

**Tehnička škola Čakovec**

**ELABORAT**

**ZAVRŠNOG RADA**

**PRORAČUNSKE TABLICE**

|  |  |
| --- | --- |
| **Mentor:** | **Učenik:** |
| Krešimir Kočiš, prof. | Aleksandar Toplek |

**Čakovec, svibanj 2012.**

Ova stranica je namijerno ostavljena prazna.

|  |
| --- |
|  |

**Tehnička škola Čakovec**

**Prosudbeni odbor za završni rad**

Učenik: **Aleksandar Toplek**

Razred: **4. ET2**

Školska godina: **2011./2012.**

Obrazovno područje: **Računalstvo**

Naziv zadatka: **PRORAČUNSKE TABLICE**

Opis zadatka:

Potrebno je kreirati aplikaciju za tablično računanje. Aplikacija, osim tabličnog računanja unesenih podataka, treba omogućavati i rad s formulama, oblikovanje teksta, oblikovanje obruba, više stranica po dokumentu, izradu grafova te ispis. Koristiti tehnologije C# i Windows Presentation Foundation.

Učenik će se za konzultacije obratiti svojem mentoru.

|  |  |  |
| --- | --- | --- |
| Zadatak zadan: | Rok predaje pisanog rada: | Predviđeni datum obrane: |
| **30. studenoga 2011.** | **25. svibnja 2012.** | **4. – 9. lipanja 2012.** |

Mentor:

Krešimir Kočiš, prof.

Ova stranica je namijerno ostavljena prazna.

Sadržaj

[Ideja projekta 1](#_Toc325361331)

[Korištene tehnologije 2](#_Toc325361332)

[C# (C Sharp) 2](#_Toc325361333)

[XAML (Extensible Application Markup Language) 3](#_Toc325361334)

[WPF (Windows Presentation Foundation) 3](#_Toc325361335)

[MEF (Managed Extensibility Framework) 4](#_Toc325361336)

[Razvojni alati 5](#_Toc325361337)

[Sublime Text 2 5](#_Toc325361338)

[Microsoft Expression Blend 4 6](#_Toc325361339)

[Visual Studio 11 Beta 6](#_Toc325361340)

[Metro Tables 12](#_Toc325361341)

[Uvod 12](#_Toc325361342)

[Pokretanje aplikacije 12](#_Toc325361343)

[Početna stranica 14](#_Toc325361344)

[Radni list 17](#_Toc325361345)

[Manipulacija radnim listovima 21](#_Toc325361346)

[Alatna traka 21](#_Toc325361347)

[Otvaranje već postojećih dokumenata 22](#_Toc325361348)

[Formule 23](#_Toc325361349)

[Dodaci za aplikaciju 26](#_Toc325361350)

[Zaključak 28](#_Toc325361351)

[Literatura I](#_Toc325361352)

[Knjige I](#_Toc325361353)

[Osnovne internetske stranice I](#_Toc325361354)

[Ostali internetski izvori I](#_Toc325361355)

[Prilozi IV](#_Toc325361356)

[Dodaci VII](#_Toc325361357)

Ova stranica je namijerno ostavljena prazna.

# Ideja projekta

Ideja projekta je kreirati aplikaciju za tablično računanje. Aplikacija, osim standardnih opcija kao što su unašanje podataka i otvaranja standardnih datoteka, treba omogućiti izračune i rad s formulama, oblikovanje teksta i obruba te podržavati više stranica po dokumentu.

Ovaj projekt vrlo je sličan Microsoft Excelu – najpoznatijoj aplikaciji za tablično računanje. Cilj ovog projekta nije da se kopiraju već postojeća rješenja već da se dodaju karakteristike koje nisu dostupne u već postojećim rješenjima. Pa tako ovaj projekt, umjesto da se orijentira prema korisnicima stolnih računala, cilja na korisnike tableta. Tableti su u posljednje vrijeme postali vrlo popularni i broj korisnika im raste iz dana u dan. Kako je Excel, a i ostale aplikacije za tablično računanje, namjenjen upravo korisnicima stolnih računala, dolazi do problema pri korištenju takvih alata na tabletima gdje je dodir primarno sredstvo komuniciranja s računalom. Kako bi omogučili korisnicima tableta lakšu i jednostavniju izradu tablica, kreirao sam aplikaciju koja radi baš to – olakšava izradu tablica na tabletima.

Sama ideja nije previše komplicirana ali kad se upustite u izradu aplikacije ovakvog tipa, naiđete na brojne prepreke koje je dosta teško savladati. Ostala softverska rješenja razvijali su timovi od par stotina ljudi, a ja – sam, ne mogu postići ni približno tako dobre rezultate ali mogu pokazati neke nove ideje.

Odlučio sam se za ovaj projekt zbog toga što sam želio vidjeti na koji način i koliko vremena treba utrošiti da bi se izradila jedna solidna aplikacija za tabličnu obradu, kako funkcionira leksička obrada formula, kako se formule prevode, kako se prevedene vrijednosti na kraju evaluiraju i da li je jednostavno pisati dodatke za aplikacije.

Korištene tehnologije

Sve korištene tehnologije dizajnirao je Microsoft. Time se postiže stabilnost i dugotrajnost proizvoda što je vrlo važno kod ovakvog tipa projekta. Microsoft je vodeći na tržištu novih tehnologija te pruža odličnu podršku do čak deset godina za sve nove tehnologije, pa tako i za tehnologije koje se koriste u ovoj aplikaciji. Kako je za sve ove tehnologije potreban .NET Framework 4 pa na više, preporuka je da se ova aplikacija koristi na Microsoft Windows 7 i novijim operativnim sustavima.

C# (C Sharp)   
Programski jezik – osnova projekta.

C# je objektno-orijentirani programski jezik. Kaže se još da je *strong typed*, *functional*, *generic* i *component-oriented*. Pojavljuje se 2001. godine izlaskom prve inačice .NET Frameworka. U početku je bio vrlo sličan *Javi* ali su se tokom njihovog razvoja počinjali uvelike razlikovati zbog različitih načina implementacija novih mogućnosti.

/// <summary>

/// Komentar klase

/// </summary>

public class Klasa {

private string varijabla;

/// <summary>

/// Komentar svojstva

/// </summary>

public string Svojstvo {

get { return this.varijabla; }

set { this.varijabla = varijabla; }

}

/// <summary>

/// Komentar konstruktora

/// </summary>

public Klasa() {

this.varijabla = "varijabla pocetno";

}

/// <summary>

/// Komentar metode

/// </summary>

/// <returns>Vraca linije</returns>

public IEnumerable<string> DobaviLinije() {

// ...

}

}

C# je vrlo sličan C++ programskom jeziku jer pokušava zadržati čitkost i matematičku formu - ono na što su programeri već naviknuti, pa navikavanje na C# ne bi trebalo biti teško. C# je do danas prošao mnoge preinake pa je tako trenutno aktivna inačina verzije 5.

Prvo se piše ključna riječ „*class*“ i zatim željeni naziv klase. Varijable mogu biti tipa *Integer*, *Double*, *String*, itd. Te naziv varijable ne smije početi s brojkom ili nekim specijalnim znakom. Konstruktor se poziva kada se iz neke klase radi objekt. On bi trebao inicijalizirati sve varijable i svojstva. Metode mogu vraćati neku vrijednost ili jednostavno vratiti prazninu, odnosno ništa *void*. Unutar metoda se nalaze instrukcije koje čine program. Ako je metoda „*static*“ onda se može pozvati bez da se prethodno kreira objekt.

Slika 1 - C# primjer klase

XAML (Extensible Application Markup Language)   
Jezik za opisivanje sučelja aplikacije.

XAML se prvi puta pojavljuje 2006. godine kao jezik za opisivanje sučelja prilikom korištenja WPF, Silverlight i WF tehnologija koristeći .NET Framework 3.0. Jezik je vrlo sličan XML i HTML standardu pa ga je jednostavno usvojiti. Primarno je namjenjen dizajnerima kako bi lakše komunicirali s programerima u izradi aplikacije.

Glavna razlika između XAML-a i npr. HTML-a jest ta da koristeći XAML programer ne postavlja svaki podatak na sučelje, već se za prikaz podataka koristi takozvani *Binding*. Upotrebom *bindinga*, dizajner definira kako se zove varijabla iz koje treba uzeti ili u koju treba zapisati neku vrijednost, a programer definira varijablu te mijenja ili čita njenu vrijednost. Na taj način dobiva se podvojenost dizajnera i programera, a samim time i fleksibilnost aplikacije, odnosno sučelja.

Još jedna vrlo korisna mogućnost XAML-a jest ta da se *tagovi* odnosno kontrole, mogu dodavati i nisu statički. Na taj način dizajnerima se omogućuje potpuna sloboda pisanja i prilagodba jezika njihovim potrebama.

WPF (Windows Presentation Foundation)   
Biblioteka kontrola kojima se objedinjuju C# i XAML.

WPF se prvi puta pojavljuje 2006. godine kao zamjena za već zastarjeli *Windows Forms*. WPF kao osnovu koristi bilo koji CLI (eng. *Common Language Infrastructure*) te XAML jezik za izradu sučelja.

Iscrtavanje *Windows Formsa* zasnivalo se na GDI (eng. *Graphics Device Interface*) podsustavu koji je znatno sporiji od danas dostupnih rješenja. Iz tog razloga WPF više ne koristi GDI već *DirectX* biblioteku kao osnovu za iscrtavanje. *DirectX* omogućuje, aplikacijama izrađenim u WPF-u, da koriste sve mogućnosti modernih grafičkih kartica kao što su 2D i 3D iscrtavanja, efekti, itd.

### MEF (Managed Extensibility Framework)

Biblioteka klasa koje olakšavaju izradu i korištenje dodataka za aplikaciju.

MEF se prvi puta se pojavljuje 2011. godine u .Net Framework 4.0 pa je time najnovija tehnologija koja se koristi u ovoj aplikaciji. MEF olakšava izradu dodataka za aplikaciju tako da standardizira osnove dodataka. Stoga se za implementaciju dodataka u aplikaciju ne mora pisati cijela infrastruktura iz početka. Time se dobiva na robusnosti i modularnosti aplikacije, a i omogućuje jednostavnije održavanje.

MEF je namijenjen manjim aplikacijama iz razloga što omogućuje široki spektar korištenja. Kako bi dodali „*dodatak*“ u aplikaciju, mora se izraditi biblioteka koja sadrži sve potrebne klase. Ta se biblioteka treba učitati koristeći već postojeće metode iz MEF biblioteke.

Svaki dodatak se označi kao *Export*,odnosno izvozna klasa, a aplikacija ima jedan parametar *Import* kojim naznačuje da prima dodatke. Kada se u zadanoj mapi nalazi barem jedan dodatak, aplikacija učitava taj dodatak te se on, kako je već zadano, implementira u aplikaciju kao da je dio nje.

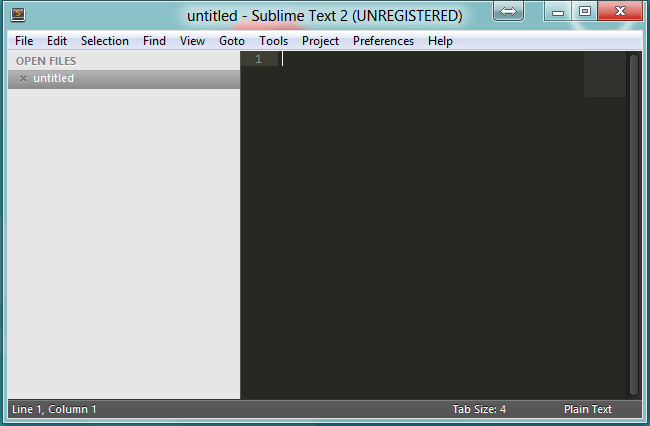
## Razvojni alati

Za izradu Metro Tables aplikacije koristio sam nekoliko alata. Ti alati su dostupni besplatno za nekomercijalne svrhe te se mogu preuzeti s interneta.

