Analiza i usporedba tehnologija za izradu grafičkih korisničkih sučelja u stolnim aplikacija

Kurtanjek, Karlo

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Karlo Kurtanjek

Analiza i usporedba tehnologija za izradu grafičkih korisničkih sučelja u stolnim aplikacijama

ZAVRŠNI/DIPLOMSKI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Karlo Kurtanjek

Matični broj: 0016152345

Studij: Informacijski i poslovni sustavi

Analiza i usporedba tehnologija za izradu grafičkih korisničkih sučelja u stolnim aplikacijama

ZAVRŠNI/DIPLOMSKI RAD

Mentor/Mentorica:

Doc. dr. sc. Marko Mijač

Varaždin, ožujak 2024.

Karlo Kurtanjek

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog rada jest primjena različitih .NET tehnologija za izradu grafičkih sučelja u stolnim aplikacijama. Rad je strukturiran u teorijski i praktični dio. U teorijskom dijelu detaljno su obuhvaćeni tradicionalni pristupi za izradu sučelja kao što je Windows Forms, ali i suvremena rješenja kao što su WPF (Windows Presentation Foundation) i .NET MAUI, koji omogućuje razvoj aplikacija na više platformi. Nadalje, provedena je analiza i usporedba mogućnosti između tehnologija ističući njihove prednosti i ograničenja. Uz navedeno, obrađena je i cjelina vezana za smjernice za rad s tehnologijama te iskorištavanje njihovog potencijala u procesu izrade grafičkih sučelja. U praktičnom dijelu korišten je alat Visual Studio u kombinaciji s programskim jezikom C# kako bi se implementirala tri grafička sučelja za istu aplikaciju, koja su izrađena koristeći različite .NET tehnologije. Nakon implementacije, temeljem stečenog iskustva provedena je analiza i usporedba odabranih .NET tehnologija kako bi se istaknule njihove prednosti i nedostaci u njihovoj primjenjivosti u kontekstu razvoja grafičkog sučelja. Uz završni rad, priložena su grafička sučelja.

Ključne riječi: grafička sučelja; .NET tehnologija; dizajn sučelja; usporedba alata; stolna aplikacija

Sadržaj

1. Uvod

U današnjem svijetu razvoj grafičkih sučelja u stolnim aplikacijama postaje sve zahtjevniji i kompleksniji. S obzirom na vrlo velik broj dostupnih mogućnosti, alata i tehnologija, programeri se suočavaju s izazovima kao što je odabir i implementacija najprikladnije tehnologije i alata kako bi se zadovoljile sve potrebe i zahtjevi krajnjih korisnika aplikacija. Osim stalnog napretka i razvoja novih tehnologija, također rastu i zahtjevi korisnika. S obzirom na sve veću konkurenciju, potrebno je izraditi grafičko sučelje koje će zadovoljiti potrebe korisnika, ali i privući ga i zadržati za daljnje korištenje.

Kako bi se postiglo kvalitetno i zadovoljavajuće grafičko sučelje, .NET ekosustav pruža razne tehnologije i alate. U sklopu .NET ekosustava u ovom radu se obrađuju Windows Form, Windows Presentation Foundation i .NET MAUI tehnologije. U ovom radu glavni cilj je teorijski opisati svaku od navedenih tehnologija te u praktičnom dijelu nakon njihove implementacije napraviti usporedbu između njih. Osim opisa tehnologija, naglasak je i na smjernicama korištenja svake od tehnologija kako bi se iskoristio njihov puni potencijal. Takve smjernice su vrlo korisne s obzirom da svaka tehnologija ima svoj način korištenja te je nemoguće postići kvalitetne rezultate ukoliko se svaka tehnologija koristi na jednak način. Kod praktičnog dijela prikazati će se mogućnosti te izrada grafičkog sučelja pomoću Windows Forms, Windows Presentation Foundation i .NET MAUI tehnologija.

Motivacija za odabir ove teme jest osobni interes za razvojem korisničkih sučelja, ali i njihova važnost i potreba za razumijevanjem tijekom same izrade i zadovoljavanja potreba korisnika. Glavna svrha ovog rada jest istaknuti prednosti i nedostatke .NET tehnologija, ali i pružiti uvid u njihovu primjenu i prepoznavanje odgovarajuće tehnologije u skladu s određenim zahtjevima.

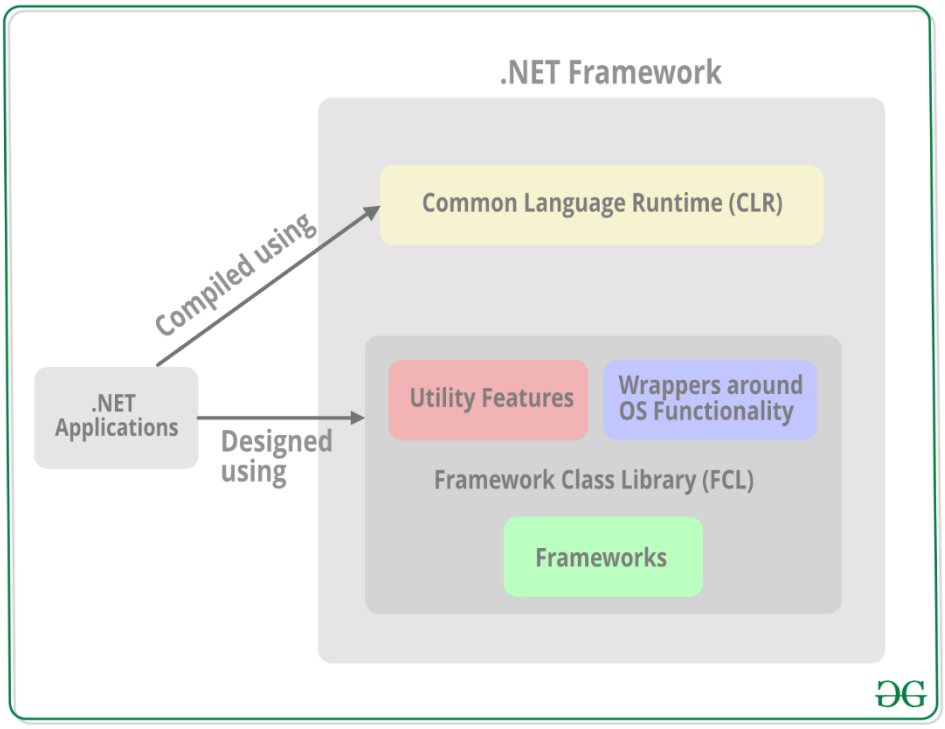
2. .NET tehnologija

.NET tehnologija softverski okvir otvorenog koda kojeg je predstavio Microsoft te služi za izradu i razvoj desktop, web i mobilnih aplikacija koje se mogu izvoditi na bilo kojem operativnom sustavu. Ova tehnologija je sadašnjost, ali i budućnost razvoja aplikacija temeljenih na oblaku.

