чини да само један корисник у једном тренутку може да ради на бази. Проблем усклађивања је тренутно решен преко GitHub репозиторијума, што свакако није идеално, јер подразумева да све промене мануелно уноси админ. Серверска база би омогућила да више корисника истовремено приступа истим подацима и да се све то аутоматски усклађује, што би било веома корисно за овај пројекат.

Поред тога, апликацију је потребно користити и тестирати кроз наредних неколико бодовања ђака генерације. Тек кроз стварну употребу могу се уочити додатни недостаци, нејасноће у корисничком интерфејсу или места на којима би рад могао бити бржи и једноставнији. На основу тих запажања апликација би се могла даље прилагођавати потребама школе.

Овај рад показује како се комбинацијом WPF апликације, C# програмског језика и SQLite базе података може направити практичан програм који решава конкретан школски проблем.

ЛИТЕРАТУРА

[1] В. Благојевић, Релационе базе података 1, ИЦНТ, Београд, 2006

[2] С. Наков, Fundamentals of Computer Programming with C#, Фабер, Велико Тарново, Бугарска, 2013

[3] Windows Presentation Foundation for .NET documentation | Microsoft Learn (What is Windows Presentation Foundation - WPF | Microsoft Learn) прегледано 25.5.

[4] https://en.wikipedia.org/ (https://en.wikipedia.org/wiki/Extensible_Application_Markup_Language) прегледано 25.5.

[5] https://sqlite.org/ прегледано 25.5.

[6] https://medium.com/ (https://medium.com/@pmmanav/classes-in-c-113885bcd85b) прегледано 26. 5.

[7] https://www.c-sharpcorner.com/ (https://www.c-sharpcorner.com/article/understanding-classes-and-types-of-classes-in-c-sharp-a-complete-guide/) прегледано 26.5.

[8] https://www.mg.edu.rs/ (https://www.mg.edu.rs/uploads/files/dokumenta/pravno/Pravilnik%20o%20nagradjivanju%20ucenika%20(1).pdf) прегледано 27.5.