* Visual Studio 11 Beta
* Microsoft Expression Blend 4
* Sublime Text 2

### Sublime Text 2

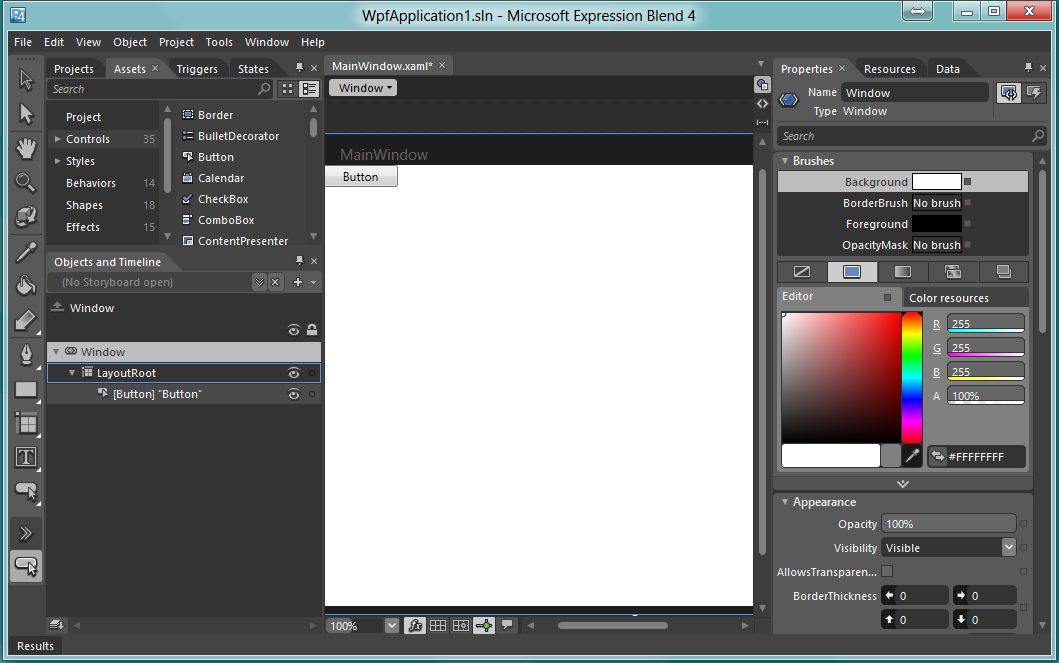
Alat je osmišljen kao zamjena za aplikaciju *notepad.*  Kao takav ima mnogo mogućnosti koje uvelike mogu olakšati posao pri razvoju softvera. Jedna od najvažnijih karakteristika je umanjeni prikaz dokumenta s desne strane aplikacije. Taj prikaz omogućuje lakše snalaženje u dokumentima s više tisuća linija i time doprinosi produktivnosti korisnika. Ovaj alat nije besplatan ali se može koristiti koliko god se želi uz prisutnost reklama i upita za kupnju.



Slika 2 - Sublime Text 2

### Microsoft Expression Blend 4

Microsoft Expression Blend 4 je alat prvenstveno namjenjen dizajnerima. Koristio sam ga da bi lakše izradio sve animacije i osnovni dizajn korisničkog sučelja. On dolazi u skupu aplikacija za izradu dizajna internetskih stranica, video zapisa i drugih, a namjenjen je za dizajniranje aplikacijama baziranih na *Silverlight* i *WPF* tehnologijama. Ima vrlo dobro razrađeno sučelje za manipulaciju kontrolama i upravljanje vremenskom linijom animacije. Pruža potpuno razvojno okruženje za pisanje XAML-a. Microsoft Expression Studio 4 koji sadrži Microosft Expression Blend 4 može se preuzeti s Microsoftove stranice besplatno u probnom roku od 30 dana.

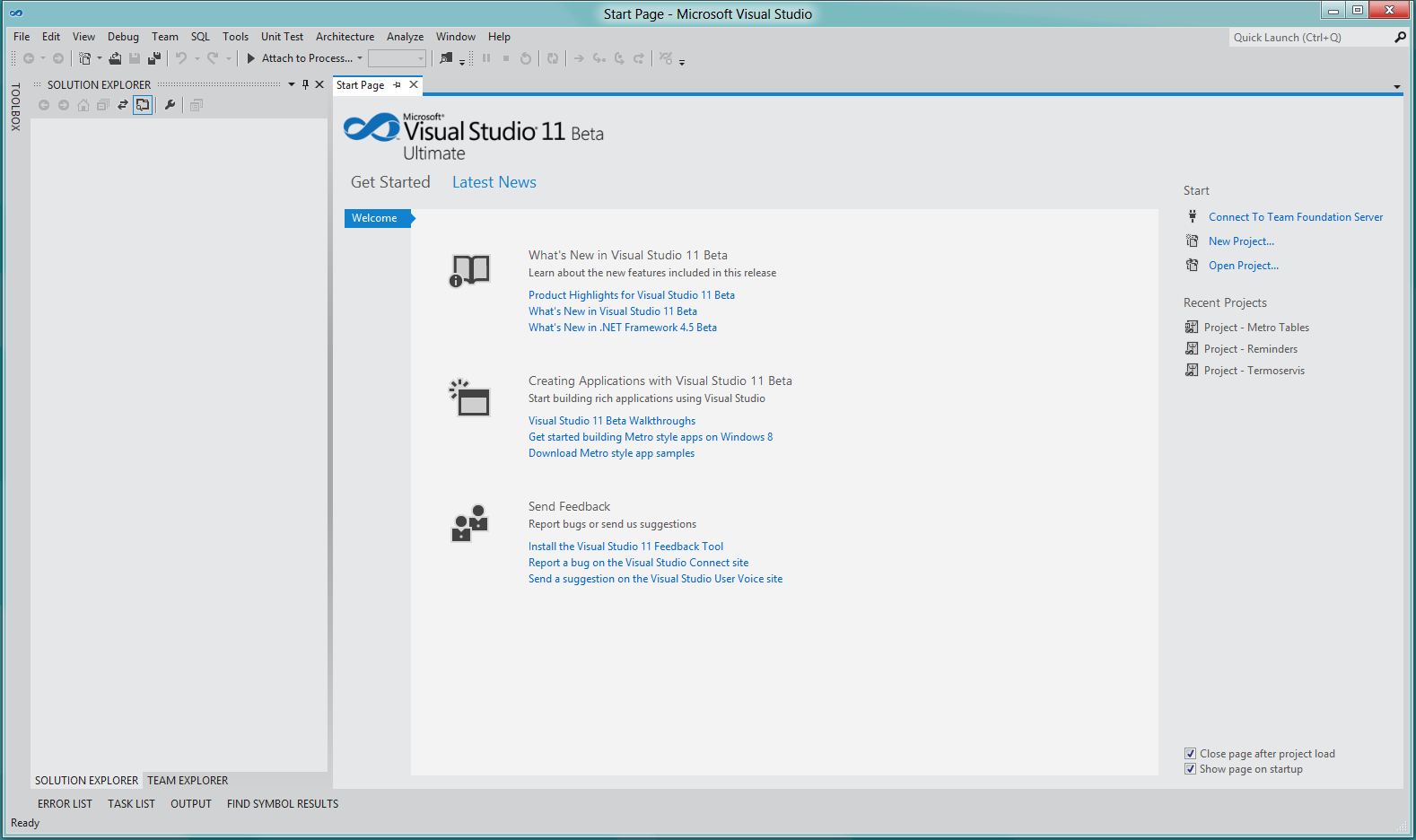


Slika 3 - Microsoft Expression Blend 4

### Visual Studio 11 Beta

Za izradu ovog rada koristio sam alat pod nazivom Micorosft Visual Studio 11*.* Alatjejoš u beta fazi razvoja pa je tako aplikacija na samom rubu mogućnosti WPF tehnologije. Iako je alat u beta fazi, vrlo je stabilan i moguće ga je koristiti bez previše problema i grešaka.

Kad pokrenemo Visual Studio, prvo nam se pokaže takozvani „*Start Page*“. Na početnoj stranici dostupne su poveznice najčešće korištenih projekata te neke poveznice za učenje korištenja Visual Studia. Sučelje se sastoji od 4 glavna djela. Izborna traka gdje se može doći do svim opcija i postavki, alatne trake koje se mogu prilagoditi prema potrebama korisnika i sadrže prečice do najčešće korištenih opcija, trake stanja koja daje informacije o trenutnom stanju projekta na kojem radimo, te središnjeg dijela u kojem se nalaze razni prozori za informiranje, manipulaciju dokumentima i projektom.



Slika 4 - Visual Studio 11 (Start page)

Kako bi započeli s radom na nekom projektu, prvo ga moramo kreirati ili otvoriti ukoliko on već postoji. U našem slučaju, projekt je već kreiran i samo moramo odabrati lokaciju projekta na disku.

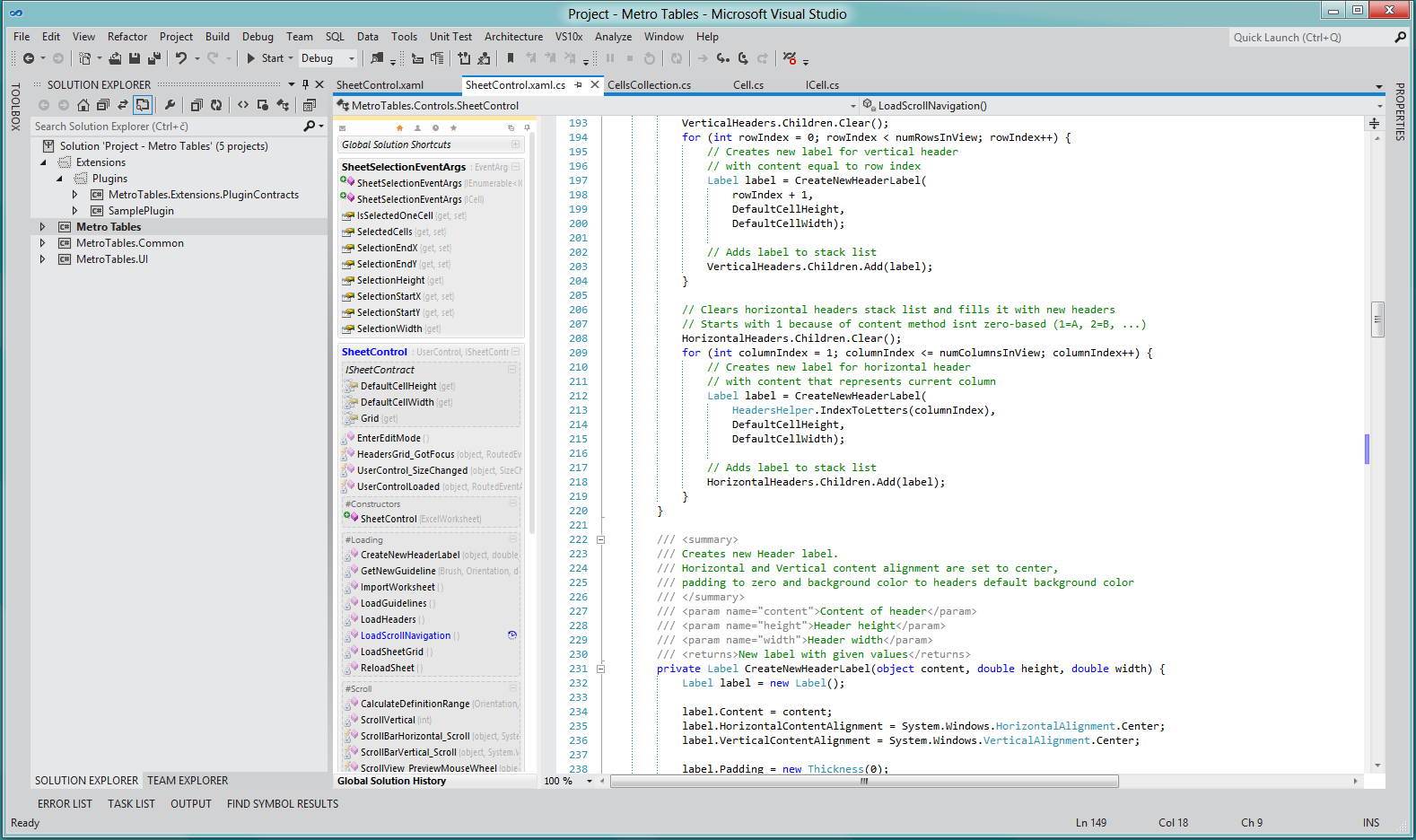
Da bi otvorili neki dokument iz projekta prvo moramo „*navigirati*“ do njega uz pomoć prozora „*Solution Explorer*“ koji se u Visual Studiu najčešće nalazi s lijeve strane.