Prva verzija .NET okvira 1.0 objavljena je 2002. godine koja je razvijena od strane Microsofta. Cijeli .NET ekosustav je kontinuirano rastao, nudeći programerima širok spektar alata i tehnologija za razvoj aplikacija. Bitna značajka ovakve tehnologije jest podrška raznim programskim jezicima, kao što su C#, VB.NET i F#.

Glavni elementi .NET tehnološkog okvira su Common Language Runtime (CLR) i Framework Class Library (FCL). CLR je ključna komponenta za izvođenje svih programa koje podržava .NET okvir. On je odgovoran za učitavanje i izvršavanje koda na siguran i učinkovit način. Također pruža usluge .NET aplikacijama, kao što su rukovanje iznimkama, sigurnost i upravljanje memorijom. Pruža strukturu za razvoj aplikacija pomoću skupa biblioteka, poznatijim pod .NET Framework Class Library. Ove biblioteke omogućuju razne funkcionalnost, kao što su mrežno povezivanje, rad s bazama podataka ili dizajn korisničkih sučelja.

FCL pruža funkcionalnosti za izvođenje i izradu različitih vrsta aplikacija. Prema članku na GeeksforGeeks ("NET Framework Class Library (FCL)", 2021), funkcionalnost Framework Class Library može se generalno podijeliti u tri glavne kategorije: pomoćne značajke napisane u .NET-u, omotače za funkcionalnosti operativnog sustava i različite okvire. Važno je napomenuti da ove kategorije nisu strogo definirane, te mnoge klase mogu pripadati više od jedne kategorije. FCL biblioteke su dizajnirane tako da budu nezavisne o platformama i da podržavaju različite programerske jezike koje se provode u .NET okviru, a temelji se na objektno orijentiranom programiranju.



Slika 1. FCL i CLR u .NET aplikacijama (.NET Framework Class Library,2021)

.NET aplikacije koriste kombinaciju FCL i CLR te time dobivaju snažnu podršku za svoj razvoj. Zahvaljujući FCL, razvijaju se razne funkcionalnosti koje čine sami temelj aplikacija. S druge strane, CLR se brine o kompajliranju, izvršavanju te upravljanju .NET aplikacija. Ovakvim spojem, .NET aplikacije imaju pouzdanu platformu za izvršavanje te implementaciju potrebnih funkcionalnosti.

2.1 Windows Forms

Windows Forms je tehnologija koja je predstavljena zajedno s prvom verzijom .NET-a i Visual Studia 2001. godine. Glavna namjena Windows Forms tehnologije je jednostavnost u razvoju stolnih aplikacija za Windows operacijske sustave, pružajući komponente za jednostavno i brzo kreiranje interaktivnih korisničkih sučelja.

(Sells i Weinhardt, 2006) navode da je Windows Forms bio jedan od prvih jednostavnih načina za pružanje komponenti grafičkog korisničkog sučelja za .NET Framework. Izgrađen je na postojećem Windows API-ju, a neke od kontrola u Windows Forms-u zapravo samo omotavaju osnovne Windows komponente.

Windows Forms je pružao korištenje jezika više razine, kao što je C#, umjesto C++ koji se tada koristio za razvoj korisničkih sučelja. To je bila svojevrsna revolucionarna promjena koja je omogućavala poduzećima da relativno brzo i lako kreiraju poslovne aplikacije.

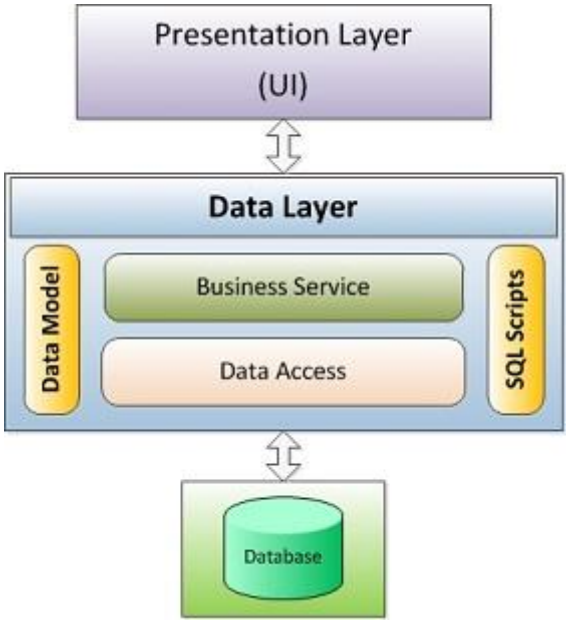
Aplikacije temeljene na Windows Forms tehnologiji vođene su događajima koje podržava .NET razvojni okvir. Ovakve aplikacije većinu vremena provedu čekajući na neki događaj koji zatim poziva određenu funkciju.

Takav događaj može biti klik na gumb koji zatim može otvoriti novu formu ili odraditi već unaprijed definiranu funkciju. Ovakvim pristupom postiže se interaktivnost korisničkog iskustva.

Kod Windows Forms aplikacija često se koristi troslojna arhitektura koja se sastoji od prezentacijskog sloja, podatkovnog sloja i sloja baze podataka (3-Tier Architecture in C# - Javatpoint, 2021.)

Prezentacijski sloj sadrži grafičko sučelje te je odgovoran za prikaz informacija korisniku. On sadrži logiku grafičkog sučelja te obično sadrži forme pomoću kojih korisnici vrše interakciju. Prezentacijski sloj je najviši sloj te je poznat kao sloj korisničkog sučelja. Podatkovni sloj nalazi se između prezentacijskog sloja i sloja baze podataka. Sadrži poslovnu logiku, odnosno procese koji upravljaju ponašanjem samog programa. Njegova uloga je protok podataka i operacija između prezentacijskog sloja koji prikazuje podatke i sloja baze podataka koji dohvaća i daje podatke.

Sloj baze podataka je odgovoran za upravljanje pohranom i dohvaćanje informacija i podataka. Sadrži logiku pristupa podacima koja zatim u komunikaciji s bazom podataka izvršava operacije poput ažuriranja, dohvaćanja ili brisanja podataka. Komponente koje su prisutne u sloju baze podataka mogu biti modeli podataka, SQL upiti, klase entiteta i slično.

