Aplikacija za savjetovanje u strateškim igrama

Rajčevac, Matija

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:348568

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Preddiplomski sveučilišni studij Računarstvo

APLIKACIJA ZA SAVJETOVANJE U STRATEŠKIM IGRAMA

Završni rad

Matija Rajčevac

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 16.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Matija Rajčevac
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3975, 21.10.2019.
OIB studenta:	39917699794
Mentor:	Izv. prof. dr. sc. Ivica Lukić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Aplikacija za savjetovanje u strateškim igrama
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	16.09.2020.
Datum potvrde ocjene Odbora:	23.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 19.09.2020.

Ime i prezime studenta:	Matija Rajčevac
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3975, 21.10.2019.
Turnitin podudaranje [%]:	14

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za savjetovanje u strateškim igrama**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Ivica Lukić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD……1

1. Zadatak završnog rada……1

2. RAZVOJNO OKRUŽENJE……2

1. .NET Framework……2
2. C# programski jezik……3
3. SQL ……4

1. Naredbe SQL jezika……...5

3. RAZVOJ APLIKACIJE……..7

1. Teamfight Tactics……7
2. Analiza problema……8
3. Kreiranje E-R i relacijskog modela……9
4. Logičko oblikovanje baze podataka……10

1. Tablica „Champion“……..10

2. Tablica „Ability“……11

3. Tablica „Stats“……...11

4. Tablica „Item“……12

5. Tablica „Class“……..12

6. Tablica „Origin“……13

5. Popunjavanje baze podataka……...13
6. Kreiranje Windows Form Aplikacije……..15

4. KORIŠTENJE APLIKACIJE……23

5. ZAKLJUČAK……...26

LITERATURA……27

SAŽETAK……...28

ABSTRACT……29

PRILOZI……30

1. UVOD

U današnje vrijeme upravljanje velikim skupinama podataka se može pronaći u svakoj tvrtki, mnogim aplikacijama te u svakom području gdje je potrebna koordinacija između nekih korisnika ili spremanje i korištenje intelektualnih vrijednosti. Za to se koriste organizirani skupovi podataka s kojima je lagano baratati ili takozvane baze podataka.

Ima jako puno definicija baze podataka. Definiciju koja najbolje opisuje rješenje problema kojeg rješavamo u ovom radu je rekao James Martin: „Baza podataka je skup istovrsnih podataka s višestrukom namjenom. Korisnik nije zainteresiran za sve vrste podataka u bazi, već samo za one koji su mu potrebni u njegovom poslu. Korisnik može imati uvid u samo jednu, njemu potrebnu datoteku koja ima uvijek istu i to vrlo jednostavnu strukturu, iako je u biti izvedena iz mnogo kompleksnije strukture podataka. Različiti korisnici uzimaju u obzir različite datoteke izvedene iz iste baze podataka. Dakle, iako je baza podataka zajednička većem broju korisnika, različiti korisnici je različito shvaćaju.“[1]

Ovim radom će se pokazati jedan od načina na koji se baze podataka mogu u kombinaciji sa windows aplikacijom koristiti kako bi olakšali korisniku određeni zadatak, odnosno davati mu samo one informacije koje korisnik u tome trenutku treba.

Ovaj rad se sastoji od pet poglavlja. U prvom poglavlju je uvod koji govori koji je zadatak ovog rada. U drugom poglavlju obraduje se teorijski dio teme, drugim riječima u kojem se razvojnom okruženju piše aplikacija i općenite informacije i potrebna predznanja za pisanje same aplikacije. U trećem poglavlju naveden je praktički dio teme, detaljniji opis problema kojeg aplikacija rješava, proces kreiranja baze i objašnjenje koda. U četvrtom poglavlju objašnjen je način korištenja aplikacije. U petom poglavlju je naveden zaključak cjelokupnog rada.

1.1. Zadatak završnog rada

U radu je potrebno izraditi veliku bazu podataka i aplikaciju koja će korištenjem tih podataka olakšati korisniku igranje određene strateške igre. U ovom slučaju ćemo kao igru uzeti u obzir

[1] [1] J. Martin, Computer Data-base Organization

Teamfight Tactics od izdavača Riot Games. Za izradu baze podataka koristiti će se SQL jezik dok će se za izradu windows aplikacije koristiti C#.

2. RAZVOJNO OKRUŽENJE

Za kreiranje aplikacije koristio se program Microsoft Visual Studio 2019. Taj program daje mogućnost kreiranja Windows Form aplikacije korištenjem .NET platforme.

2.1. .NET Framework

.NET okvir je dio .NET platforme. .NET okvir (Framework) je proizašla iz Microsofta, te je to razvojni okvir koji nam daje novo sučelje za programiranje aplikacija. Radi se o posebnoj infrastrukturi koja je programerima olakšala, a samim time i ubrzala razvoj aplikacija svih vrsta i oblika tako što nudi gotova rješenja i funkcionalnosti. .NET podržava pokretanje web stranica, usluga, desktop aplikacije i puno više na Windows uređajima.

Dvije najznačajnije komponente .NET okvira jesu takozvani Common Language Runtime ili skraćeno CLR i Class biblioteka.

- CLR je izvršni mehanizam koji obrađuje pokrenute aplikacije. Pruža usluge poput upravljanja nitima, sakupljanja smeća, sigurnost tipova, rukovanje iznimkama itd.
- Class Library nudi skup API-ja i tipova za uobičajene funkcije. Pruža vrste za nizove, datume, brojeve itd. Class Library uključuje API-je za čitanje i pisanje datoteka, povezivanje sa bazama podataka, crtanje i još mnogo toga.

.NET aplikacije napisane su na programskom jeziku C #, F # ili Visual Basic-u. Kod se kompajlira u jezično-agnostički zajednički srednji jezik (CIL). Sastavljeni kod pohranjuje se u sklopove - datoteke s nastavkom datoteke .dll ili .exe.

Također se sastoji od skupa .NET Enterprise servera (koji obuhvaća SQL server). Za nas je to bitno pošto nam to omogućuje pisanje aplikacije korištenjem C# programerskog jezika i kreiranje baze podataka s kojom će navedena aplikacija raditi. [2]

2.2. C# programski jezik

C# (C - Sharp) je programski jezik razvijen od strane Microsofta koji se pokreče na .NET Framework-u. Koristi se za razvijanje web aplikacija, desktop aplikacija, mobilnih aplikacija, igrica i puno više. C# je u potpunosti objektno orijentiran jezik, što znači da je svaki podatak predstavljen kao objekt neke klase.

C# je veoma jednostavan programski jezik, pogodan za nove korisnike, te uz jednostavnost nudi sigurnost, brzinu, objektnu-orijentiranost i usmjerenost prema internetu.

Osnovna sintaksa C# jezika slična je sintaksi ostalih jezika u stilu C kao što su C, C++ i Java. Kao na primjer:

- Točka-zarez se koristi za označavanje kraja iskaza.
- Vitičaste zagrade koriste se za grupiranje izjava.
- Izjave su obično grupirane u metode (funkcije), metode u klase i klase u namespace-ove
- Varijable se dodjeljuju pomoću znaka jednakosti, ali u usporedbi s dva uzastopna znaka jednakosti.
- Uglate zagrade koriste se s nizovima kako za njihovo deklariranje tako i za dobivanje vrijednosti prema zadanom indeksu u jednom od njih.