Unutar „*Solution Explorer“* prozora nalazi se lista svih dokumenata, mapa i ostalih resursi koji se koriste u projektu. Kako projekt može sadržavati par stotina dokumenata, svi su dokumenti organizirani po mapama. Time se postiže brži rad i preglednosti projekta.

Tipovi dokumenata koje sam koristio u projektu su: C# Source file (*\*.cs*), XAML file (*\*.xaml*) i slike (*\*.png* i *\*.jpg*). Ovi dokumenti su svi podržani od strane Visual Studia što znači da se mogu pregledavati i uređivati bez nepotrebnih prebacivanja u druge alate.

Kako bi aplikacija bila modularna, projekt se nalazi u skupu projekata. To omogućuje da se jednostavno upravlja s više projekata bez nepotrebnih otvaranja više instanci alata.

Kada otvorimo neki dokument koji sadrži kôd programa (npr. \*.cs dokument), otvara se novi prozor u Visual Studio koji nam omogućuje napredno uređivanje teksta odnosno kôda programa.

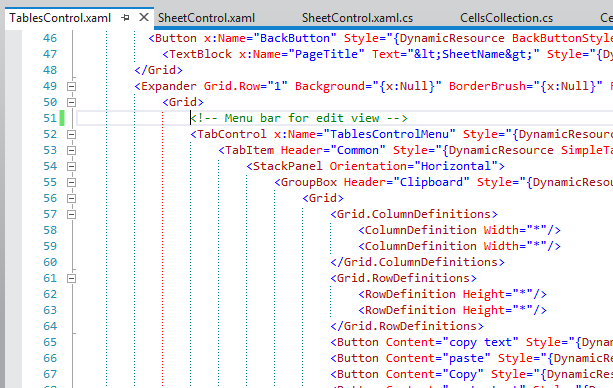
****

Slika 5 - Visual Studio 11 (Code Editor)

Glavne napredne mogućnosti ovakvog editora su bojanje teksta. Boje odgovaraju točno određenim tipovima i strukturama u programu. Tako je komentar zelene boje, klasa svijetlo plave a tekst crvene boje. Ovakav način označavanja kôda znatno poboljšava brzinu čitanja i pisanja programa, a i pomaže pri organizaciji same strukture kôda.

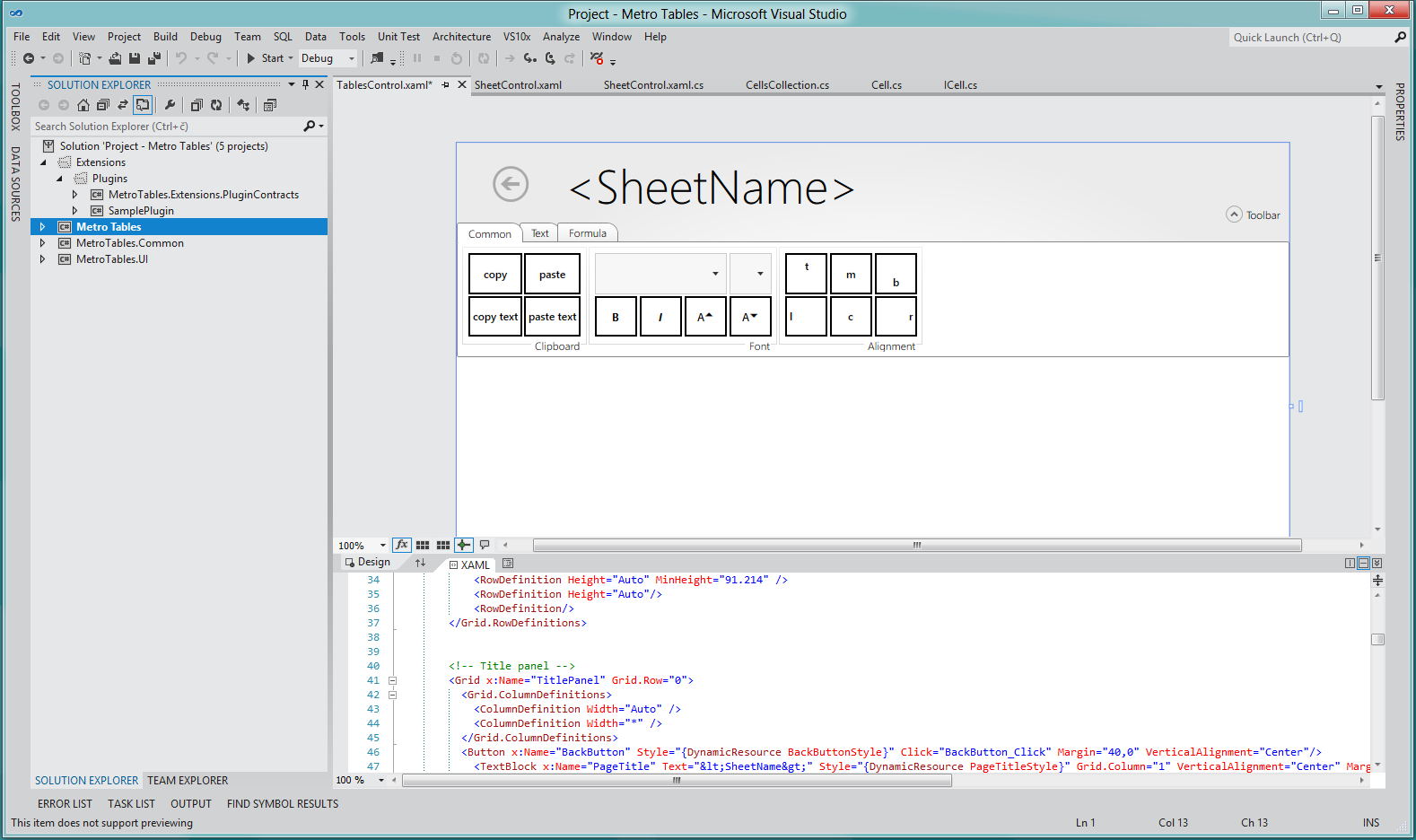
Jedna od velikim mogućnosti editora Visual Studia jest „*Intellisense*“ – Microsoftova implementacija opcije za samo-dopunjavanje. Intellisense prepoznaje riječ koju želimo napisati, te ponudi dopunu još kad korisnik nije napisao riječ do kraja. *Intellisense* uz dopunjavanje pruža i pomoć pa se prije odabira riječi (klase ili varijable) uz tu riječ pojavljuje i komentar, te opis. Opis i komentar su važni tako da korisnik ne mora pamtiti sve detalje oko pojedine klase ili varijable.

Ukoliko pak otvaramo XAML dokument (*\*.xaml*), ne otvara nam se ovakav editor već editor namjenjen baš za uređivanje XAML kôda. XAML (eng. *Extensible Application Markup Language*) naime nije programski jezik već jezik za opisivanje. Razvijen je od strane Microsofta i vrlo je sličan HTML-u, a zapravo potječe od XML jezika.



Slika 6 - Visual Studio 11 (XAML Editor)

XAML editor isto kao i editor kôda ima opciju za bojanje teksta i *Intellisense* što uvelike pomaže pri opisivanju stilova kontrola i izradi sučelja. Čitanjem XAML-a možemo si otprilike predočiti izgled aplikacije. Ali kada aplikacija postane malo kompleksnije, predočavanje postaje preteško.

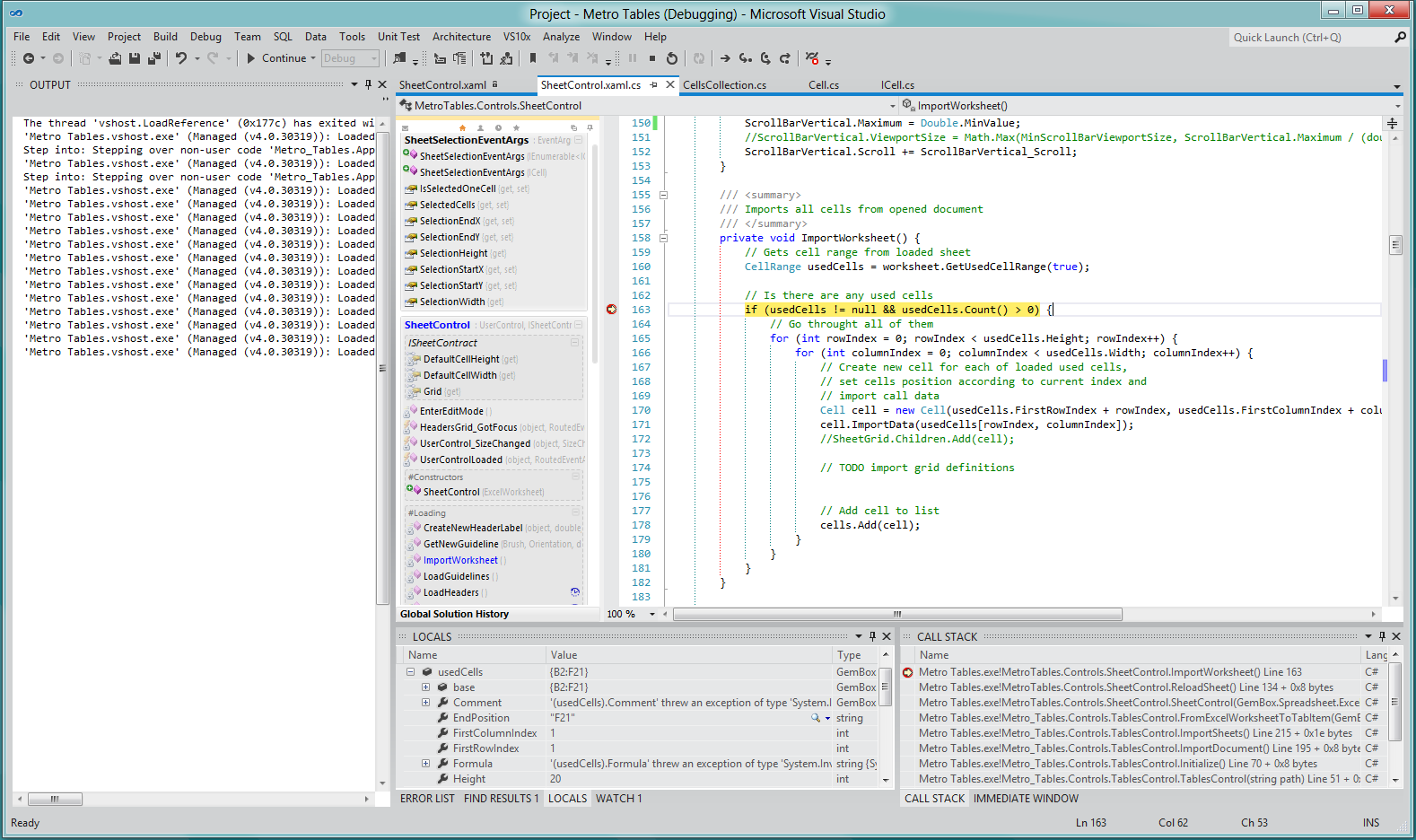


Slika 7 - Visual Studio 11 (Designer)

U takvim slučajevima, koristi se alat „*Designer*“ - dodatak XAML editoru koji prikazuje aplikaciju tako da pretvara napisani XAML kôd u kontrole koje možemo pomicati i mijenjati. „*Designer*“ omogućuje vrlo jednostavnu (skoro interaktivnu) manipulaciju kontrolama po trenutnom prozoru. Kako korisnik pomiće kontrole, XAML kôd se automatski prilagođava promjenama.

Kada mislimo da je projekt spreman za testiranje, uz Visual Studio dolazi i alat za otkrivanje grešaka odnosno „*debugiranje*“. To zapravo nije jedan alat već skup alata koji pomažu korisniku Visual Studia da si lakše predoči problematične dijelove programa. *Debug* opcija je vrlo močna i teško je raditi na većim projektima bez korištenja tog ili njemu sličnih alata.