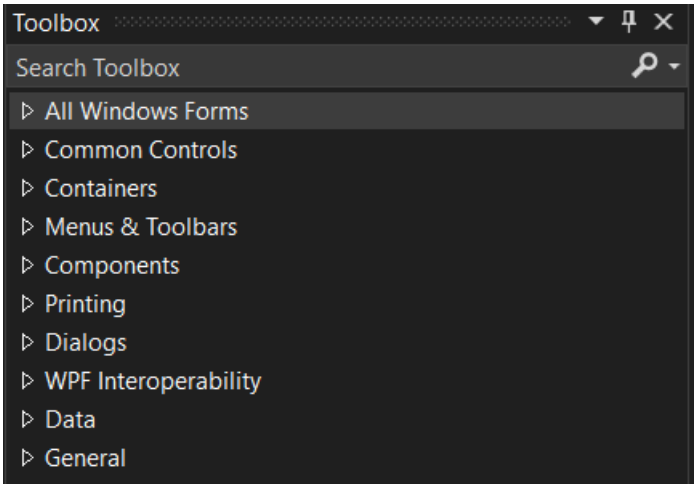


Slika 2. Arhitektura Windows Forms tehnologije (Windows Form arhitektura,2014)

Svi vizualni elementi koji su dostupni u Windows Forms proizlaze iz klase Control. Svim vizualnim elementima upravlja se pomoću metoda i svojstvima (eng. „Properties“). Metode uključuju radnje poput prikazivanja ili sakrivanja vizualnih elemenata, a za to bi koristili metode kao što su „Show()“ odnosno „Hide()“. Svojstva (eng. „Properties“) se dijele na dvije vrste:

- Svojstva za dohvaćanje vrijednosti člana klase (eng. „getter“)
- Svojstva za postavljanje vrijednosti člana klase (eng. „setter“)

Oni se koriste kako bi postavili određena svojstva vizualnih elemenata kao što je tekst, boja ili dužina elemenata.

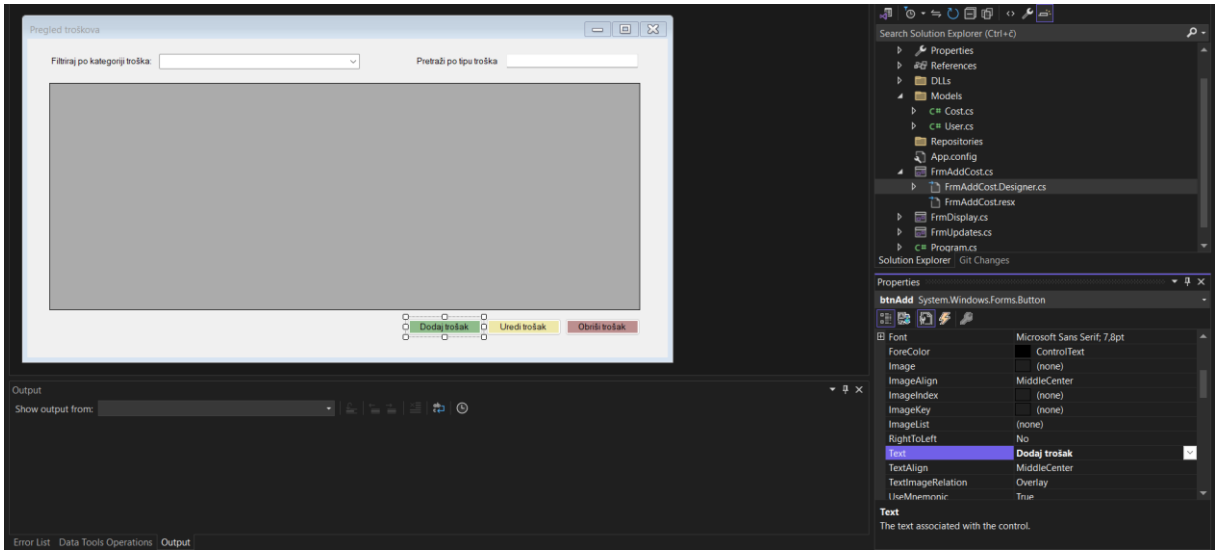


Slika 3. Vizualni elementi Windows Forms (autorski rad)

Neki od elemenata koji se najčešće koriste pri uporabi Windows Formsa su:

- Gumb (eng. Button)
- Oznaka (eng. Label)
- Tekstni okvir (eng. Textbox)
- Prikaz mreže podataka(eng. Data Grid View)

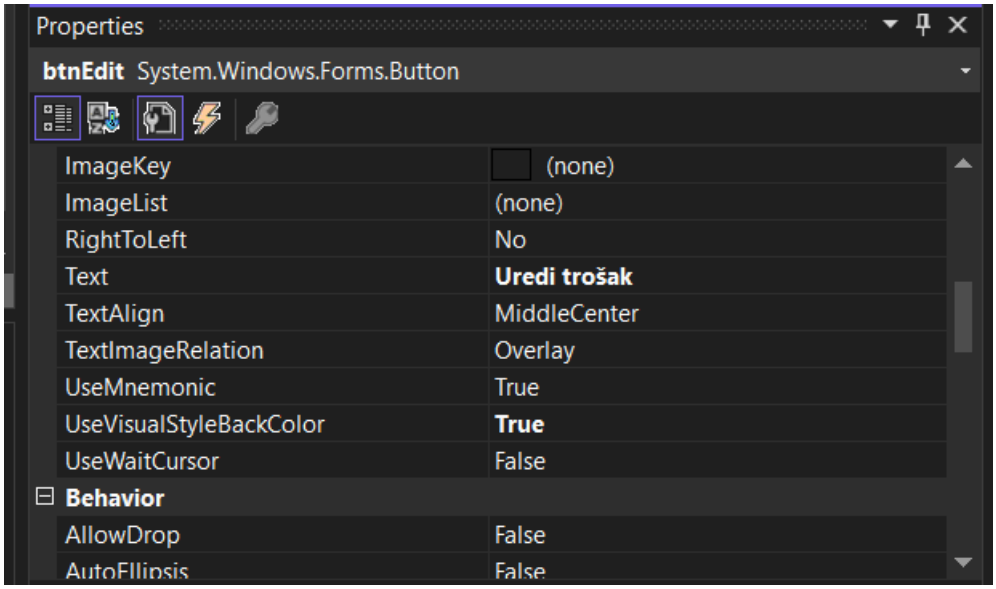
Kako bi se takvi elementi mogli urediti te prilagoditi potrebama, koristi se Windows Forms Designer. Windows Forms Designer je alat koji se koristi za upravljanje i umetanje kontrola nad vizualnim elementima te njihovo raspoređivanje prema željenom rasporedu unutar Windows Forme. Također, omogućuje dodavanje koda za kontrolu i rukovanje njihovim događajima, koji zatim implementiraju korisničke interakcije.