Neke značajne značajke C# jezika koje ga razlikuju od C, C++ i Jave su:

- Prijenosnost - specifikacija jezika ne navodi zahtjeve za generiranjem koda za prevoditelj: to jest, ne navodi da kompajler C # mora ciljati vrijeme izvođenja zajedničkog jezika ili generirati zajednički srednji jezik (CIL) ili generirati bilo koji drugi specifični format.
- Metaprogramiranje - Metaprogramiranje putem atributa C # dio je jezika. Mnogi od ovih atributa dupliciraju funkcionalnost GCC-ovih i VisualC++-ovih direktiva pretprocesora ovisnih o platformi.
- Polimorfizam - Za razliku od C ++, C # ne podržava višestruko nasljeđivanje, iako klasa može implementirati bilo koji broj sučelja. Ovo je odluka dizajnera vodećeg arhitekta za izbjegavanje komplikacija i pojednostavljivanje arhitektonskih zahtjeva u čitavom CLI-u. [3]

2.3. SQL

Strukturni upitni jezik (engl. *Structured Query Language*) ili skraćeno SQL, je programski jezik visoke razine. Jedan od najpopularnijih programskih jezika za izradu, traženje, ažuriranje i brisanje podataka iz relacijskih baza podataka.

Osobine SQL-a:

- Jezik sličan engleskom. Kao dio naredbe upotrebljava riječi kao što su: SELECT, INSERT, DELETE i slično.
- Ne-proceduralni jezik.
- U jednom trenutku obrađuje skup slogova.
- Jednostavan, drugim riječima mogu ga koristiti korisnici različitih profila, profesionalni programeri i neprofesionalni korisnici.
- Osigurava naredbe za rješavanje raznih zadaća:
    - Upite nad podacima
    - Dodavanje, mijenjanje, brisanje redaka u tablicama
    - Kreiranje, mijenjanje, brisanje objekata sheme
    - Kontrolu pristupa bazi podataka i objektima sheme
    - Osigurava konzistentnost baze podataka

Objekti u SQL-u su: Baza podataka, tablica, pogled ili virtualna tablica, pravilo integriteta, indeks, procedura, okidač i sinonim. [4]

Bitne osobine sintakse za napomenuti:

- SQL ne razlikuje velika i mala slova, sljedeće dvije linije rade isto.
    - select * From STUDENT where GODINA= 'prva'
    - SELECT * FROM student WHERE godina= 'prva'
- Uobičajena je praksa da se ključne riječi pišu velikim slovima, a atributi i imena tablica malim slovima(kao što je kod druge točke napisano).
- U vrijednostima koje unosimo velika i mala slova se razlikuju, tako da iduće nije isto
    - SELECT * FROM student WHERE godina = 'prva'
    - SELECT * FROM student WHERE godina = 'PRVA'

2.3.1. Naredbe SQL jezika

SQL DML - (engl. *Data Manipulation Language*)

Koriste se za dohvaćanje, pohranu, promjenu i brisanje podataka u bazi.

- SELECT - dohvaćanje podataka iz baze
- UPDATE - izmjena postojećih podataka
- DELETE - brisanje postojećih podataka
- INSERT - dodavanje novih podataka [4]

SQL DDL - (engl. *Data Definition Language*)

Koristi se za definiciju objekata u bazi: kreiranje i izmjenu strukture objekata.

- CREATE - kreiranje objekata baze
- DROP - uklanjanje objekata baze
- ALTER - izmjena definicije objekata baze
- GRANT - definiranje prava pristupa podacima
- REVOKE - uklanjanje definicije prava pristupa podacima [4]

Za kreiranje naše aplikacije biti će nam potrebne naredbe SELECT, INSERT, UPDATE, CREATE.

Sintaksa naredbe CREATE TABLE:

```
CREATE TABLE  ime_tablice
        ("ime_stupca1" tip_podataka_stupca1,
         "ime_stupca2" tip_podataka_stupca2,
         ... )
```

Sintaksa naredbe INSERT:

```
INSERT [ INTO]
{ table_name | view_name }
{ [ ( column_list ) ]
        { VALUES
                ( { DEFAULT | NULL | expression } [ ,...n] )
                        | derived_table
        }
} | DEFAULT VALUES
```

Sintaksa naredbe UPDATE:

```
UPDATE { table_name | view_name }
 SET { column_name = { expression | DEFAULT | NULL } } [,...n]
{ { [ FROM { < table_source > [ ,...n ] ]
        [ WHERE < search_condition > ] } ] }
```

Sintaksa naredbe SELECT

```
SELECT [ ALL | DISTINCT ] select_list
[ INTO new_table ]
[ FROM table_source ]
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression  [ ASC | DESC ] ]
```

3. RAZVOJ APLIKACIJE

U ovom poglavlju naveden je problem kojeg naša aplikacija mora riješiti, te postupak kreiranja te aplikacije. Kako bi znali definirati problem prvo se moramo upoznati sa igricom za koju pravimo „savjetnika“.

3.1. Teamfight Tactics

Slika 3.1. Snimka zaslona unutar TFT-a

Najjednostavnije rečeno Teamfight Tactics ili skraćeno TFT je strateška igrat zasnovana na rundama gdje vas suprotstavlja protiv sedam protivnika u utrci da izgradite najmoćniju ekipu da se bori u vaše ime. Vaš cilj: Budite zadnja osoba koja stoji.

Kako igrati: Koristite Novčiće (označeno žutom bojom) između svake runde za kupnju novih prvaka (engl. *Champion*) koji vam se nasumice ponude (označeno zelenom bojom). Na lijevoj strani (označeno ružičastom) piše koliko je efekta i koji su efekti aktivni, svaki prvak ima klasu (engl. *Class*) i podrijetlo (engl. *Origin*) koji daju određene efektu ukoliko ima dovoljan broj prvaka iste klase i podrijetla. I sa desne strane (označeno crvenom) vide se ostali protivnici i koliko života(eng. Health)

imaju. Svaku rundu se vaš tim bori protiv protivničkog i osoba koja izgubi gubi dio života. Kada život padne na 0, igrač ispada iz runde. Runda traje dok ne ostane samo jedna osoba.

3.2. Analiza problema

Dobar tim se sastoji od prvaka koji imaju aktivne efekte, koji su visoka razina i na kojima su izgrađena odgovarajuća stavka (engl. item). Pošto je ponuda prvaka i stavki nasumična treba igraču dati informacije koji prvaci imaju sinergiju sa kojima i koje stavke na njima imaju velik utjecaj.

Zadatak je napraviti aplikaciju koja ovisno o prvaku kojeg korisnik ima ispisuje bitne informacije o samom prvaku, te također koji drugi prvaci imaju sinergiju sa njim i koje stavke odgovaraju tom prvaku.