*Debug* alat nam daje pristup i uvid u trenutna stanja varijabli tokom izvršavanja programa, te time programer može vidjeti na kojem mjestu dolazi do grešaka. Druga opcija kojom se može ispitivati ispravnost programa je korištenje *Output* prozora. U Output prozoru prikazuju se poruke od .NET biblioteke te od same aplikacije u izvršavanju. Tako se pozivanjem jednostavne naredbe može u *Output* prozoru pokazati informativna poruka koja onda programeru daje do znanja što se događa tokom izvršavanja programa.



Slika 8 - Visual Studio 11 (Debug)

Tu su još i alati za otklanjanje pogreški unutar dretvi, pronalaženje mogućih problematičnih dijelova kôda prije samog izvršavanja, lista prije pozvanih metoda, takozvani „*Call stack“* , itd.

Visual Studio je vrlo opširno razvojno sučenje i zbog toga je, prema mnogima, najbolji alat za razvijanje aplikacija na Windows platformi. Alat koji sam ja koristio još je u *beta* fazi pa je stoga besplatan u nekomercijalne svrhe dok je puna inačica dostupna na Microsoftovoj stranici u više verzija ali po dosta visokim cijenama. Ukoliko razvijate softver u edukacijske svrhe, Microsoft studentima nudi besplatne edukacijske alate, među koje spada i Visual Studio, na stranicama *DreamSpark.com*.

# Metro Tables

## Uvod

Metro Tables je naziv aplikacije koji potječe od naziva „*Metro*“, kako se naziva novo sučelje Windows 8 operativnog sustava, te „*Tables*“ što označava rad s tablicama – temu završnog rada. Kako već postoje mnoge aplikacije za izradu proračunskih tablica, odlučio sam se malo odmaknuti od već postojećih rješenja i napraviti aplikaciju za tablet računala.

Da bi se korisnicima tablet računala olakšala izrada proračunskih tablica, sučenje je potrebno malo uvečati i omogućiti jednostavan pristup često korištenim naredbama i operacijama.

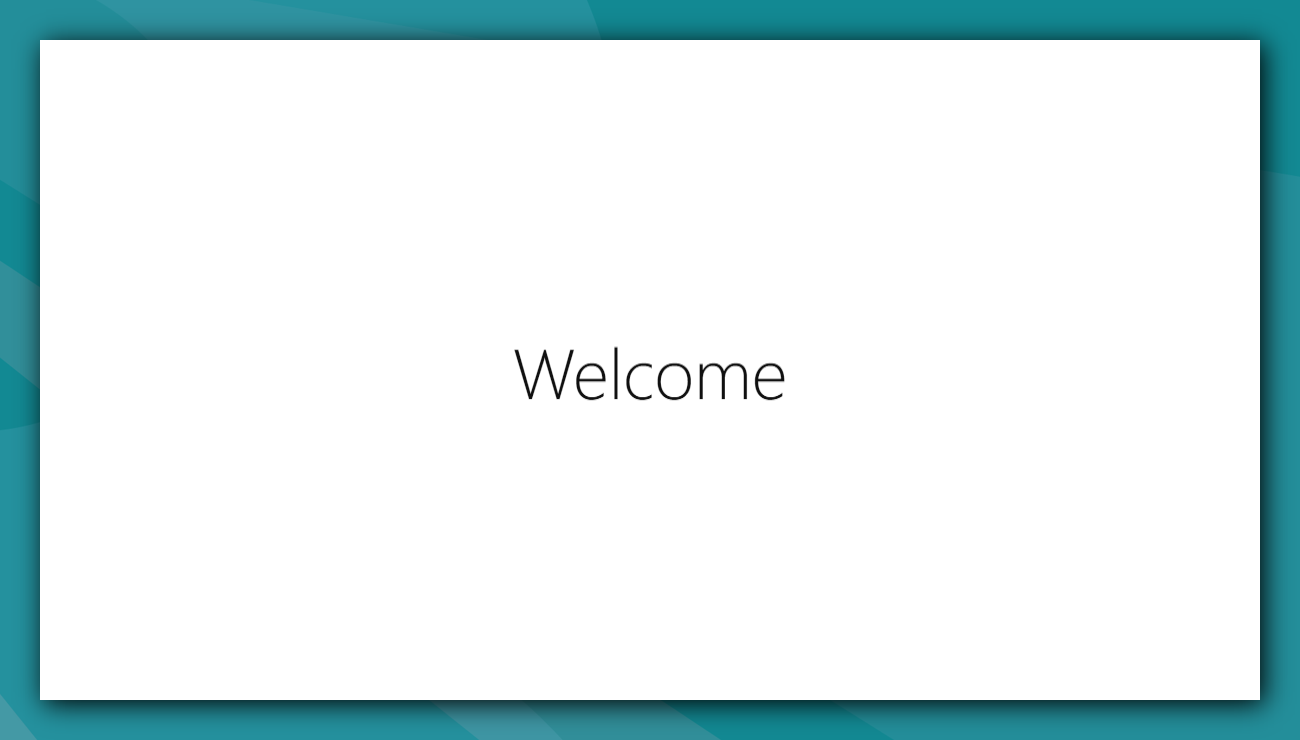
## Pokretanje aplikacije

Pokrenuti aplikaciju korisnik može na nekoliko načina. Jedan od najjednostavijih jest da korisnik dva puta klikne na prečac. Aplikacija zatim učitava sve potrebne biblioteke za njeno pokretanje te se pokazuje prvi prozor.

Prvi prozor je tipa *Window* što je uobičajeno za WPF aplikaciju. Taj prozor nema obrub, funkciju pomicanja niti bilo kakav drugi dizajnerski detalj. Prozor je jednostavno prazan – bijele boje. Da bi se nešto prikazalo mora se kreirati *Frame*. *Frame* je kontrola koja omogućuje prikazivanje *Page* kontrola i navigaciju.

*Page* kontrole su vrlo korisne jer omogućuju neprimjetnu izmjenu stranica što rezultira dobrim korisničkim iskustvom bez previše skakanja s prozora na prozor. *Page* kontrola još olakšava izradu sučelja pa tako aplikacija izgleda kao da se cijela izvršava unutar istog prozora. To je točno, ali se izmjenjuju *Page* kontrole. Time se postiže da dizajner ne mora raditi s velikim brojem kontrola odjednom, već samo s kontrolama koje se prikazuju na određenoj *Page* kontroli.

Kada se doda *Frame* kontrola na početni prozor, *Frame* „*navigira*“ do pozdravne stranice. Pozdravna stranica sadrži poruku pozdrava i pokreće učitavanje aplikacije, odnosno generiranje potrebnog sadržaja.

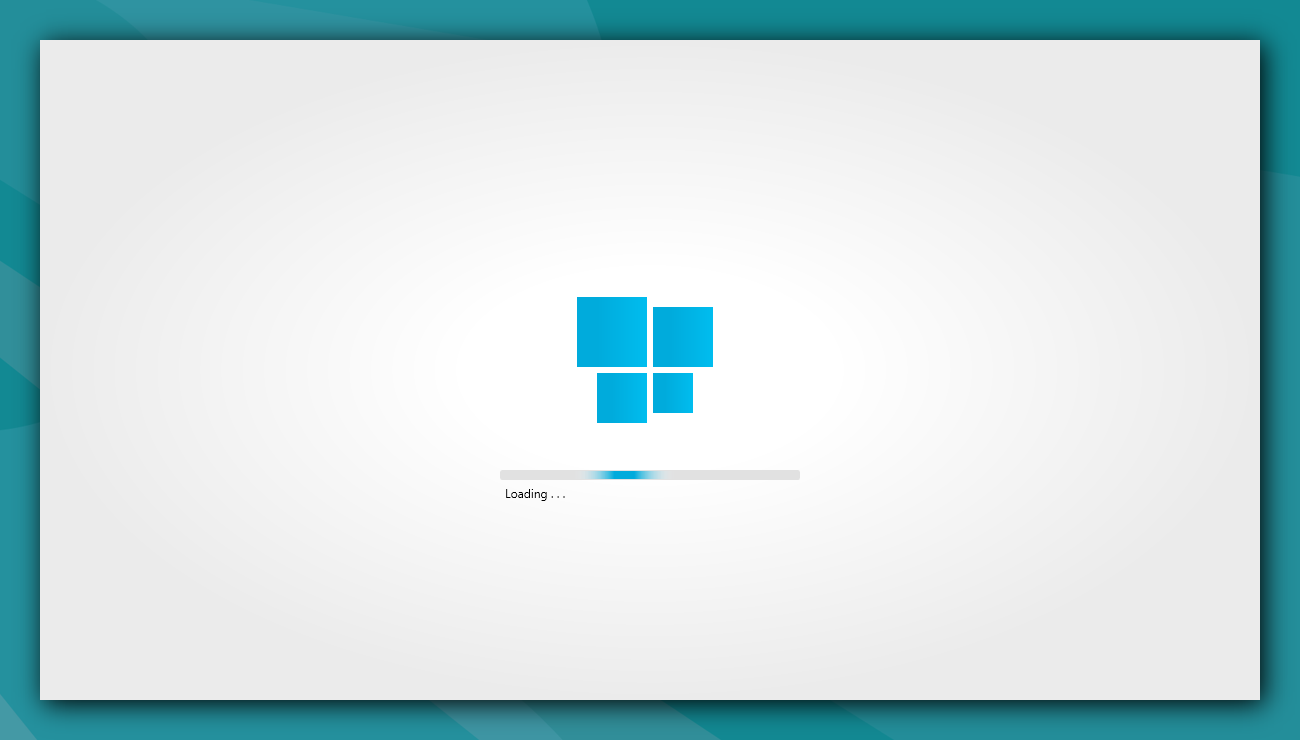


Slika 9 - Metro Tables (Welcome Page)

Učitavanje potrebnog sadržaja izvršava se tako da se učita sljedeća stranica koja se treba prikazati i pokrene se njeno učitavanje, odnosno kreira se novi objekt koji predstavlja *Page* kontrolu koju želimo prikazati nakon završetka učitavanja. Učitavanje u ovom slučaju ne traje predugo zbog toga što se ne generira mnogo zahtjevnog sadržaja. Jedini razlog što je učitavanje implementirano u ovu aplikaciju jest taj da korisnik ne čeka nekoliko sekundi sa praznim prozorom, već da mu se da na znanje da će aplikacija biti upotrebljiva uskoro.

Pozdravna poruka se nakon nekoliko sekundi makne s vidljivog područja aplikacije i u vidljivo područje dolaze logo, kontrola stanja i poruka stanja.

Kada kontrola stanja postane vidljiva prelazi u takozvano *Intermediate* stanje koje označava da aplikacija učitava potrebne resurse i generira sadržaj. U ovom stadiju, aplikacija kreira novi objekt tipa „*HomePage“* koji predstavlja početnu stranicu aplikacije. Kako bi sučelje i dalje ostalo upotrebljivo, odnosno ne bi došlo do prestanka kretanja animacija, koriste se dretve. Dretve, svaka za sebe, učitava jedan dio aplikacije, te tako održavaju korisničko sučelje osjetljivim na korisnikove zahtjeve.