Slika 4. Windows Forms designer alat i svojstva za podešavanja gumba

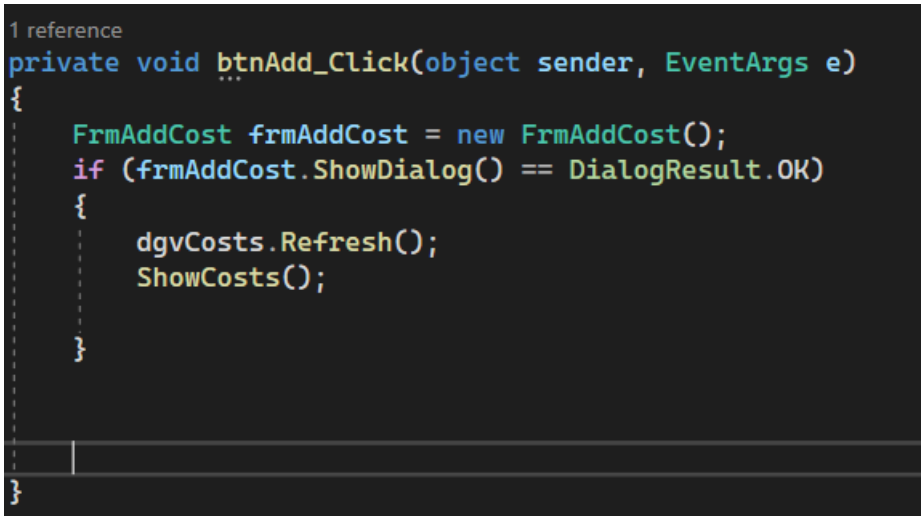
Tijekom same izrade Windows Forme, vizualni elementi se jednostavno dodaju tehnikom povuci-ispusti (eng. Drag and drop). Samim njihovim postavljanjem, vizualni elementi automatski dobivaju unaprijed određene vrijednosti i postavke. One se mogu promijeniti ručno pod karticom „Svojstva“ (eng. Properties). Također, vrijednosti i postavke mogu se promijeniti pomoću koda, odnosno tijekom izvođenja radnji korisnika ili promjena.

Na primjer, može se kreirati rukovatelj događaja koji čeka pritisak na gumb koji zatim otvara novu Windows formu. Ovakve rukovatelje događaja je moguće dodati dvoklikom na element na koji se želi dodati rukovatelj. Svaka kontrola u aplikaciji Windows Forms konkretna je instanca klase.



Slika 5. Svojstva gumba (autorski rad)

Na slici su prikazana svojstva vizualnog elementa gumba. Iako su neka od tih svojstava unaprijed definirana, poput položaja gumba ili njegove veličine, primijetimo da su neka svojstva prilagođena, kao što je tekst na gumbu koji sada glasi "Uredi trošak". Ova personalizacija omogućava prilagođavanje korisničkom iskustvu i jasno komunicira funkciju ili radnju koju gumb obavlja. U nastavku slijedi kod koji predstavlja rukovanje događajem klika na gumb nazvan "btnAdd". Kada korisnik klikne na ovaj gumb, pokreće se metoda btnAdd_Click. Ova metoda definira ponašanje koje se događa kada se dogodi klik na gumb. U ovom konkretnom slučaju, kada korisnik klikne na gumb, otvara se nova forma za dodavanje troška (`FrmAddCost`). Nakon što korisnik završi unos na ovoj formi, ažurira se prikaz tablice troškova. Ovaj rukovatelj događaja omogućava interakciju s korisničkim sučeljem i izvršava odgovarajuće radnje kada korisnik interagira s gumbom "btnAdd".



Slika 6. Rukovanje događajem (autorski rad)

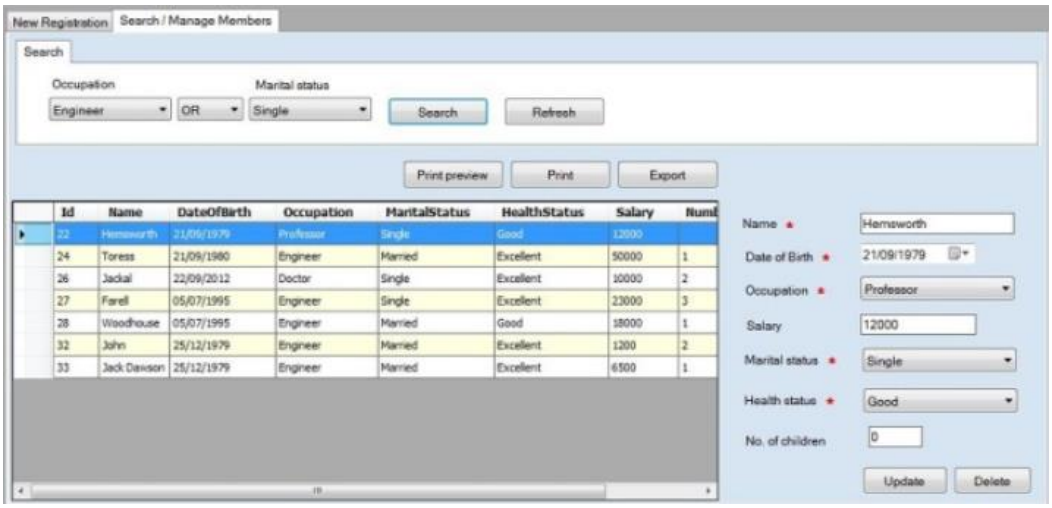
Jedna od glavnih prednosti Windows Forms tehnologije jest sama brzina kreiranja i razvoja aplikacija zahvaljujući svojoj jednostavnosti i intuitivnosti. Osim toga, ova tehnologija ima moćnu podršku biblioteke kontrola, što olakšava upravljanje i dodavanje novih elemenata u grafička sučelja. Samim tim što su takvi elementi unaprijed definirani i implementirani, omogućuju programerima da efikasno oblikuju potrebna grafička sučelja. Potrebno je istaknuti podršku za događaje koja je jedna od ključnih karakteristika Windows Forms tehnologije. Svojom jednostavnošću omogućuju jednostavnu interakciju korisnika sa grafičkim sučeljem te ga čini dinamičnim.

Osim jednostavnosti prilikom kreiranja, Windows Forms tehnologija također pruža visoku prilagodljivost stilizacije i raspoređivanja elemenata. Svojstva elemenata kao što su boja, font, veličina su lako promjenjiva te je na taj način vrlo jednostavno postići usklađenost s ostatkom dizajna aplikacije ili željenim identitetom.

Windows Forms tehnologija ima sposobnost integracije i pruža podršku kod kompatibilnosti unutar .NET ekosustava, što znači da se aplikacije bazirane na Windows Forms tehnologiji mogu integrirati s drugim tehnologijama, kao što je to Windows Presentation Foundation koja će također biti opisana u ovom radu.