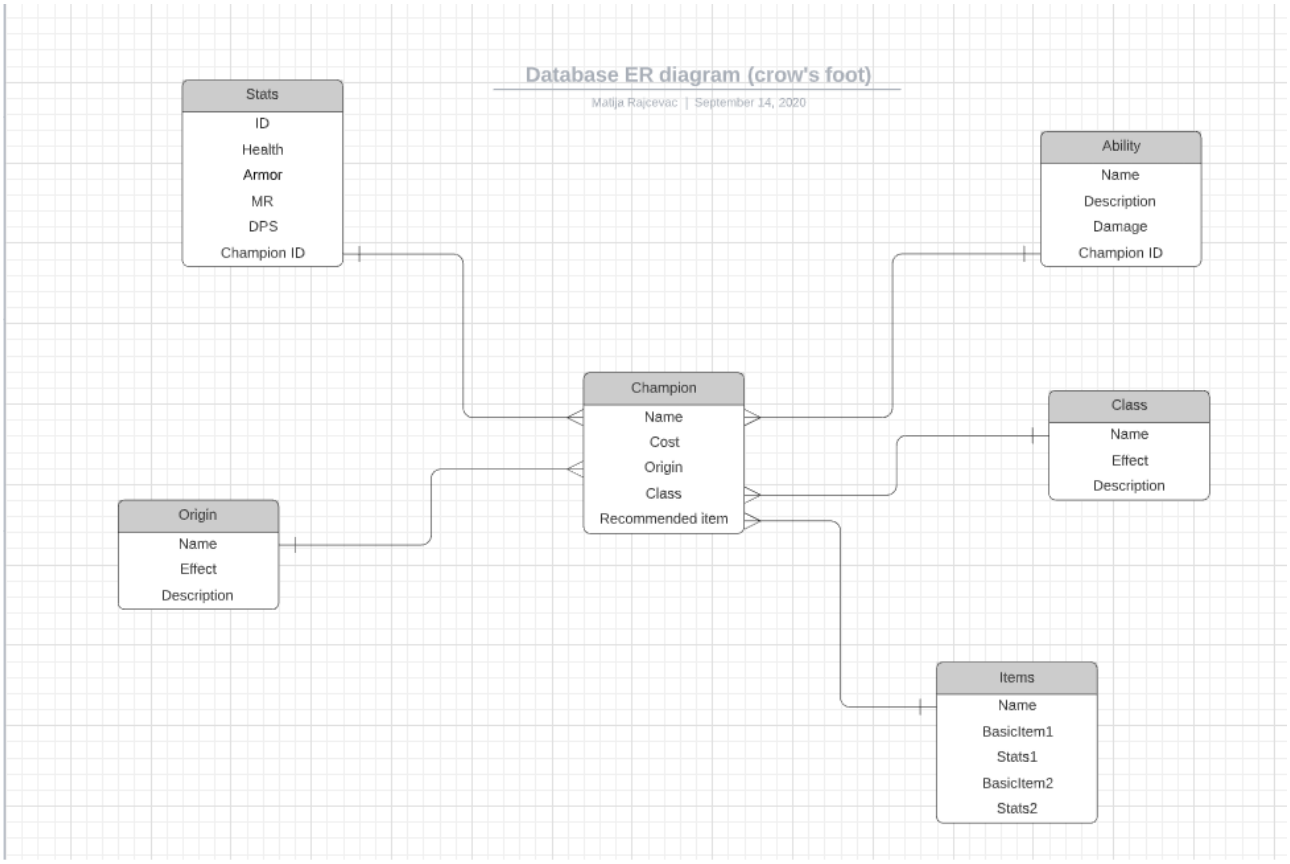
Slika 3.2. Ponuđene kartice prvaka

Svaki prvak ima svoje ime, koliko novčića košta, sposobnost (engl. *Ability*), karakteristike (engl. *Stats*), klasu, podrijetlo i preporučene stavke. Također je bitno uz ime sposobnosti naglasiti koliko štete (engl. *Damage*) radi i opis same sposobnosti, a karakteristike se sastoje od atributa Health, Armor, Magic Resist i Damage per Second. Klasa i podrijetlo imaju efekt koji daju i koliko je prvaka potrebno da aktivirat taj efekt. Svaka stavka je izgrađena od dvije temeljne stavke koje imaju svoje karakteristike.

Te informacije su bitno znanje koje korisnik treba imati kako bi izgradio najmoćniju ekipu i porazio sve protivnike. Kako bi sve to mogli spremiti i ispisati u aplikaciji potrebno je napraviti i popuniti bazu podataka.

3.3. Kreiranje E-R i relacijskog modela

Analizom problema možemo napraviti E-R dijagram koji govori o odnosu atributa i veze.



Slika 3.3. E-R dijagram baze podataka

Preslikavanje ER modela u relacijski model:

CHAMPION (<u>Name</u>, Cost, Origin, Class, Recommended Items)

ABILITY (<u>Name</u>, Description, Damage, Champion ID)

STATS (<u>ID</u>, Health, Armor, MR, DPS, Champion ID)

ORIGIN (Name, Description, Effect)

CLASS (Name, Description, Effect)

ITEM (ID, BasicItem1, Stats1, BasicItem2, Stats2)

3.4. Logičko oblikovanje baze podataka

U ovom poglavlju prikazano je kreiranje i logički izgled baze podataka u Microsoft Visual Studio-u.

3.4.1. Tablica „Champion“

Name	Data Type	Allow Nulls	Default
Name	varchar(50)	☐	
Origin	varchar(50)	☑	
Class	varchar(50)	☑	
RecommendedItem1	varchar(50)	☑	
RecommendedItem2	varchar(50)	☑	
RecommendedItem3	varchar(50)	☑	
Cost	varchar(50)	☑	
		☐	

Design | T-SQL

```
CREATE TABLE [dbo].[Champion] (
    [Name]            VARCHAR (50) NOT NULL,
    [Origin]          VARCHAR (50) NULL,
    [Class]           VARCHAR (50) NULL,
    [RecommendedItem1] VARCHAR (50) NULL,
    [RecommendedItem2] VARCHAR (50) NULL,
    [RecommendedItem3] VARCHAR (50) NULL,
    [Cost]            VARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([Name] ASC),
    FOREIGN KEY ([Origin]) REFERENCES [dbo].[Origin] ([Name]),
    FOREIGN KEY ([Class]) REFERENCES [dbo].[Class] ([Name])
);
```

Slika 3.4. Tablica "Champion"

Tablica „Champion“ sadrži atribute Name, Cost, RecommendedItem, te dva strana ključa Origin i Class koji povezuju tablicu „Champion“ sa tablicama „Origin“ i „Class“.

3.4.2. Tablica „Ability“

Name	Data Type	Allow Nulls	Default
Name	varchar(50)	☐	
Description	varchar(MAX)	☑	
Damage	varchar(50)	☑	
ChampionID	varchar(50)	☑	
		☐	

Design | T-SQL

```
CREATE TABLE [dbo].[Ability] (
    [Name]        VARCHAR (50)  NOT NULL,
    [Description] VARCHAR (MAX) NULL,
    [Damage]      VARCHAR (50)  NULL,
    [ChampionID]  VARCHAR (50)  NULL,
    PRIMARY KEY CLUSTERED ([Name] ASC),
    FOREIGN KEY ([ChampionID]) REFERENCES [dbo].[Champion] ([Name])
);
```

Slika 3.5. Tablica "Ability"

Tablica „Ability“ sadrži atribute Name, Description, Damage i strani ključ Champion ID koji povezuje tablicu „Ability“ sa tablicom „Champion“.