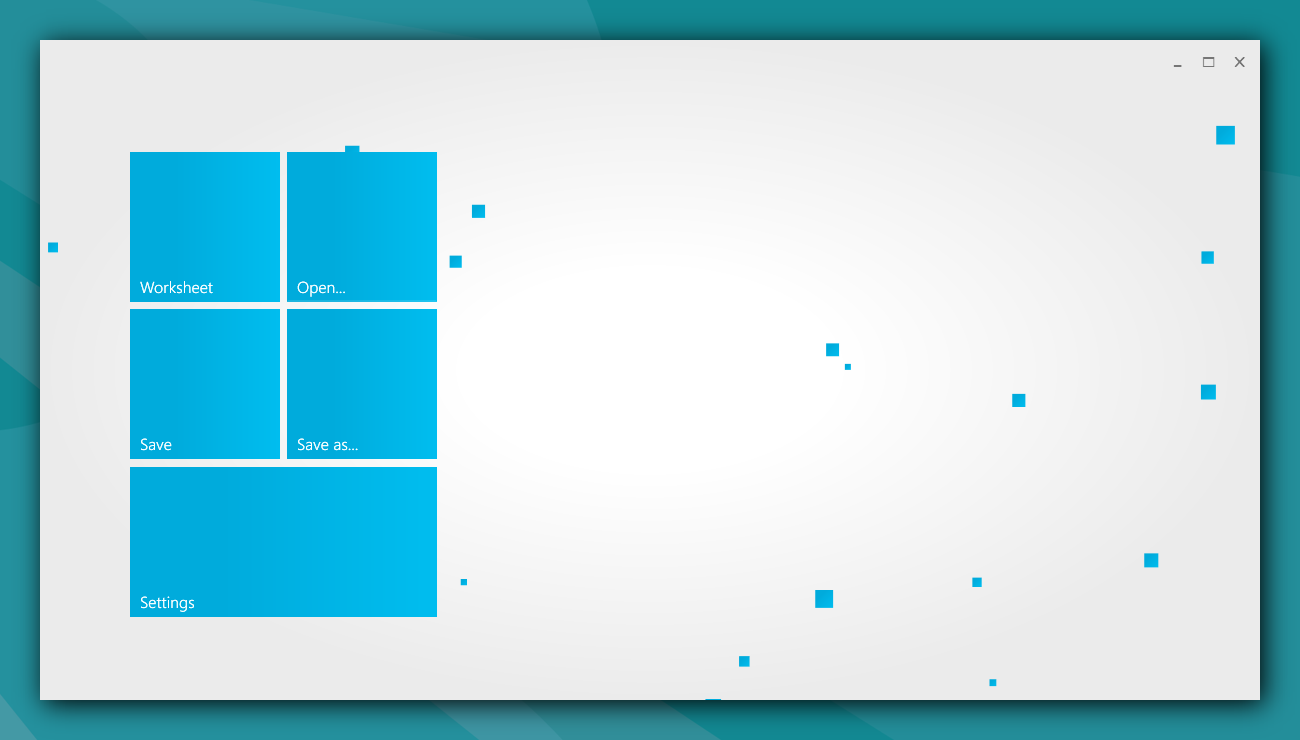
Slika 10 - Metro Tables (Učitavanje)

Nakon što je sljedeća stranica učitana, može se pozvati *Frame* da „*navigira*“ na učitanu stranicu. Prije toga se pokreće animacija kojom se kontrole učitavanja miću iz vidljivog područja.

## Početna stranica

Početna stranica ukrašena je brojnim, nasumično generiranim, kvadratićima koji su razbacani po cijelom vidljivom području aplikacije. U gornjem desnom kutu nalaze se tri gumba za kontrolu prozora – smanji, povečaj i zatvori, tim redosljedom. Ti gumbi prisutni su na većini prozora Windows operativnog sustava. Kako bi korisnik mogao manipulirati prozorom, u gornjem dijelu dodan je pojas širine 45 pixela unutar kojih korisnik može pritisnuti lijevim klikom miša te aktivirati pomak prozora.

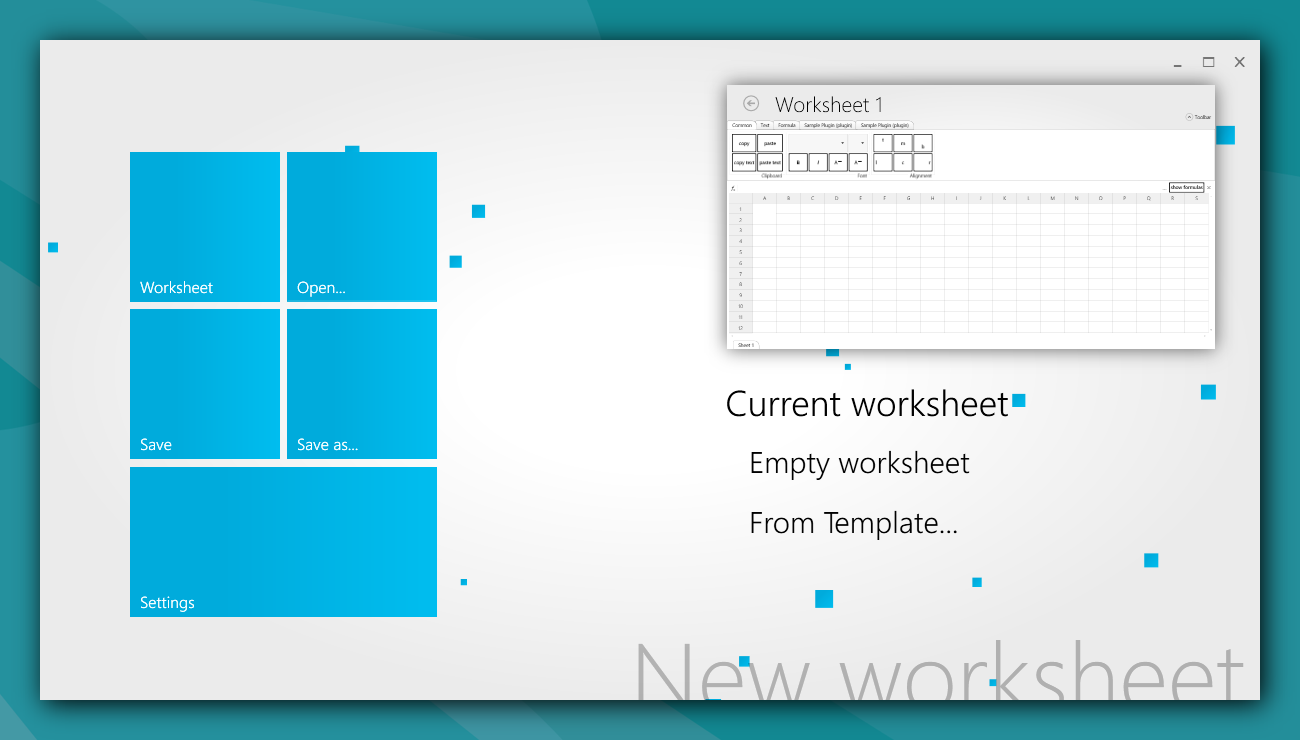
Kako bi se korisniku ponudile opcije, koriste se takozvani *Tile* gumbi. Ovi gumbi specifični su za Windows 8 operativni sustav i poznati su njegovim korisnicima. Naime, ovaj tip gumbi uvelike olakšava odabir opcije za korisnike tablet računala. Ti gumbi su veliki, jednostavni i sadrže tekst koji ukratko govori za što služe. Svaki gumb ima svoju namjenu, pa je tako dostupno 5 gumbi: „*Worksheet*“, „*Open...*“, „*Save*“, „*Save As...*“ i „*Settings*“.



Slika 11 - Metro Tables (početna stranica)

„*Worksheet*“ gumb koriste korisnici kad žele manipulirati radnim listom, odnosno kad žele pregledati trenutni list ili odabrati neki drugi list. Tako pritiskom na gumb Worksheet, na desnoj strani aplikacije, pojavljuje se dodatni izbornik koji korisniku pruža još veću kontrolu. Dodatne mogućnosti koje korisnik može odabrati jesu: „*Current* *worksheet*“, „*Empty* *worksheet*“, „*From template...*“. Bilo koju od ovih opcija korisnik odabire tako da dva puta klikne na tekst željene opcije. Zadani odabir je uvijek prva opcija pa tako korisnik, da bi brže odabrao, može jednostavno dva puta kliknuti na „*Worksheet*“ *Tile* gumb što je jednako kao dao je dva puta kliknuo na prvu (zadanu) opciju. Tako „*Current worksheet“*  prikazuje trenutni otvoreni radni list, „*Empty worksheet*“generira novi – prazan radni list, a „*FromTemplate...*“kreira novi radni list iz već postojećeg predloška.

„*Open...*“ gumb korisnik koristi kad želi otvoriti već postojeći radni list. Radni list se može nalaziti u *.xslx* formatu. To je *OpenDocument* [24] tip datoteke pa se može koristiti u razne svrhe i nije potrebna nikakva licenca za njegovo korištenje. To je ujedino i primarni format Microsoft Excela pa je tako vrlo koristan i prihvatljiv. Korisniku se od dodatnih mogućnosti nudi: „*From* *file...*“, „*From* *URL*...“, „*Recent*“. Opcija „*From file“* omogućuje otvaranje datoteka na lokalnom disku računala, „*From URL...“* omogućuje da se datoteka otvori se nekog udaljenog sjedišta (http, https, ftp, itd.) , a „*Recent...“*  za otvaranje prijašnjih datoteka.



Slika 12 - Metro Tables (dodatne opcija za Worksheet)

„*Save*“ gumb sprema trenutno otvorenu radnu listu te ne nudi dodatne opcije za spremanje. Ovo je vrlo jednostavan gumb pa ako je dokument već bio spremljen, odnosno ako nije novo kreirani dokument, sprema se na zadanu lokacije. U suportnom radi isto kao i „*Save as*...“ gumb.

„*Save* *as*...“ gumb korisniku omogućuje da dokument s promjenama spremi u standardnom formatu *.xslx*. Nudi par dodarnih opcija: „*Save* *as* *file*...“, „*Save* *as* *Text*...“ i „*Custom* *save*...“. Opcija „*Save as file*...“omogućuje normalno snimanje datoteke na lokalni disk računala u standardnom formatu, „*Save as Text...“* omogućuje spremanje radnog lista u tekst formatu i služi samo za jednostavne tablie, a „*Custom save...“* omogućuje dodacima da spremaju u vlastitom formatu.

„*Settings*“ gumb služi za postavljenje aplikacije odnosno postavljenje zadanih vrijednosti aplikacije i njen izgled. Ovdje spadaju korisničke postavke kao što su zadani naziv novog dokumenta, veličina i obitelj zadanog fonta, itd.

Prikaz trenutnog radnog lista vidljiv je samo kada se odabere „*Worksheet*“ gumb. Prikaz je vrlo realan i u stvarnom vremenu, pa će se tako vrijednosti, koje su ovisne o trenutnom vremenu ili se učestalo izračunavaju, osvježavati i u prikazu liste.

## Radni list

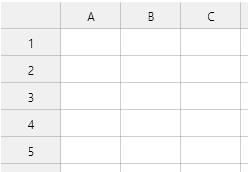
Radni list (eng. Workspace) se sadrži od trake alata, tablice, trake formule i popisa listova. Da bi korisnik mogao pregledavati tablicu, ona sadrži nekoliko ključnih stavki.

Vertikalna i horizontalna traka za pomak omogućuju jednostavno kretanje po većim dokumentima, a i olakšavaju vizualnosnalaženje. U ovoj aplikaciji ove trake su implementirane tako da se pojavljuju kada ima dostupnih ćelija na određenoj poziciji. Uz to, ukoliko se korisnik nalazi na potpuno praznom listu, trake nude mogućnost kretanja do pozicije na kojoj se ne nalazi ćelija, odnosno pritiskom gumbi gore-dole ili lijevo-desno pomiču tablicu u smjeru. To se postiže tako da ukoliko korisnik klikne neki od tih smjerova, na poziciji koju bi korisnik trebao vidjeti kreira se samo jedna ćelija kako bi se i nakon korisnikova vraćanja na početak dokumenta ta pozicija sačuvala, te bi trake pomaka sačuvale veličinu. Ovakvo ponašanje ne potječe iz niti jednog poznatog alata za izradu proračunskih tablica, već je rezultat mojeg iskustva s izradom tablica. Ovo ponašanje doprinosi ukoliko se žele usporediti veličine nekih brojeva. Naprimjer ukoliko korisnik želi vidjeti koliko bi mjesta zauzela tablica veličine 700 polja (vertikalno), da ne bi trebao raditi tako veliku tablicu, samo odlazi na tu poziciju unutar dokumenta i može otprilike vidjeti razliku veličina trenutne trake pomaka i prijašnje trake pomaka. Traka pomaka radi samo kada je pokazivač unutar polja ćelija, te prilikom prelaza na naslovna polja pomicanje više ne radi, odnosno radi samo pritiskom tipka dore-dolje-lijevo-desno na tipkovnici kojim korisnik upravlja trenutno odabranu ćeliju.