Iako je Windows Forms moćan alat koji pruža brojne funkcionalnosti, potrebno je istaknuti i nedostatke koje ova tehnologija ima. Jedan od najvećih nedostataka je sigurnost, budući da WindowsForms direktno komunicira sa Windows API-jem. Nadovezujući se na izgradnju pomoću Windows API-ja, Windows Forms aplikacije i grafička sučelja imaju ograničenu prenosivost na druge platforme i operacijske sustave. Windows Forms tehnologija koristi Graphics Device Interface(GDI) za renderiranje grafike. GDI je zastarjela tehnologija koja

koristi procesor za većinu grafičkih operacija. Korištenjem GDI onemogućene su visoke performanse potrebne za moderne aplikacije koje posjeduju kompleksnija i složenija sučelja. U novijim tehnologijama koristi se DirectX koji omogućuje renderiranje grafike korištenjem grafičke kartice. S obzirom na tradicionalan i sada već zastarjeli pristup izradi grafičkih sučelja, sučelja kreirana pomoću Windows Forms tehnologije izgledaju zastarjelo i ne pružaju jednake značajke i mogućnosti u odnosu na sučelja kreirana pomoću drugih tehnologija.
Ovaj primjer prikazuje kako se upotrebom Windows Forms tehnologije uspješno može kreirati sučelje koje je jednostavno i intuitivno za korisnika, ali i sučelje koje prikazuje i manipulira podacima na efikasan način.



Slika 7. Grafičko sučelje kreirano pomoću Windows Forms (Sučelje u Windows Forms,2014)

2.2. Windows Presentation Foundation (WPF)

Windows Presentation Foundation, skraćenicom WPF, je još jedna od brojnih tehnologija koja se koristi prilikom izrade i razvoja grafičkih sučelja za stolne aplikacije. Kreirana je od strane Microsofta krajem 2006. godine te je napisana u C# programskom jeziku. Predstavljena je zajedno sa Microsoftovim .NET Frameworkom 3.0, kao zamjena za dotadašnje tehnologije poput Windows Forms. Glavni alat za razvoj i rad sa WPF tehnologijom je Visual Studio, ali se može koristiti i Expression Blend, koji omogućuje dizajniranje grafičkog sučelja putem vizualnog sučelja.
Veoma bitna značajka kod WPF tehnologije jest razdvajanje korisničkog sučelja i poslovnog modela. Ovakvim načinom rada se omogućuje dizajnerima da se fokusiraju na sami dizajn korisničkog sučelja i to koristeći XAML (Extensible Application Markup Language), jezik koji je temeljen na XML-u (Exstensible Markup Language). Tako se poslovna logika i poslovni

model prepušta programerima, koji ju izrađuju u programskim jezicima kao što su C# ili VB.NET. Također je došlo do promjene arhitekture. Ostale tehnologije koje nudi Microsoft su bili omotači (eng. „wrappers“) oko User32 i GDI32 DLL-ova, a WPF minimalno koristi User32. User32 i GDI32 su komponente koje pružaju izradu i upravljanje grafičkim sučeljima. User32 je biblioteka koja pruža funkcije za upravljanje elementima unutar grafičkog sučelja kao što je stvaranje i upravljanje prozorima, unos podataka. User32 upravlja porukama koje sučelje prima ili šalje korisniku te obrađuje događaje vezane uz korisnika (klik mišem, unos s tipkovnice). GDI32 je biblioteka koja omogućuje crtanje grafike unutar Windows aplikacija. GDI32 pruža crtanje grafike i osnovnih grafičkih oblika, kreiranjem ili odabirom različitih fontova te kreiranjem ili manipulacijom slikama. Ipak, u modernijim aplikacijama koristi se DirectX pa možemo reći kako se User32 i GDI32 biblioteke uglavnom više ne koriste.

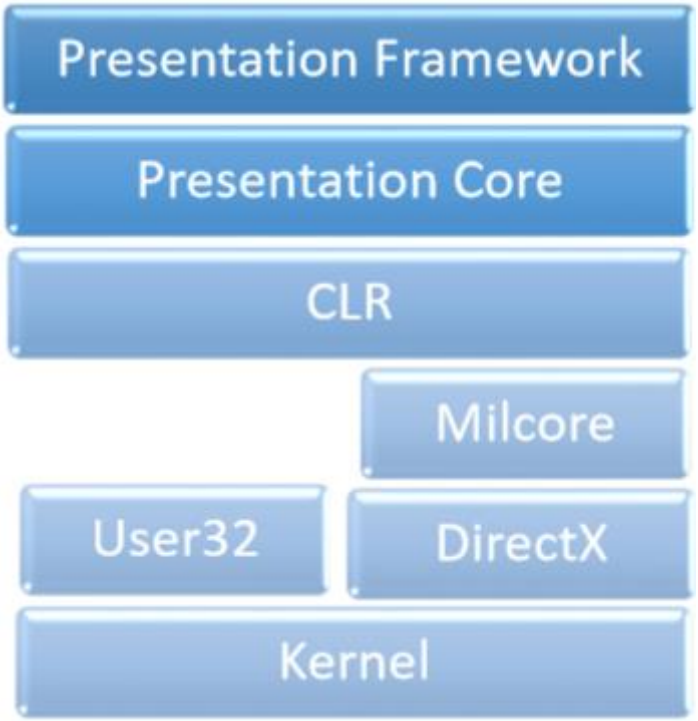
Glavne komponente WPF-a su PresentationCore, PresentationFramework, Common Language Runtime (CLR), Milcore i Kernel. Prezentacijski okvir i prezentacijska jezgra napisani su u upravljanom kodu. Prezentacijska jezgra pruža omotač (eng. „wrapper“) za Milcore te implementira značajke kao što su:

- Sustav slanja poruka putem objekta (eng. „Object-based messaging system“)
- Sustav svojstava (eng. “Property system“)
- Sustav rasporeda („Layout system“

Arhitektura WPF tehnologije sastoji se od upravljanog i neupravljanog koda te je većina WPF-a napisana u upravljanom kodu.

Milcore je napisan u neupravljanom kodu, što omogućuje integraciju s DirectX-om, koji je odgovoran za prikaz sučelja. Oni zajedno pružaju podršku za samu vizualizaciju, kao što je 2D i 3D površina, sastavljanje WPF elemenata, manipulacija sadržaja kontrolirana timerom.

Kompozicijski mehanizam u WPF-u je vrlo osjetljiv na izvedbu, zbog čega je bilo nužno odreći se mnogih prednosti koje nudi Common Language Runtime (CLR) kako bi se postigla zadovoljavajuća brzina rada (What Is WPF? - GeeksforGeeks, 2024.)

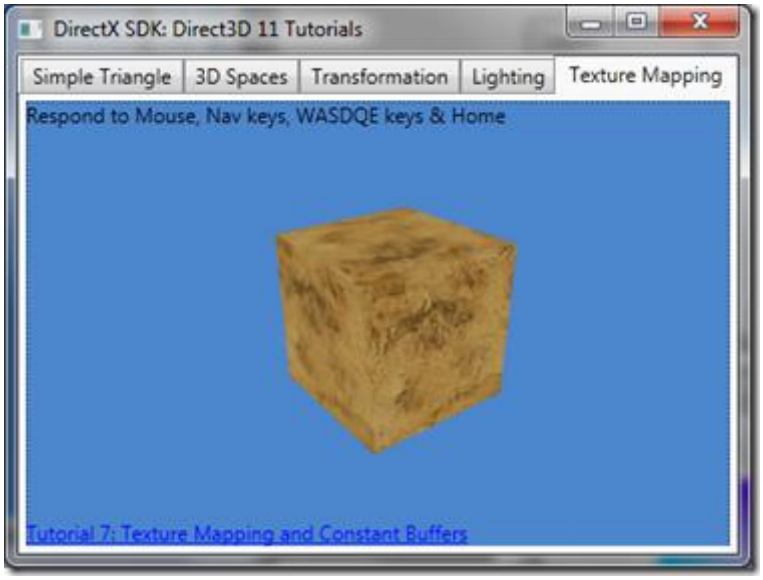


Slika 8. Arhitektura WPF tehnologije (WPF arhitektura, bez dat.)