3.4.3 Tablica „Stats“

Name	Data Type	Allow Nulls	Default
StatsID	int	☐	
Health	nchar(20)	☑	
Armor	nchar(10)	☑	
MR	nchar(10)	☑	
Dps	nchar(20)	☑	
ChampionID	varchar(50)	☑	
		☐	

Design | T-SQL

```
CREATE TABLE [dbo].[Stats] (
    [StatsID]    INT           NOT NULL,
    [Health]     NCHAR (20)    NULL,
    [Armor]      NCHAR (10)    NULL,
    [MR]         NCHAR (10)    NULL,
    [Dps]        NCHAR (20)    NULL,
    [ChampionID] VARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([StatsID] ASC),
    FOREIGN KEY ([ChampionID]) REFERENCES [dbo].[Champion] ([Name])
);
```

Slika 3.3. Tablica "Stats"

Tablica „Stats“ sadrži atribute Health, Armor, Magic Resist (MR) i Damage per Second (DPS), te strani ključ Champion ID koji ju povezuje sa tablicom „Champion“.

3.4.4. Tablica „Item“

Id	varchar(50)	☐
Item1	varchar(50)	☑
Stat1	varchar(20)	☑
Item2	nvarchar(50)	☑
Stat2	varchar(20)	☑
		☐

Design | T-SQL

```
CREATE TABLE [dbo].[Item] (
    [Id]    VARCHAR (50)  NOT NULL,
    [Item1] VARCHAR (50)  NULL,
    [Stat1] VARCHAR(20)     NULL,
    [Item2] NVARCHAR (50) NULL,
    [Stat2] VARCHAR(20)     NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Slika 3.4. Tablica "Item"

Tablica „Item“ sadrži imena dvije temeljne stavke i njihove karakteristike.

3.4.5. Tablica „Class“

Name	Data Type	Allow Nulls	Default
Name	varchar(50)	☐	
Description	varchar(MAX)	☑	
Effect	varchar(250)	☑	
		☐	

Design | T-SQL

```
CREATE TABLE [dbo].[Class] (
    [Name]        VARCHAR (50)  NOT NULL,
    [Description] VARCHAR (MAX) NULL,
    [Effect]      VARCHAR (250) NULL,
    PRIMARY KEY CLUSTERED ([Name] ASC)
);
```

Slika 3.5. Tablica "Class"

Tablica „Class“ sadrži atribute Name, Description i Effect.

3.4.6. Tablica „Origin“

Name	Data Type	Allow Nulls	Default
Name	varchar(50)	☐	
Description	varchar(MAX)	☑	
Effect	varchar(50)	☑	
		☐	

Design | T-SQL

```
CREATE TABLE [dbo].[Origin] (
    [Name]        VARCHAR (50)  NOT NULL,
    [Description] VARCHAR (MAX) NULL,
    [Effect]      VARCHAR (50)  NULL,
    PRIMARY KEY CLUSTERED ([Name] ASC)
);
```

Slika 3.6. Tablica "Origin"

Tablica „Origin“ sadrži atribute Name, Description i Effect.

3.5. Popunjavanje baze podataka

Aplikacija od korisnika ne traži nikakav unos podataka, već svi podaci s kojima baza radi se upisuju od strane programera. Ovakav pristup rješavanja ovog problema je pogodan pošto se igrica konstantno ažurira i podaci se mijenjaju kako bi bila igrica ravnoteži, a ažuriranje same baze podataka je vrlo jednostavan i brz proces.

Popunjavanje tablica u bazi podataka se vrši naredbom INSERT.

```sql
INSERT INTO Champion
VALUES
(
    'Ahri',
    'Star Guardian',
    'Sorcerer',
    'Blue Buff',
    'Rabadons Deathcap',
    'Morellonomicon',
    '2C'
);
INSERT INTO Ability
VALUES
(
    'Orb of Deception',
    'Ahri fires an orb in a line dealing magic damage to enemies it passes through.
    It then returns to her, dealing true damage to all enemies it passes through.',
    '175 / 250 / 425',
    'Ahri'
);
INSERT INTO Stats
VALUES
(
    '1',
    '550 / 990 / 1782',
    '40',
    '20',
    '32 / 57 / 102',
    'Ahri'
);
```

Slika 3.7. INSERT naredba za popunjavanje jednog entiteta u tablicama Champion, Ability i Stats.