Slika 13 - Metro Tables (vertikalni/horizontalni pomak)

Naslovna polja nalaze se na lijevom i gornjem rubu polja ćelija i pokazuju na trenutnu oznaku ćelije. Naslovna polja služe za orijentaciju prilikom horizontlanog-vertikalnog pomaka kako se korisnik ne bi zbunio zbog preskakivanja nekoliko ćelija. Naslovna polja zapravo služe za označavanje trenutne ćelije ap se kao takve koriste prilikom pisanja formula. S lijeve strane nalazi se vertikalno naslovno polje koje pokazuje redni broj označene ćelije. Redni broj pokazuje se u brojevima od 1 do teoretskih 2 147 483 647 koliko je granica cijelobrojnog tipa u koji se sprema redni broj (integer). Gornje naslovno polje odnosno horizontalno naslovno polje, sadrži naziv označene ćelije koji se stvara na poseban način. Ovaj tip oznake koristi se i u Microsoft Excel alatu, a i u većini drugih alata za izradu proračunskih tablica. Oznake se kreiraju tako da se uzima broj, te se pretvara u slova. Tako je prvo slovo 'A', drugo 'B', treće 'C', itd. Sve dok se ne dođe do slova 'Z' kada nam je ponestalo slova. Tada se dodaju dodatna slova pa je 27. stupac označen s 'AA', 28. stupac s 'AB', itd. Ovaj način označivanja vrlo je sličan dekatskom brojevnom sustavu samo što je u ovom slučaju baza 27 umjesto 10. Ovime se dobiva na jednostavnijem razlikovanju stupaca od redaka pri pisanju formula pa se tako npr. Prvi stupac – deseti redak može zapisati kao A10 gdje A predstavlja prvi stupac (horizontalno) a 10 redni broj (vertikalno).

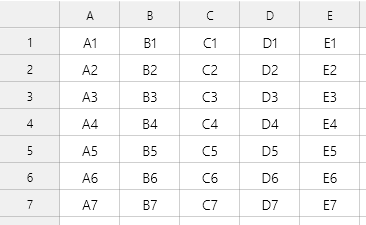


Slika 14 - Metro Tables (naslovna polja)

Vjerojatno najvažniji dio na listu jest polje ćelija. Polje ćelija sastoji se od mnogo ćelija koje mogu sadržavati neki tekst, formulu, pa čak i objekt dodatka. Kako bi mogli razaznati jednu ćeliju od druge, pri izradi tablice koristi se mreža takozvane *Guidelines*, odnosno linije vodilje. Linije vodilje obrisi su rubova ćelija te se mogu isključiti kako bi dobili realan prikaz tablice. Između linija vodilja nalaze se ćelije. Svaka ćelija može imati četiri obruba lijevi-desni-gornji-donji. Da linije vodilje ne bi smetale obrubu ćelije nalaze se na najnižen sloju tako da se, prilikom dodavanja obruba ćeliji, obrub vidi nezavisno o tome da li su linije vodilje uključene ili isključene.

Svaka ćelija ima dva osnovna stila – označena i neoznačena ćelija. Stil označene čelije javlja se kad korisnik odabere čeliju ili nekoliko ćelija. Odabir nekoliko ćelija vrši se tako da korisnik klinke na prvu čeliju s kojom želi započeti odabir, te vuće pokazivač prema čeliji s kojom želi završiti označivanje. Desnim klikom na označenu ćeliju, pojavljuje se pomočni izbornik kojim korisnik može na brzinu odabrati neku od najčešće korištenih naredbi kao što su kopiraj, zaljepi, izreži, obriši, itd.

Kako se npr. na polju veličine 10 sa 26 treba prikazati 260 ćelija, a svaka od tih ćelija treba brinuti da li je korisnik odabrao tu ćeliju, da li treba promjeniti stil, da li se ulazi u stanje uređivanja sadržaja, itd., na polju ćelija se, dok je list prazan, pokazuje samo jedna ćelija (ona na koordinatama A0) i ne razlikuje se od ostalih koje nastaju tijekom izrade tablice. Da bi korisnik mogao odabrati ćeliju tamo gdje je još nema, razradio sam algoritam za određivanje koordinata ćelije koju korisnik želi odabrati. Prije nego što označimo to polje, treba kreirati ćeliju i postaviti je na te koordinate, pa je onda možemo odabrati. Korisnik ne primjećuje da zapravo ćelija ne postoji na tim koordinatama jer je kreiranje jedne ćelije vrlo brz postupak. Kreiranje više stotina ćelija odjednom bio previše dugotrajan i zahtjevan posao. Na ovaj način optimizirali smo vrijeme podizanja aplikacije koje bi bilo za par sekundi duže što je vrlo nepoželjno.



Slika 15 - Metro Tables (polje ćelija s testnim podacima)

Prilog 2 je dio kôda koji pokazuje kako radi klasa za dinamičko pribavljanje i dodavanje ćelija na radni list. Klasa se naziva „*CellsCollection“* jer zapravo samo predstavlja dvodimenzinalno dinamičko polje za ćelije:

public List<List<Cell>> Cells;

Kada želimo pribaviti neku ćeliju iz kolekocije poziva se naredba unutar koje trebamo navesti na kojoj poziciji (vertikalnoj i horizontalnoj) se nalazi željena ćelija. Ovo se vrši tako da u varijablu ćelije spremimo rezultat naredbe:

Cell cell = this.cells[row, column];

Sad se poziva metoda koja se nalazi u isječku kôda. Metoda zatim odlazi pod *get* dio kôda jer se zahtjeva rezultat metode, a ne postavljenje vrijednosti. Zatim se ispituje da li se dana vrijednost vertikalne i horizontalne pozicije ne nalazi u području koje pokriva dvodimenzionalno polje ili je vrijednost ćelije na toj poziciji unutar dvodimenzionalnog polja jednaka *null*. Ukoliko je to slučaj, od klase se zahtjeva da se kreira nova ćelija. Na kraju *get* dijela metode – vreća se ćelija s dane pozicije jer sada, čak i ako prije nije postojala, postoji ćelija na toj poziciji. Zbog ovoga prilagođavanja veličine polja ova metoda se naziva dinamička.

Ukoliko pak želimo postaviti neku ćeliju na danu poziciju, prvo moramo, ukoliko je potrebno, proširiti polja ćelija ove klase. Za to imam jednostavno riješenje:

while (this.Cells.Count <= rowIndex)

Cells.Add(new List<Cell>());

Petlja se vrti tako dugo dok nam nedostaje mjesta u polju da bi dodali novu ćeliju. Isto tako se i radi s poljem stupaca. Zatim je potrebno dodati definicije redova i stupaca, nešto što je specifično za *Grid* kontrolu.

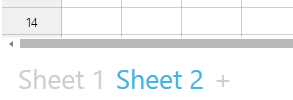
Na kraju kad imamo sve dovoljno prošireno, danu ćeliju stavljamo u polje ćelija te je dodajemo na listu objekata *Grid* kontrole na radnom listu.

Sad kad trebamo neku ćeliju iz radnog lista, samo pozovemo ovu metodu kolekcije ćelija i bez problema možemo dobiti rezultat, čak i ako nismo kreirali tako veliko polje ćelija. Na ovaj način dobili smo (skoro) beskonačno polje ćelija.

## Manipulacija radnim listovima

Manipulacija radnim listovima omogućuje da korisnik u jednom dokumentu kreira više od jednog radnog lista. Na taj način, smanjuje se korisnikova potreba da ima otvoreno više instanci iste aplikacije kako bi radio na kompleksnijim tablicama.

Ja sam ovo riješio tako da sam dodao kontrolu koja omogućuje kretanje među više radnih listova, njihovo dodavanja, brisanje, skrivanje, otkrivanje i preimenovanje. Sve te mogućnosti dostupne su za svaki radni list u dokumentu tako da korisnik desnim klikom miša otvori izbornik iznad nekog radnog lista te odabere željenu operaciju.

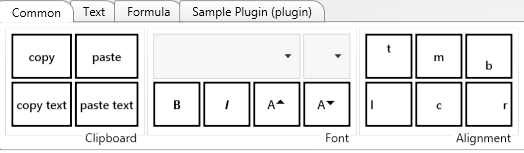


Slika 16 - Metro Tables (manipulacija radnim listovima)

Ovaj način manipulacije je vrlo raširen i nije jedinstven za ovu aplikaciju. Kako bi aplikacija zauzimala što manje radne memorije računala, nakon što se odabere neka radna lista, prijašnja se briše iz memorije, a nova se učitava. Ovo ne traje dugo pa je stoga vrlo korisno kod računala nižih specifikacija. Iako ovo unosi mali vremenski razmak između prebacivanja s lista na list, oslobođenje memorije koje ovo donosi bitnije je nego kratak vremenski razmak.

## Alatna traka

Alatna traka nalazi se u gornjem dijelu aplikacije. Ova traka sadrži sve mogućnosti do kojih korisnik može doći preko nekakvih prečica, a i više. Kako bi traka bila upotrebljiva na tablet računalima, svi gumbi i ostale kontrole malo su uvečane kako bi korisnik lakše, koristeći prst ili neki drugi način unosa, odabrao koju operaciju želi izvršiti bez previše muke. Tekst koji opisuje što koja operacija radi je napisan malim tiskanim slovima, ali velikim fontom kako bi bio vidljiv na prvi pogled.



Slika 17 - Metro Tables (alatna traka)

Alatna traka raspoređena je na glavne kategorije. Osnovne kategorije koje su dostupne bez korištenja ikakvih dodataka su „*Common“* koja sadrži načešće korištene opcije, „*Text“* koja sadrži sve opcije vezane uz tekst (veličina fonta, tip fonta, boja teksta, obrub teksta, itd.), te „*Formula*“ koja sadrži sve opcije vezane uz formule i kalkulacije unutar radnih listova.

Da bi alatna traka bila preglednija, unutar svake osnovne kategorije nalaze se i podkategorije kao što su *Clipboard*, *Font*, *Alignment*, *Borders*, itd. Na naj način dobiva se na preglednosti trake kada sadrži mnogo opcija sličnog sadržaja.

Podkategorije sadrže razne tipove gumbi i okvira za unos. Svaki gumb pokušava na svoj način dočarati njegovu funkciju koristeći neki jednostavan znak. Tako npr. za gumbe koji postavljaju poravnanje, tekst unutar gumbi je poravnan prema onome kako bi ćelija poravnana tom opcijom izgledala. ComboBox kontrola omogućuje odabir jedne stavke s liste te se kao takva koristi za odabir tipa fonta, veličinu fonta, itd.

Jedna opcija koja je ovdje da bi se korisnicima tablet računala olakšalo pregledavanje dokumenta jest mogućnost da se alatna traka u potpunosti sakrije. Ta opcija vrlo je korisna sa zaslonima male razlučivosti.

## Otvaranje već postojećih dokumenata

Kako bi aplikacija bila što korisnija, mora podržavati i formate koji nisu izvorno od iste aplikacije. Tako kako bi proširili broj korisnika koji mogu koristiti ovu aplikaciju, dodao sam podršku za pregledavanje i uređivanje Excel dokumenata. Kako Excel koristi već spomenuti *OpenDocument* standard, nije previše teško izraditi biblioteku koja čita i zapisuje u Excel formatu. Kako ne bi pisao zamorne prevoditelje sadržaja u XML standard, poslužio sam se već postojećim rješenjem. Biblioteka koju sam koristio naziva se *GemBox.Spreadsheet* [15]. Ova bibliotka omogućuje jednostavno pristupanje svim poljima i radnim listovima unutar Excelovog dokumenta, pa tako i omogućuje zapisivanje u taj dokument. Kontrola je vrlo jednostavna za korištenje. Potpuna dokumentacija kako se koristi ova bibloteka dostupna je na njihovoj stranici. Inače, ova biblioteka je komercijalna pa se korištenje plaća, ali nude i besplatnu verziju koja ima par ograničenje – maksimalno pet radnih listova po dokumentu i sto linija po radnom listu. Kako je ova aplikacija samo ogledni primjerak, nema potrebe za večim kapacitetom. Da bi se počela koristiti potpuna verzija ove bibloteke, potrebno je samo dobiti licencu i preuzeti potpunu verziju biblioteke i spremna je za rad. Promjene u aplikaciji nisu potrebne da bi se koristila potpuna verzija.