Prezentacijski okvir implementira prezentacijske značajke za krajnje korisnike, kao što je povezivanje podataka (eng. „data binding“) ili animacije temeljene na vremenu. Common Runtime Language (CLR) pruža mnoge značajke prilikom izvođenja programa, kao što je upravljanje memorijom, sigurnost i integracija s okruženjem.

Bitna značajka koja se pojavljuje kod WPF tehnologije je Direct3D. Sva grafika, koja uključuje stavke poput gumba ili prozora, se prikazuju pomoću Direct3D. Korištenjem Direct3D-a, omogućen je prikaz bogatih i složenijih 3D grafika i efekata unutar standardnih grafičkih sučelja. Kod grafike bitnu ulogu ima i XAML koji se koristi kako bi se pružila lakša integracija i dodavanje 3D elemenata u samo sučelje. XAML se može koristiti za definiranje i stiliziranje 3D efekata, ali i animacija i interakcija.

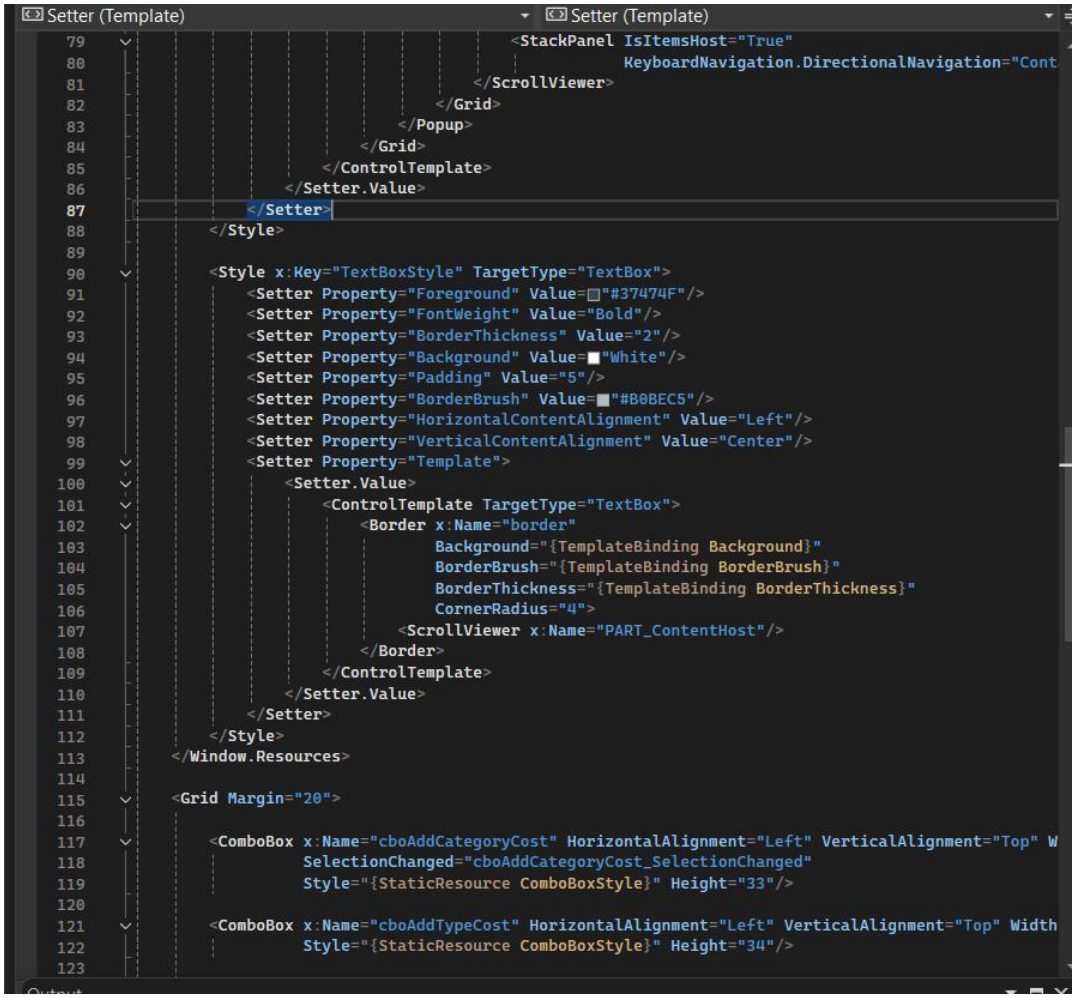
Korištenjem Direct3D, dešavaju se promjene vezane uz hardver i korištenje komponenata računala. WPF tehnologija fokus stavlja na vektorsku grafiku kojom omogućuje prenošenje zadataka na grafički procesor (GPU) koji je optimiziran za paralelna izračunavanja piksela. Zahvaljujući tome, povećava se brzina osvježavanja zaslona te se tako smanjuje i samo opterećenje procesora (CPU). Sam fokus na vektorsku grafiku omogućuje skaliranje elemenata bez da se gubi na njihovoj kvaliteti čime se poboljšava pristupačnost.



Slika 9. Prikaz 3D elementa korištenjem Direct3D (3D element u WPF, 2011)

Iako se dinamičko povezivanje podataka (eng. „data binding“) koristio i u prijašnjim tehnologijama za razvoj grafičkih sučelja, kao što su ASP.NET i Windows Forms, u WPF tehnologiji se cjelokupan koncept dinamičkog povezivanja podataka razvija i integrira zahvaljujući upotrebi XAML-a.

Integracijom povezivanja podataka i XAML-a se omogućuje jasno definiranje kako će se podaci povezati s elementima u grafičkom sučelju bez potrebe za složenim kodom. Dinamičko povezivanje podataka je važno radi kreiranja dinamičkih i interaktivnih grafičkih sučelja i aplikacija u kojima se podaci automatski ažuriraju, bez potrebe za ručnim upravljanjem istih.



Slika 10. Primjer korištenja XAML jezika za definiranje izgleda tekstualnog okvira u WPF tehnologiji

WPF tehnologija nudi četiri vrste povezivanja podataka koje omogućuju njihovo međusobno povezivanje i manipuliranje.

Jednosmjerni (eng. „one way“) – podaci se prenose sa poslužitelja na grafičko sučelje, a korisnik ima pristup samo za čitanje podataka.