```sql
INSERT INTO Class
VALUES
(
    'Sorcerer',
    'All Alies gain incresed Spell Power',
    '2 20% Spell Power
     4 40% Spell Power
     6 70% Spell Power'
);
INSERT INTO Origin
VALUES
(
    'Star Guardian',
    'Star Guardians spellcasts grant mana to other Star Guardians (spread among them).',
    '3 - 15 Mana
     6 - 25 Mana
     9 - 60 Mana'
);
INSERT INTO Item
VALUES
(
    'Blue Buff',
    'Tear of the Goddess',
    '+15 Mana',
    'Tear of the Goddess',
    '+15 Mana'
);
```

Slika 3.8. INSERT naredba za popunjavanje jednog entiteta u tablicama Class, Origin, Item

Ukoliko želimo zamijeniti podatke u postojećim slogovima tablice koristimo naredbu UPDATE.

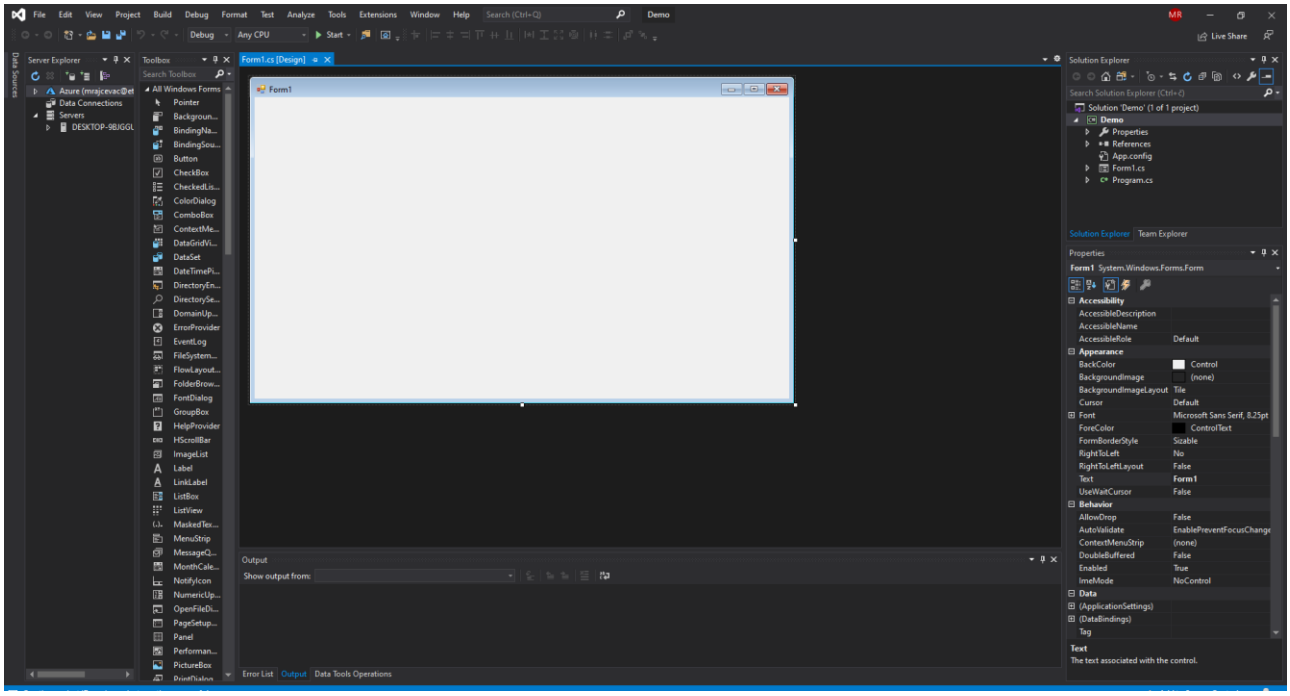
```
UPDATE Ability
SET Damage = '175 / 250 / 425'
WHERE ChampionID = 'Ahri';
```

Slika 3.9. UPDATE naredba

Sa ovom naredbom mijenjamo atribut Damage u tablici Ability gdje je ChampionID = „Ahri“.

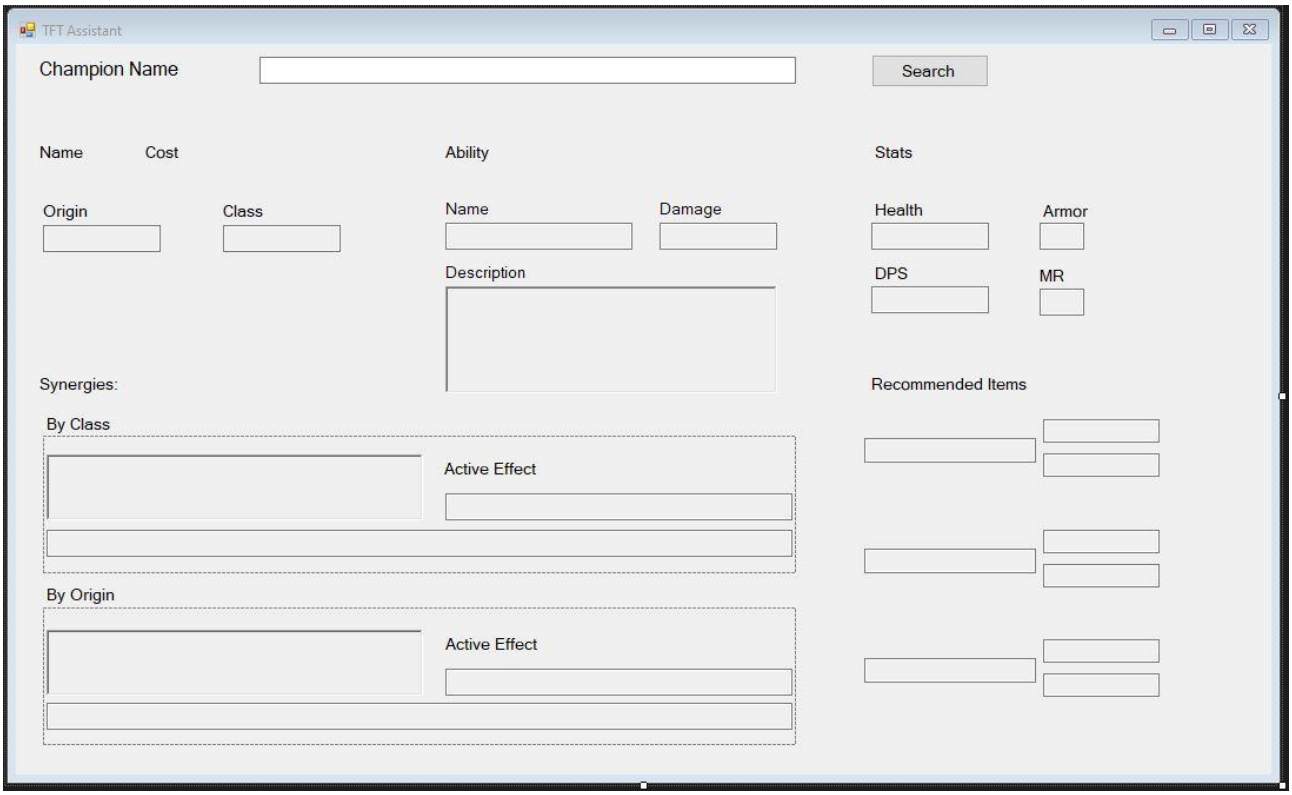
3.6. Kreiranje Windows Form Aplikacije

Pri programiranju koristio se program Microsoft Visual Studio 2019, a kod se pisao u C# programskom jeziku.



Slika 3.10. Izgled početnog Windows Form App sučelja

Prvi korak razvoja aplikacije je bio dodavanje svih potrebnih alata na obrazac koji će nam dati mogućnost pretraživanja baze podataka te ispisivati sve potrebne podatke iz te baze podataka.



Slika 3.11. Izgled aplikacije nakon dodanih potrebnih alata

Alati se u formu postavljaju principom povlačenja i ispuštanja (engl. Drag and Drop).

Od alata su se koristili text box-ovi od kojih je jedan korišten za upisivanje imena prvaka kojeg želimo pretražiti u bazi, a ostali su korišteni za ispisivanje informacija o tome prvaku, oni su postavljeni samo za čitanje. Ostatak teksta je ispisan pomoću label-a. Oni nam samo kazuju koji točno podataka vidimo u text box-u. Te imamo jedan button pomoću kojega ćemo pokretati pretraživanje baze podataka.

Idući korak razvoja aplikacije bio je pisanje koda s kojim će aplikacija pretraživati bazu i ispisivati podatke u određena mjesta.

```
Form1.cs   Form1.cs [Design]
Demo                                   Demo.Form1
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10
11  namespace Demo
12  {
        3 references
13      public partial class Form1 : Form
14      {
            1 reference
15          public Form1()
16          {
17              InitializeComponent();
18          }
19      }
20  }
21
```

Slika 3.12. Početni izgled koda aplikacije

Aplikacija nešto radi kada se stisne dugme Search i kada je u searchTextBox-u upisano ime prvaka.

```
1 reference
private void searchBtn_Click(object sender, EventArgs e)
{
    if (searchTextBox.Text == null)
    {
        throw new System.ArgumentException("Parameter cannot be null", "searchTextBox");
    }
```

Sav kod se piše u metodi searchBtn_Click. Te izbacuje iznimku(eng. *Exception*) ukoliko je searchTextBox prazan.