## Formule

Formule su u aplikaciji za proračunske tablice najosnovniji dio. Kako niti jedna aplikacija za proračunske tablice nije bez njih, tako ih podržava i moja aplikacija. Oko formula sam imao najviše problema jer sam se prvi puta susreo s ovakvim tipom obrade podataka.

Osnova formula jest kontrola za pisanje formula bez koje formule ne bi imale smisla. Ta kontrola nalazi se između alatnog izbornika i polja ćelija.

C:\Users\Aleksandar\SkyDrive\Documents\JustBuild\Projects - Software\Project - Metro Tables\Documentation\Screens\App\AppFormulaBar.PNG

Slika 18 - Metro Tables (kontrola formula)

Kako formule mogu imati vrlo raznolike oblike, jednostavan prolaz formulom i računanje nije dovoljno. Morao sam implementirati 3 komponente od kojih svaka izvršava vrlo važan zadatak – *lexer*, *parser* i *evaluator*. Kako se ove tri komponente mogu kasnije upotrebiti za neki drugi projekt, omogućio sam da se operacije i operandi mogu prilagoditi vlastitim potrebama. Tako operacije i operandi nisu integrirani u niti jednu od ovih tri komponenti, već se koriste kao zasebne biblioteke.

*Lexer* je komponenta koja vrši leksičku analizu unesene formule. Leksička analiza znači da kada se *lexeru* dade formula, on mora znati razdvojiti operacije od operandi. Tako je npr. ako je zadana formula „= 3 + 2“, *lexer* mora ovaj izraz razdvojiti na *tokene -* kako je navedeni u tablici.

|  |  |
| --- | --- |
| Token | Tip tokena |
| = | Početak |
| 3 | Operand |
| + | Operator |
| 2 | Operand |
| end | Kraj |

Tabela 1 - Metro Tables (primjer leksera)

Kao što se vidi iz primjera, *lexer* ne zna o čemu se radi, ali zna razlikovati operacije od operandi. Da bi to bilo moguće, moraju se zadati definicije. Definicije su u mom slučaju zadane u obliku *regular expressiona,* te opisuju sve moguće znakove koji se mogu pojaviti u formuli. Ukoliko se pojavi znak koji ne spada u formulu, izbacuje se greška i ne nastavlja se s izračunom formule.

Nakon leksičke analize formule, dobiva se lista *tokena*. Ti *tokeni* govore *parseru* o čemu se radi i što da provjerava. Kako smo *parser* implementirali tako da je modularan, prilikom podizanja aplikacije potrebno je učitati dostupne biblioteke operanada i operacija. U našem slučaju implementirane su sljedeće operacije i operandi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vrsta operacije** | **Naziv operacije** | **Simbol operacije** | **Redosljed** | **Primjer** |
| Aritmetički operatori | Zbrajanje | + | 2 | = 2.5 + 125.2333 |
| Oduzimanje | - | 2 | = 2.5 - 98 |
| Množenje | \* | 3 | = 144 \* 98 |
| Dijeljenje | / | 3 | = 10 / 3 |
| Potenciranje | ^ | 4 | = 2^14 |
| Modus broja | % | 5 | = 512 % 2 |
| Operatori usporedbe | Jednako/Rezultat | = | 1 | = 25 = 36 |
| Veće | > | 1 | = 144 > 2^16 |
| Manje | < | 1 | = 1 < 0 |
| Veće ili Jednako | >= | 1 | = 788 >= 920/4 |
| Manje ili Jednako | <= | 1 | = 455 <= 455 |
| Različito | <> | 1 | = 125 <> 5^3 |
| Spajanje teksta | Spoji | & | 0 | = „Foo“ & „Bar“ |

Tabela 2 - Metro Tables (tabela implementiranih operacija)

|  |  |  |
| --- | --- | --- |
| **Naziv operanda** | **Podržane operacije** | **Primjer** |
| Decimal | |  | | --- | | + | | - | | \* | | / | | ^ | | % | | = | | > | | < | | >= | | <= | | <> | | -2,2350012411547889  -2,33E+25  -2,33E25 |

Tabela 3 - Metro Tables (tabela implementiranih operanada)

Sve ostale potrebne operande i operacije moguće je jednostavno i naknadno implementirati, ali u ovom slučaju, dovoljno je i da se mogu parsirati malo kompliciranije matematičke formule.

Sada kada imamo sve tokenei učitane sve dostupne operacije i operande, možemo započeti s parsiranjem. Parsiranje se vrši tako da odlazimo do sljedećeg tokena te ispitujemo da li je on operacija ili operand te pozivamo za svaku implementiranu operaciju i operand metodu koja nam govori dali je *token* tog tipa. Kada se dođe do tipa koji se podudara s tokenom, on se parsira odnosno pretvara u taj tip i stavlja na listu parsiranih vrijednosti. Ovo se ponavlja tako dugo dok se ne dođe do završnog tokena odnosno tokena *Kraj.*

Kada sve tokene parsiramo, formulu spremamo u ćeliju kako se pri ponovnom evaluairanju ne bi trebala leksički obrađivati i parsirati. Ukoliko želimo dobiti vrijednost, odnosno rezultat neke formule, moramo je evaluirati. To vrši *evaluator*, ali prije same evaluacije, potrebno je listu parsiranih vrijednosti sortirati. Ovo sortiranje koristi se skoro kod svih matematičkih evaluatora (npr. džepni kalkulator). *Reverse Polish notation* [34] je oblik matematičkog zapisa gdje se operandi stavljaju prije operacija kako bi se izbjeglo korištenje zagrada. Da bi implementirali ovaj matematički zapis, koristio sam *Shunting-yard algoritam* [37]. Detalji argoritma opisani su u dokumentu. Rezultat ovog sortiranja jest formula bez zagrada, odnosno raspored operacija i operanada takav da se ne trebaju koristiti zagrade, a da se zadrži poredak računskih operacija.

Kada imamo ovako sortiranu listu parsiranih vrijednosti, možemo početi s evaluacijom. Evaluacija je sada vrlo jednostavna. Na praznu listu se stavljaju operandi sve dok se ne dođe do operacije. Operacija makne s liste potrebne operande te stavi na listu operanada rezultat operacije. To se ponavlja tako dugo dok na listi parsiranih vrijednosti ne ostane samo jedna operacija, odnosno rezultat formule.

Na taj način dobili smo jednostavan, modularan i matematički ispravan *evaluator* formula. Ovo mi je oduzelo mnogo vremena jer nisam bio upoznati s niti jednim od ovih koncepata. Najviše vremena mi je ipak na kraju oduzela modularnost parsiranja jer nisam mogao naći nikakav primjer takvog nečeg. Funkcije se mogu implementirati koristeći vanjske biblioteke, bez potrebe za mijenjanjem aplikacije.

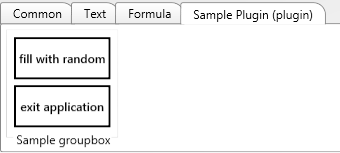
## Dodaci za aplikaciju

Još jedan od ciljeva ovog projekta je bio izraditi aplikaciju koja se može prilagoditi korisniku. Da bi se to postiglo, na neki način, korisnik mora moći dodati željene opcije u aplikaciju bez potrebe da se aplikacija izdaje za svakoga korisnika zasebno. Da bi to postigao, koristio sam MEF (eng. Managed Extensibility Framework).

Pisanje dodataka za alatni izbornik sam implementirao, a mogu se dodati i druge mogućnosti kao što su dodaci formata spremanja, dodaci kontekstualnog izbornika, dodatne teme, itd.

Dodaci za alatni izbornik pišu se jednako kao i svaka kontrola. Jedina razlika je ta što ovi dodaci moraju primati objekt koji odgovara kontroleru radnih listova. Preko tog kontrolera, dodatak komunicira s radnim listovima. Kako je dodatku poslan samo objekt kontrolera radnih listova, ne i objekt cijele aplikacije, dodatak ima samo određene ovlasti, pa tako npr. dodatak ne može zatvoriti aplikaciju, ali može obrisati sadržaj radnog lista. Stoga se ne smiju dodavati dodaci nepoznatog porijekla ili sumljive naravi.

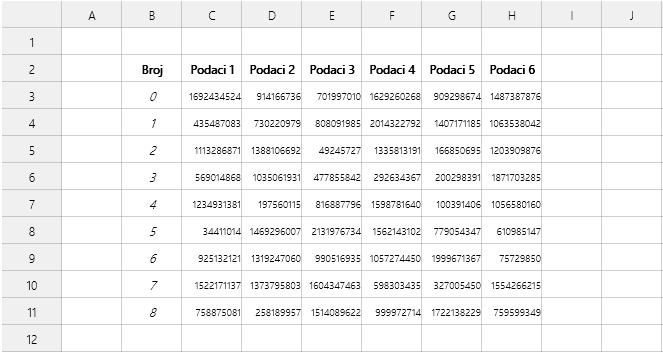
Da bi izradili i uklopili sučelje dodatka sučelju aplikacije, dodacima je dostupna klasa sa svim definicijama stilova koje koristi aplikacija. Tako dodatak može izgledati kao dio aplikacije.



Slika 19 - Plugin (primjer sučelja)

Kao što se vidi sa slike (Slika 19), sučelje dodatka potpuno je identičnog stila kao i ostatak alatnog izbornika, te se kao takvo po ničemu ne razlikuje od ostalih kontrola na izborniku.

Ovaj dodatak je samo primjer kako se pristupa polju ćelija. Tako se na slikama (Slika 19 i Slika 20) vidi da je polje zaista ispunjeno nasumičnim brojevima, a da gumb za zatvaranje aplikacije ne radi jer dodatak nema dovoljno ovlasti.



Slika 20 - Plugin (primjer nasumične ispune)

# Zaključak

Za ovaj zadatak odlučio sam se zbog toga što me zanimalo kako bi bilo izraditi aplikaciju koju inače rade cijeli timovi ljudi. Uvidio sam da je to zaista naporan zadatak i da treba paziti na mnogo detalja, te da se kao pojedinac ne mogu niti približiti mogućnostima drugih alata koji su u razvoju već par godina, pa i duže.

Ovo je ipak bilo vrlo zanimljivo iskustvo i mnogo toga sam naučio. Za koncepte leksičke obrade, parsiranja i evaluiranja nisam niti znao da postoje, već sam mislio da se sve to na neki način jednostavno radi. Sad otprilike znam kako to zapravo radi.

Tu su još i ekstenzije, odnosno dodaci za aplikaciju s kojima nikad prije nisam imao dodira. Stvarno me iznenadila jednostavnost izrade i implementacije dodataka koristeći MEF biblioteku, te se radujem novom zadatku gdje ću moći upotrijebiti ova nova znanja.

Na početku sam krenuo s izradom normalne aplikacije za izradu proračunskih tablica. Ta aplikacija je bila doslovno kopija Microsoftovog Excel alata za izradu proračunskih tablica. Ali želio sam više. Krenuo u izradu aplikacije za tablet računala jer mi nije imalo smisla raditi nešto što već postoji i zasigurno je mnogo bolje nego što moja aplikacija može biti. Zato sam se oslonio na inovacijski dio pa sam tako pokušavao dodati svoje nove ideje o tome kako bi to trebalo izgledati umjesto da kopiram što već postoji.

Bilo je zanimljivo iskustvo raditi na takvom projektu i radujem se njegovom razvoju. Nakon obrane rada, namjeravam projekt objaviti javnosti, pa možda privučem i neke pomagače.

# Literatura

## Knjige

Niti jedan primjer kôda nije korišten. Knjige nusu nužno direktno povezane uz rad, već su mi pomogle, sada ili prije, u savladavanju osnovnih koncepata izrade ovog tipa aplikacija.

1. Colin Campbell i dr. Parallel Programming with Microsoft .NET  
   Microsoft Corporation, 2010.
2. Jesse Liberty, Donald Xie. Programming C# 3.0  
   O'Really Media inc., 2008.
3. Slaven Sarić. LINQ  
   Zagreb: Element d.o.o., 2008.