Jednosmjerni prema izvoru (eng. „one way to source“) – podaci se prenose sa poslužitelja na grafičko sučelje, a korisnik ima pristup samo za pisanje podataka.

Jednokratno (eng. „one time“) – podaci se prenose sa poslužitelja na grafičko sučelje samo jednom, nakon čega se prikaz podataka na grafičkom sučelju ne ažuriraju s promjenama na poslužitelju.

Dvosmjerni (eng. „two way“) – korisnik može čitati i pisati podatke na poslužitelju. Podaci se mogu prenositi sa poslužitelja na grafičko sučelje, ali se mogu prenositi i sa grafičkog sučelja na poslužitelj što omogućuje interaktivno uređivanje podataka iz grafičkog sučelja. (Binding Mode - .NET MAUI | Microsoft Learn, 2023.)

U tablici ispod prikazane su medijske značajke koje pruža WPF tehnologija. Može se zaključiti kako su u WPF tehnologiji integrirani brojni sustavi za manipulaciju i upravljanjem medijskim elementima. Novitet kod WPF tehnologije je izrada dinamičnih grafičkih sučelja pomoću animacija te korištenje Direct3D-a za složenije vizualne prikaze. U nastavku slijedi tablica temeljena na podacima preuzetih sa Micorsoftove stranice koja detaljnije objašnjava primjenu i mogućnosti medijskih značajki koje pruža WPF tehnologija.

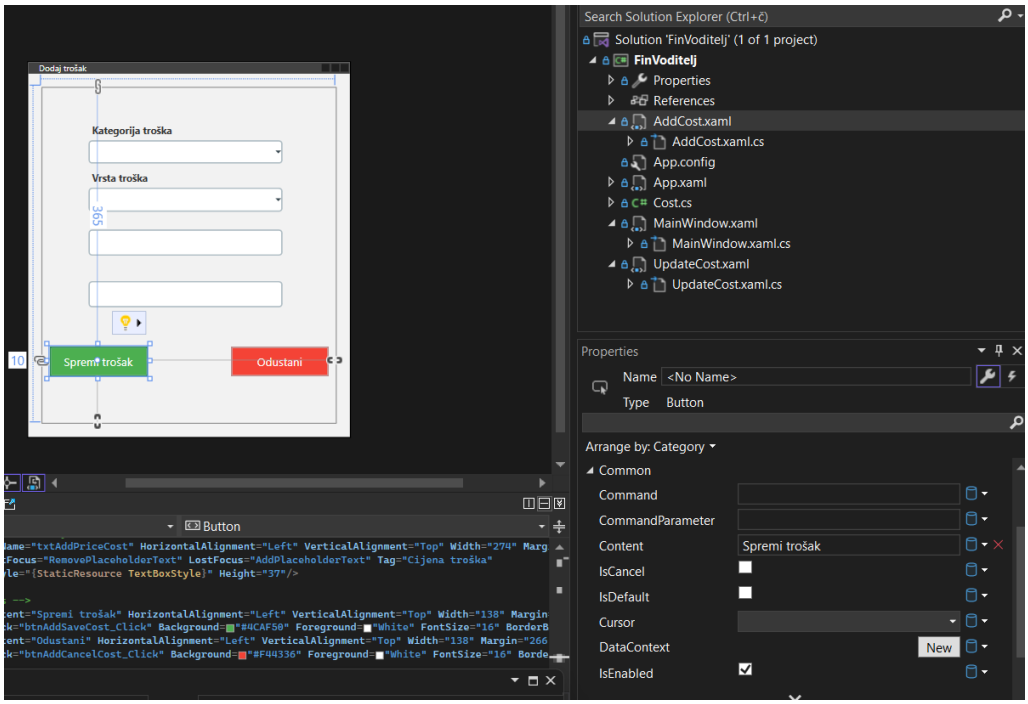
Tablica 1. Medijske značajke WPF(„Animation Overview“,“Imaging Overview“, Multimedia Overview“, 2023)

Značajka	Opis
Medijski elementi	Pruža integrirani sustav za izgradnju grafičkih sučelja s medijskim elementima poput vektorskih i rasterskih slika, zvuka i videa.
Animacija	Pruža sustav animacije za stvaranje dinamičnih i interaktivnih grafičkih sučelja.
Renderiranje 2D/3D	Pruža sustav 2D/3D renderiranja za prikaz i manipuliranje sadržajem i grafikom u dvije ili tri dimenzije.
Osnovni grafički oblici (eng. “shape primitives”)	Pruža osnovne oblike za 2D grafiku, poput kistova i olovke koji olakšavaju stvaranje crteža.
Integracija s Direct3D	Pruža podršku za 3D grafiku, zajedno sa podskupom značajki koje nudi Direct3D.
Podržani formati slika	Podržava širok spektar formata slika, najčešće korišteni: JPEG, PNG, TIFF, BMP.
Podržani video formati	Podržava WMV, AVI i MPEG datoteke uz mogućnost korištenja instaliranih kodeka na sustavu zbog korištenja Windows Media Player-a za reprodukciju videosadržaja.

Važna značajka koja oblikuje uporabu same WPF tehnologije su koncepti kontrole (eng. „control“) i predloška (eng. „templates“). Kontrole čine standardne vizualne elemente koji su vidljivi na grafičkom sučelju, poput gumba, okvira i teksta. Kontrole omogućuju programerima jednostavno dodavanje funkcionalnosti takvim elementima, poput navigacije ili interakcije, bez potrebe za samostalnim pisanjem koda. Svaka takva kontrola ima već unaprijed zadani predložak kojim su definirani vizualni identiteti i ponašanja kontrola.

Programer može takav predložak prilagoditi i promijeniti kako će se kontrola vizualno prikazivati ili kako će se kontrola ponašati u interakciji s korisnikom. Prednost korištenja predložaka jest mogućnost ponovne uporabe na različitim dijelovima grafičkih sučelja, bez potrebe za ponovnim pisanjem ili dupliranjem koda.