```
SqlConnection conn = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Matija\source\
                                         repos\TFT_Assistant\TFT_Assistant\Database1.mdf;Integrated Security=True");
conn.Open();
```

Slika 3.13. Uspostavljanje konekcije

Idući korak je bio uspostavljanje konekcije sa bazom podataka, to se vrši sa naredbom conn.Open(); a na kraju metode se konekcija mora zatvoriti sa conn.Close();

Nakon što je konekcija otvorena, može se početi pretraživanje baze podataka, za to nam je potrebna SQL naredba koju inicijaliziramo sa „new SqlCommand“ u kojoj kao parametre koristimo naredbu SELECT koja odabire one podatke iz baze koji nama trebaju i konekciju nad bazom podataka.

```
SqlCommand cmd = new SqlCommand("select Name, Cost, Origin, Class, RecommendedItem1, RecommendedItem2, RecommendedItem3 " +
    "from Champion where Name = '" + searchTextBox.Text + "'", conn);
SqlDataReader reader = cmd.ExecuteReader();
while (reader.Read())
{
    lbChampionName.Text = reader.GetValue(0).ToString();
    lbCost.Text = reader.GetValue(1).ToString();
    tbOrigin.Text = reader.GetValue(2).ToString();
    tbClass.Text = reader.GetValue(3).ToString();
    tbItem1.Text = reader.GetValue(4).ToString();
    tbItem2.Text = reader.GetValue(5).ToString();
    tbItem3.Text = reader.GetValue(6).ToString();
}
reader.Close();
```

Slika 3.14. Čitanje podataka iz tablice Champion

Ovim linijama koda kreiramo novu SQL naredbu, koja će kada se pokrene iz baze odabrati atribute Name, Cost, Origin, Class, RecommendedItem1, 2 i 3 iz tablice „Champion“ u kojem je Name jednak tekstu kojeg mi upisujemo u searchTextBox.

Također kreiramo čitača kao SqlDataReader koji čita te podatke iz tablica i preko njega koristimo naredbu GetValue kako bi dohvatili te podatke i ispisali ih u odgovarajuće text-box-ove na grafičkom sučelju aplikacije.

Nakon što čitač završi sa čitanjem moramo ga ugasiti sa reader.Close();

Ovim principom čitamo i ispisujemo iz svih tablica u njihove odgovarajuće text-box-ove.

```
SqlCommand abilityCmd = new SqlCommand("select Name, Description, Damage from Ability where ChampionID = '" + searchTextBox.Text + "'", conn);
reader = abilityCmd.ExecuteReader();
while (reader.Read())
{
    tbAbilityName.Text = reader.GetValue(0).ToString();
    tbAbilityDamage.Text = reader.GetValue(2).ToString();
    tbAbilityDescription.Text = reader.GetValue(1).ToString();
}
reader.Close();

SqlCommand statsCmd = new SqlCommand("select Health, Armor, MR, Dps from Stats where ChampionID = '" + searchTextBox.Text + "'", conn);
reader = statsCmd.ExecuteReader();
while (reader.Read())
{
    tbHealth.Text = reader.GetValue(0).ToString();
    tbArmor.Text = reader.GetValue(1).ToString();
    tbMR.Text = reader.GetValue(2).ToString();
    tbDPS.Text = reader.GetValue(3).ToString();
}
reader.Close();
```

Slika 3.15. Čitanje podataka iz tablica "Ability" i "Stats"

U ovim linijama koda kreiramo dvije nove SQL naredbe sa kojima čitamo podatke iz tablice „Ability“ i tablice „Stats“ te ih pomoću čitača dohvaćamo i upisujemo u odgovarajuće text-box-ove.

```
SqlCommand championsByClassCmd = new SqlCommand("select Name from Champion where Class = '" + tbClass.Text + "' and Name != '" + searchTextBox.Text + "'", conn);
reader = championsByClassCmd.ExecuteReader();
while (reader.Read())
{

    tbChampionsByClass.Text += reader.GetValue(0).ToString() + "     ";

}
reader.Close();

SqlCommand classCmd = new SqlCommand("select Description, Effect from Class where Name ='" + tbClass.Text + "'", conn);
reader = classCmd.ExecuteReader();
while (reader.Read())
{
    tbClassDescription.Text = reader.GetValue(0).ToString();
    tbClassEffect.Text = reader.GetValue(1).ToString();
}
reader.Close();
```

Slika 3.16. Čitanje podataka iz tablice "Class" i svih Championa koji su istog Class-a kao traženi Champion

Prvom SQL naredbom odabiremo sva imena prvaka koji pripadaju istoj klasi kao traženi prvak te ih sa čitačem čitamo i onda sve ispisujemo u jedan text-box.

Drugom SQL naredbom odabire Description i Effect iz tablice Class koja odgovara našem traženom prvaku.

```
SqlCommand championsByOriginCmd = new SqlCommand("select Name from Champion where Origin = '" + tbOrigin.Text + "' and Name != '" + searchTextBox.Text + "' ", conn);
reader = championsByOriginCmd.ExecuteReader();
while (reader.Read())
{

    tbChampionsByOrigin.Text += reader.GetValue(0).ToString() + "     ";

}
reader.Close();
SqlCommand originCmd = new SqlCommand("select Description, Effect from Origin where Name ='" + tbOrigin.Text + "'", conn);
reader = originCmd.ExecuteReader();
while (reader.Read())
{
    tbOriginDescription.Text = reader.GetValue(0).ToString();
    tbOriginEffect.Text = reader.GetValue(1).ToString();
}
reader.Close();
```

Slika 3.17. Čitanje podataka iz tablice "Origin" i svih Championa koji su istog Origin-a kao traženi Champion

Prvom SQL naredbom odabiremo sva imena prvaka koji pripadaju istom podrijetlu kao traženi prvak te ih sa čitačem čitamo i onda sve ispisujemo u jedan text-box.

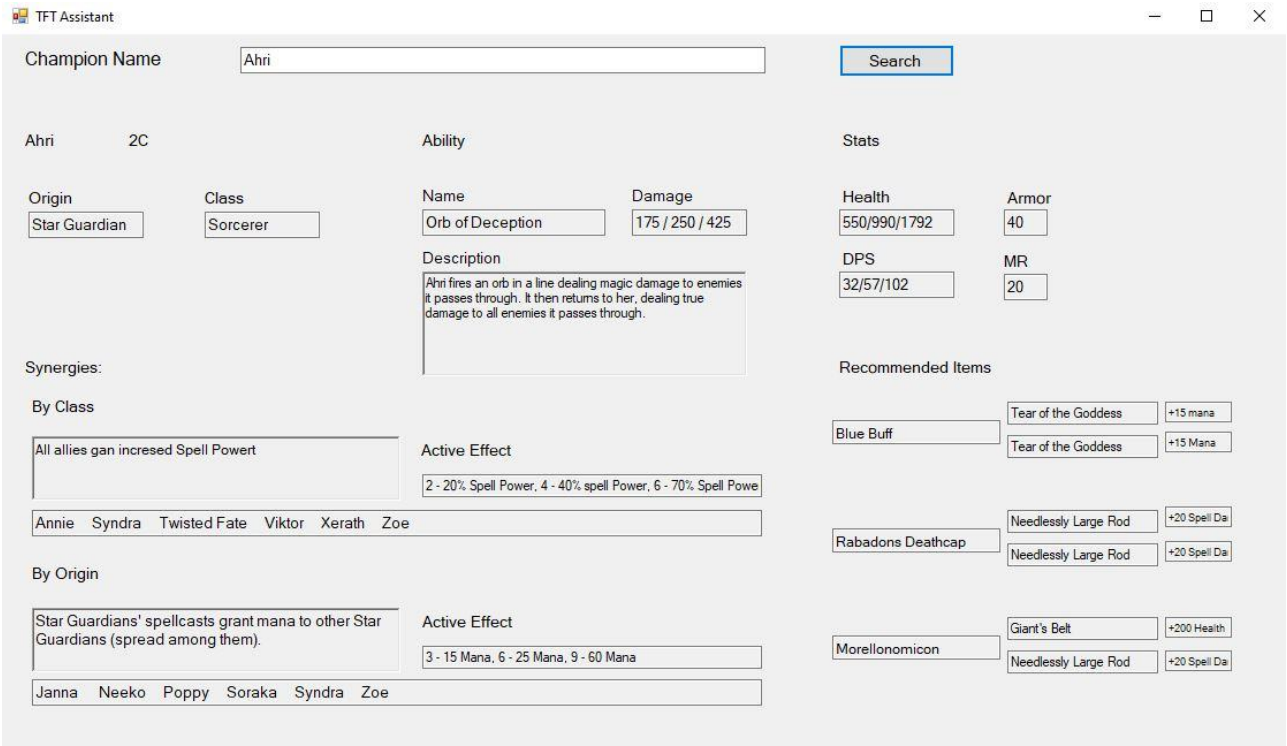
Drugom SQL naredbom odabire Description i Effect iz tablice Origin koja odgovara našem traženom prvaku.