## Osnovne internetske stranice

Ove stranice korištene su samo za opće pretrag. Sve vršene pretrage na navedenim stranicama nisu striktno vezane za temu radnje niti su njen sastavni dio.

1. Bing  
   <http://www.bing.com/>. (1.11.2011. – 24.5.2012.)
2. MSDN  
   <http://msdn.microsoft.com/en-us/>. (1.11.2011. – 24.5.2012.)
3. Stack Overflow  
   <http://stackoverflow.com/>. (1.11.2011. – 24.5.2012.)
4. Wikipedia  
   <http://www.wikipedia.org/>. (1.11.2011. – 24.5.2012.)
5. Wordpress  
   <http://wordpress.com/>. (1.11.2011. – 24.5.2012.)

## Ostali internetski izvori

Svi navedeni izvori pristupani su od 1.11.2011. do 24.5.2012.

1. Building a Parse Tree  
   <http://www.visualbasicask.com/visual-basic-general/building-a-parse-tree.shtml>
2. Charles Sturd University - Education and Training Team. Advances Excel Formulas and Functions  
   (11.2005.) [http://www.scribd.com/doc/15781250/Advanced-Excel-Formulas-and-Functions](http://www.scribd.com/doc/15781250/Advanced-Excel-Formulas-and-Functions%20)
3. Depth-first search  
   <http://en.wikipedia.org/wiki/Depth-first_search>
4. Drew Miller. A Simple Lexer in C# That Uses Regular Expressions  
   <http://blogs.msdn.com/b/drew/archive/2009/12/31/a-simple-lexer-in-c-that-uses-regular-expressions.aspx>
5. Emerita Carol E. Wolf, prof. Infix to postfix conversion algorithm  
   (Jesen 2009.) <http://csis.pace.edu/~wolf/CS122/infix-postfix.htm>
6. Equation (expression) parser with precedence?  
   <http://stackoverflow.com/questions/28256/equation-expression-parser-with-precedence>
7. Excel C# / VB.NET component – read / write XLS, XLSX, ODS, CVS, HTML with GemBox.Spreadsheet  
   <http://www.gemboxsoftware.com/spreadsheet/overview>
8. Excel functions (by category) – Excel – Office.com  
   <http://office.microsoft.com/en-us/excel-help/list-of-worksheet-functions-by-category-HP005204211.aspx>
9. John McNamara. Formula  
   <http://cpan.uwinnipeg.ca/htdocs/Spreadsheet-WriteExcel/Formula.pm.html>
10. Johnny Westlae. JohnnyWestlea – YouTube  
    <http://www.youtube.com/user/JohnnyWestlae?feature=watch>
11. Ken Friis Larsen. Grammars and parsing\* with C# 2.0  
    (2006-08-31) <http://www.itu.dk/people/kfl/parsernotes.pdf>
12. Kennedy Mathematics Department Santa Monica Collage. Parse Trees  
    <http://homepage.smc.edu/kennedy_john/PARSETREES.PDF>
13. Lexical analysis – Wikipedia, the free encyclopedia  
    <http://en.wikipedia.org/wiki/Lexical_analysis>
14. Managed Extensibility Framework CodePlex page  
    <http://mef.codeplex.com/>
15. Managed Extensibility Framework  
    <http://en.wikipedia.org/wiki/Managed_Extensibility_Framework>
16. Open Document | Online Community for the OpenDocument OASIS Standard  
    <http://opendocument.xml.org/>  
    <http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf>
17. Overview of formulas – Excel – Office.com  
    [http://office.microsoft.com/en-us/excel-help/overview-of-formulas-HP010342750.aspx?CTT=1](http://office.microsoft.com/en-us/excel-help/overview-of-formulas-HP010342750.aspx?CTT=1%20)
18. Parse stack into a binary tree?  
    <http://stackoverflow.com/questions/4149816/parse-stack-into-a-binary-tree>
19. Parse tree  
    <http://en.wikipedia.org/wiki/Parse_treeJohn>
20. Parser How To in .NET  
    <http://stackoverflow.com/questions/1574723/parser-how-to-in-net>
21. Parsing with Derivatives  
    <http://www.youtube.com/watch?v=ZzsK8Am6dKU>
22. Pete Jinks. Infix, Postfix and Prefix  
    (26.10.2004.) <http://www.cs.man.ac.uk/~pjj/cs2121/fix.html>
23. Pete Jinks. Parse Trees  
    (26.10.2004.) <http://www.cs.man.ac.uk/~pjj/cs212/ho/node8.html>
24. Poor man's „lexer“ for C#  
    <http://stackoverflow.com/questions/673113/poor-mans-lexer-for-c-sharp>
25. Recursive descent parser  
    <http://en.wikipedia.org/wiki/Recursive_descent_parser>
26. Reverse Polish notation – Wikipedia, the free encyclopedia  
    <http://en.wikipedia.org/wiki/Reverse_Polish_notation>
27. Robert Stehwien. Using ANTLR to create an Excel Like Formula Parser in Flex  
    (4.2008.) <http://arcanecoder.blogspot.com/2008/04/using-antlr-to-create-excel-like.html>
28. Scott Dattalo. Algebraic/RPN Parser in C++  
    (17.4.2001) [http://www.da1ttalo.com/parser.html](http://www.dattalo.com/parser.html)
29. Shunting-yard algorithm  
    <http://en.wikipedia.org/wiki/Shunting_yard_algorithm>
30. Smart design of a math parser?  
    <http://stackoverflow.com/questions/114586/smart-design-of-a-math-parser>
31. Tree traversal  
    <http://en.wikipedia.org/wiki/Tree_traversal>
32. txt2re: headache relief for programmers :: regular expression generator  
    <http://txt2re.com/index-csharp.php3?s=123456%20eee%20123.5e12/125-6&11>
33. Vector Excel  
    <http://www.nsl.com/papers/excel.htm>
34. Voliatile Excel Functions – Decision Models  
    <http://www.decisionmodels.com/calcsecretsi.htm>

# Prilozi

Prilog 1 – Kôd dinamičkog upravljanja ćelijama

/// <summary>

/// Gets or sets cell

/// </summary>

/// <param name="rowIndex">Vertical position of cell</param>

/// <param name="columnIndex">Horizontal position of cell</param>

/// <returns>Returns Cell at given position or creates</returns>

public Cell this[Int32 rowIndex, Int32 columnIndex] {

get {

// Check if given row and column indexes are not in range

// of current collection or cell on that position is null

if (this.Cells.Count <= rowIndex ||

this.Cells[rowIndex].Count <= columnIndex ||

this.Cells[rowIndex][columnIndex] == null) {

// Create new cell on that position

this[rowIndex, columnIndex] = new Cell(rowIndex, columnIndex);

}

// Return cell on given position

return this.Cells[rowIndex][columnIndex];

}

set {

// Create needed space for rows if needed

while (this.Cells.Count <= rowIndex)

Cells.Add(new List<Cell>());

// Create needed space for columns if needed

while (this.Cells[rowIndex].Count <= columnIndex)

Cells[rowIndex].Add(null);

// Create row definitions if needed

while (Sheet.Grid.RowDefinitions.Count <= rowIndex) {

RowDefinition rowDefinition = new RowDefinition() {

Height = new GridLength(Sheet.DefaultCellHeight)

};

Sheet.Grid.RowDefinitions.Add(rowDefinition);

}

// Add column definitions if needed

while (Sheet.Grid.ColumnDefinitions.Count <= columnIndex) {

ColumnDefinition columnDefinition = new ColumnDefinition() {

Width = new GridLength(Sheet.DefaultCellWidth)

};

Sheet.Grid.ColumnDefinitions.Add(columnDefinition);

}

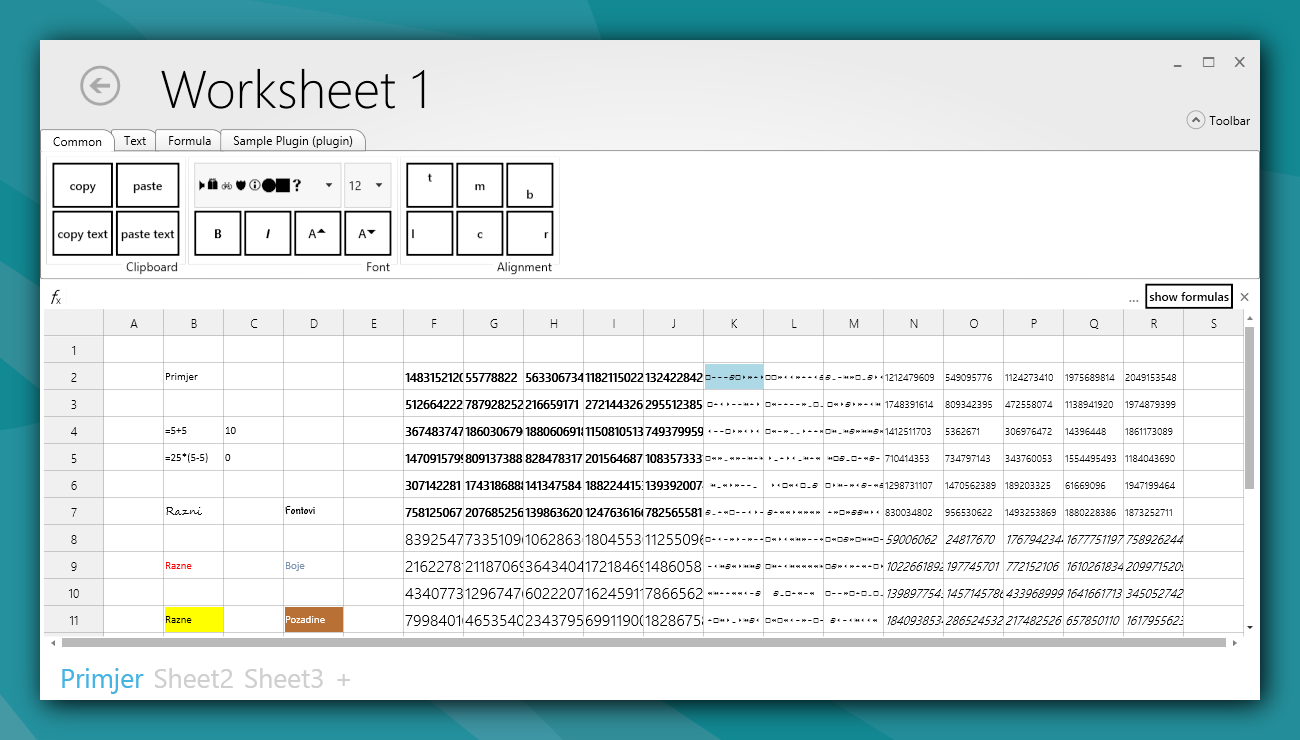
// Add cell to collection and cells grid

Cells[rowIndex][columnIndex] = value;

Sheet.Grid.Children.Add(Cells[rowIndex][columnIndex]);

}

}

**

Prilog 2 - Metro Tables primjer

Ova stranica je namijerno ostavljena prazna.

# Dodaci

**EVIDENCIJSKI LIST IZRADE ZAVRŠNOG RADA**

Ime i prezime učenika : ­­­­­­­­­­­­­­­­­\_Aleksandar Toplek\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Razred : \_4.ET2\_\_\_\_\_\_\_\_\_

Program – zanimanje : \_Tehničar za računalstvo\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mentor : \_Krešimir Kočiš, prof.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **DATUMI KONZULTACIJA** | **SADRŽAJ RADA** | **POTPIS MENTORA** |
| 24.01.2012. | Savjetovanje o općem izgledu radnje i njenim funkcijama. |  |
| 14.02.2012. | Kako koristiti Ribbon traka kao izbornik te učitavanje excel datoteka koristeći GemBox.Spreadsheet bibloteku |  |
| 03.04.2012. | Ideja za novo sučelje namjenjeno korisnicima tableta te izrada animacija u WPF-u koristeći Expression Blend |  |
| 17.04.2012. | Izrada dodataka za aplikaciju te njihovo korištenje unutar aplikacije pomoću MEF biblioteke |  |
| 08.05.2012. | Dinamičko učitavanje dodataka tokom rada aplikacije, optimizacija polja talice, unos podataka i izračunavanje formula. Razgovor o izgledu dokumentacije radnje. |  |