```
SqlCommand ItemCmd = new SqlCommand("select Item1, Stat1, Item2, Stat2 from Item where Id = '" + tbItem1.Text + "' ", conn);
reader = ItemCmd.ExecuteReader();
while (reader.Read())
{
    tbItem11.Text = reader.GetValue(0).ToString();
    tbStat11.Text = reader.GetValue(1).ToString();
    tbItem12.Text = reader.GetValue(2).ToString();
    tbStat12.Text = reader.GetValue(3).ToString();
}
reader.Close();
```

Slika 3.18. Čitanje podataka o predloženim Item-ima

Ovom SQL naredbom odabiremo podatke iz tablice „Item“ i ispisujemo u odgovarajuće text box-ove.

I za kraj moramo zatvoriti konekciju sa bazom sa conn.Close(); i zatvaramo metodu searchBtn_Click sa vitičastom zagradom.



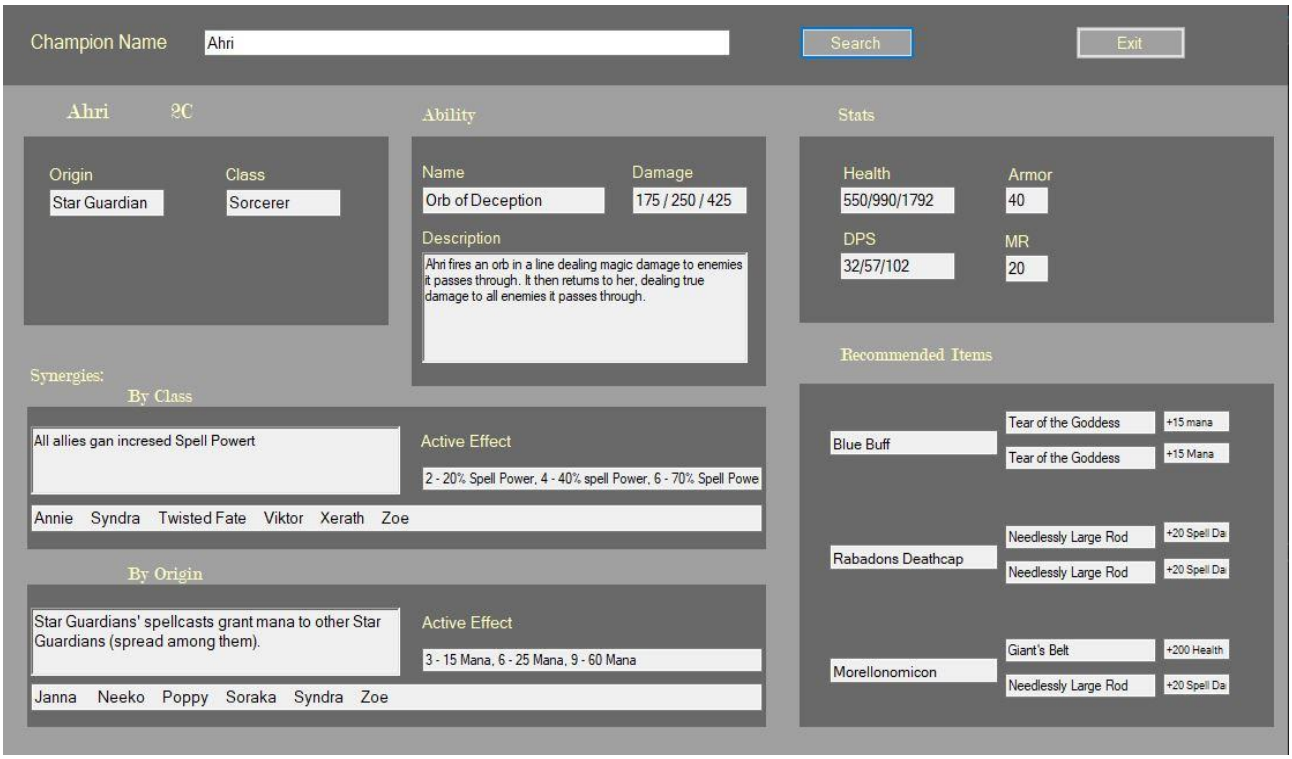
Slika 3.19. Izgled aplikacije nakon pretraživanja

Aplikacija pretražuje i ispisuje podatke iz baze podataka kako je bilo i zamišljeno. Aplikacija je funkcionalna ali izgled same aplikacije nije estetski lijep, tako da je idući korak, uređivanje samog izgleda grafičkog sučelja aplikacije.

Kako bi aplikacija izgledala modernije dodano je dugme za izlaz, a obrub je maknut.

```
1 reference
private void exitBtn_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are You Sure To Exit Programme ?", "Exit", MessageBoxButtons.OKCancel) == DialogResult.OK)
    {
        Application.Exit();
    }
}
```

Slika 3.20. dodavanje dugmeta za izlaz iz aplikacije



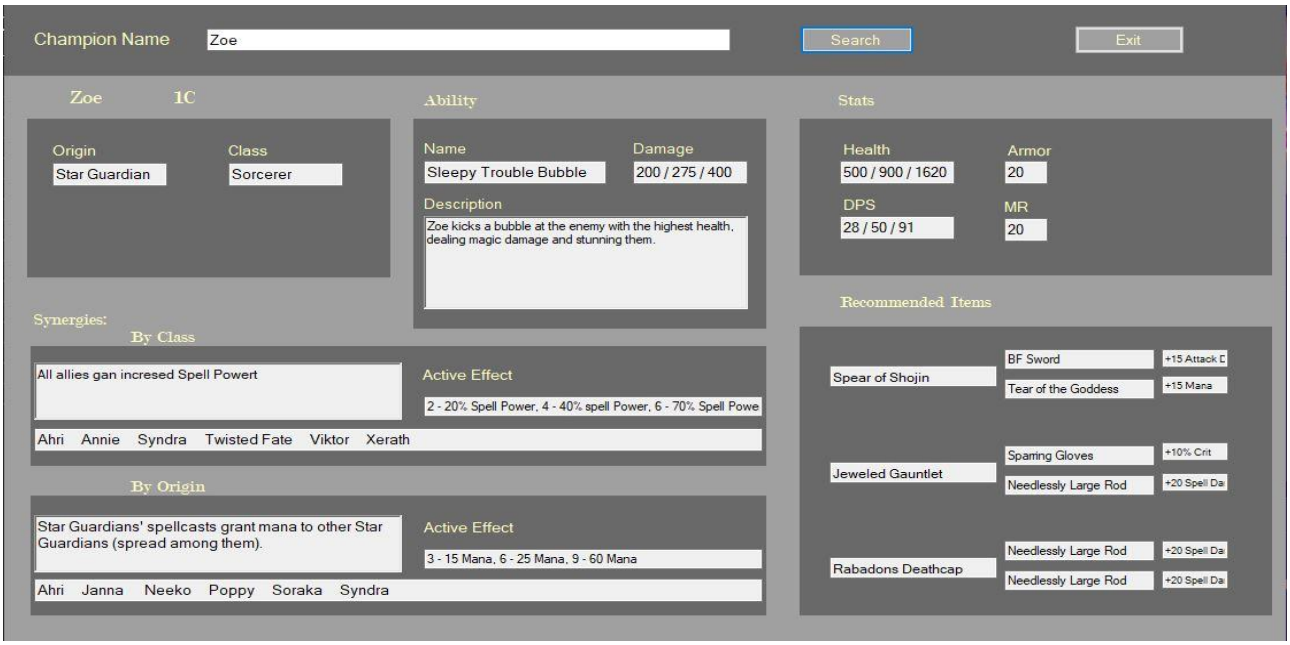
Slika 3.21. Izgled dovršene aplikacije

4. KORIŠTENJE APLIKACIJE

Korisnik u isto vrijeme pokreće partiju TFT-a i aplikaciju. Na početku runde, svih 8 igrača je smješteno na takozvani vrtuljak (eng. *Carousel*), na kojem su nasumično ponuđenih 9 prvaka, gdje igrači biraju prvaka s kojim započinju rundu.

Slika 4.1. Vrtuljak

Slika 4.2. Snimka zaslona unutar igrice



Slika 4.3. Aplikacija

U ovom slučaju kao početni prvak izabrana je „Zoe“, u traku za pretraživanje upisujemo njeno ime i aplikacija nam ispisuje osnovne podatke o njoj, te što nam je bitnije ispisuje koje stavke bi korisnik trebao na njoj graditi i od kojih temeljnih stavki se grade. Također nam ispisuje koji ostali prvaci imaju sinergiju sa njom i koji efekti će biti aktivni ako dodamo i te prvake na polje

Slika 4.4. Snimka zaslona unutar igrice

Tokom trajanja cijele partije, korisnik može držati aplikaciju u pozadini i kada ga zanimaju informacije o određenom prvaku, samo je potrebno da upiše ime prvaka u traku za pretraživanje i stisne Search dugme. Kada partija završi ili ako mu jednostavno više ne treba savjetnik izlazi iz aplikacije stiskom na dugme Exit. Strategijske igrice zahtijevaju jako dobro znanje same igrice i svih dijelova koje ta igrica nudi, upravo to ova aplikacija olakšava. Omogućava korisniku bolji pregled igre, lakše snalaženje po pitanju šta i kako, te samim time poboljšava korisnikovo iskustvo tijekom igranja igrice, jer tko voli gubiti?

5. ZAKLJUČAK

U uvodu je navedeno kako kod većine aplikacija imamo potrebu korisniku osigurati bazu podataka u koju će korisnik spremati svoje podatke ili iz koje će korisnik izvlačiti podatke koji su mu potrebni. Ovim završnim radom prikazano je kako korištenjem tehnologija kao što su .NET Framework, C# i SQL na vrlo jednostavan i brz način upravo taj problem možemo realizirati. Ova aplikacija nam nudi mogućnost izvlačenja intelektualnih vrijednosti iz baze podataka. Korisniku daje informacije o prvaku iz igrice Teafight Tactic za vrijeme partije kako bi mu olakšala igranje same igrice, a samim time i poboljšala iskustvo. Ukoliko postoji potreba za ažuriranjem podataka u bazi, taj dio obavlja programer.

LITERATURA

[1] James Martin, Computer Data-base Organization, izdavač: Prentice Hall, godina izdanja: 1977

[2] What is .NET Framework? - Microsoft 2020

https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework (datum pristupanja 14. 9. 2020.)

[3] C# documentation - Microsoft 2020

https://docs.microsoft.com/hr-hr/dotnet/csharp (datum pristupanja 14. 9. 2020.)

[4] Rahul Barta, SQL Primer: An Accelerated Introduction to SQL Basics, izdavač: Apress, godina izdanja 2018.

SAŽETAK

U ovom radu je prikazana izrada Windows Form Aplikacije po koracima, koja korisniku daje informacije koje on treba znati za prvaka kojeg pretražuje. Opisano je razvojno okruženje u kojem se razvijala aplikacija. Također je ukratko opisana igrica za koju gradimo aplikaciju, Teamfight Tactics. Aplikacija je jednostavna za korištenje, te ima prostora za nadogradnju i implementaciju novih mogućnosti. Korisnik nema pristupa bazi podataka iz koje aplikacija izvlači podatke, već ukoliko baza zahtjeva nadogradnju ili izmjenjivanje podataka to radi programer. Za razvoj aplikacije koristio se program Microsoft Visual Studio 2019, te je kod pisan korištenjem C# programerskog jezika te SQL Server lokalna baza podataka.

Ključne riječi: baza podataka, C# jezik, SQL, Teamfight Tactics, Windows Form aplikacija

ABSTRACT

Title: Application for consulting in strategy games

This paper presents the creation of a Windows Form Application step by step, which gives the user the information he needs to know about the Champion he is searching for. The development environment in which the application was developed is described. It also briefly describes the game we are building the app for, Teamfight Tactics. The application is easy to use and has room for upgrades and implement new features. The user does not have access to the database from which the application extracts data, it is developers' job if the database requires an upgrade or changes of data. Microsoft Visual Studio 2019 was used to develop the application, and the code was written using the C # programming language and SQL Server local database.

Keywords: databases, C# programming language, SQL, Teamfight Tactics, Windows Form Application

PRILOZI

Aplikacija, kompletan kod aplikacije i lokalna baza podataka se može pronaći na CD-u u datoteci TFT_Assistant.