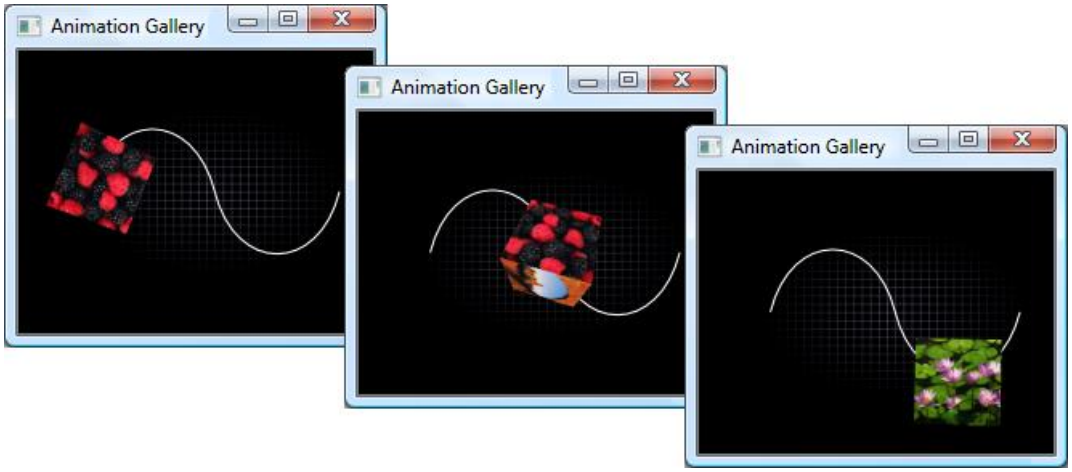
Kombinacijom koncepta kontrole i koncepta predloška, WPF tehnologija daje mogućnost za kreiranje modularnih grafičkih sučelja i aplikacija koje će biti lako nadograđivati te održavati. Kada se neki predložak ažurira, kontrole koje koriste taj predložak će se automatski promijeniti prema ažuriranom predlošku čime se postiže fleksibilnost u korištenju WPF tehnologije.



Slika 11. Primjer razvojnog okruženja za grafičko sučelje u WPF tehnologiji

Dodatna mogućnost i napredak u odnosu na svog prethodnika su 2D i 3D grafika i renderiranje. Osim što pruža biblioteke za prikaz različitih 2D oblika i efekata, WPF daje mogućnost 3D renderiranja te prikaz animacija. Animacije imaju dodatne mogućnosti koje se mogu dodati gotovo svim svojstvima, kontrolama i elementima kako bi se oni vrtjeli, tresli ili blijedjeli. S

animacijama se dodaje estetska privlačnost samom sučelju te može služiti za stvaranje zanimljivih prijelaza između stranica. U nastavku slijedi primjer animacije kreirane pomoću WPF tehnologije.



Slika 12. Animacije kreirane pomoću WPF (Animacije u WPF, 2022)

2.3 .NET MAUI

.NET MAUI je razvojni okvir za izgradnju višeplatformskih aplikacija otvorenog koda za sustave kao što su Android, Windows, macOS i iOS. MAUI je kratica od Multi-platform App User Interface te je on temeljen na Xamarinu. Xamarin je okvir za razvoj mobilnih aplikacija koji je omogućio dijeljenje i izvršavanje jednog koda na više različitih platformi. Dolazak Xamarin-a u Microsoft rezultirao je formalnom integracijom u ekosustav .NET-a, čime je omogućena službena implementacija .NET platforme za stvaranje aplikacija koje pružaju funkcionalnost na više platformi. Bitna funkcionalnost kod Xamarina je Xamarin.Forms, koja čini temelj MAUI-a.

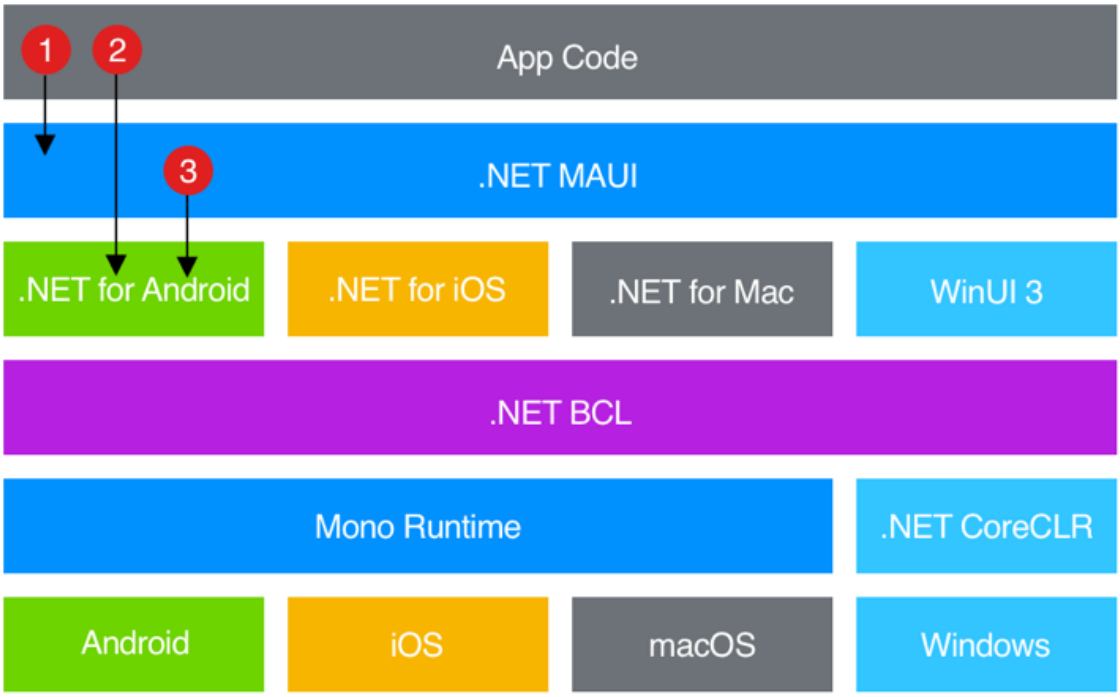
Neke od značajki koje je MAUI naslijedio od Xamarina su:

- Platformna kompatibilnost
- Integracija s .NET ekosustavom
- Prilagođene kontrole
- MVVM (Model-View-ViewModel) arhitektura

.NET MAUI službeno je izdan 23. svibnja 2022. godine. „Microsoft je dalje pojasnio da će sljedeća faza MAUI putovanja biti uvođenje dodataka, biblioteka i usluga iz .NET okvira i

ažuriranje starog projektnog sustava kako bi bio kompatibilan s .NET 6 .“ (.NET MAUI in a Nutshell — Everything You Need to Know in 2024, 2024.).

Jedna od glavnih značajki .NET MAUI tehnologije je mogućnost razvoja jednog projekta te njegova implementacija na različite platforme koje će raditi na različitim uređajima, poput stolnih računala, mobilnih uređaja ili tableta. Osim toga, pruža i dubinski pristup svakom aspektu u izvornoj platformi.



Slika 13. Arhitektura .NET MAUI aplikacije (Arhitektura .NET MAUI,2021)

Prilikom razvoja aplikacije u .NET MAUI tehnologiji kod primarno komunicira sa API-jem .NET MAUI koji zatim koristi izvorne API platforme. Također, ukoliko je to potrebno, kod aplikacije može izravno komunicirati sa API-jem platforme te sama komunikacija sa .NET MAUI API-jem tada nije potrebna.

Možemo reći kako .NET MAUI ujedinjuje API-je različitih platformi u jedan API. Takve API-je povezuje biblioteka koja je dostupna u .NET 6 okviru, koja se zove .NET 6 Base Class Library (BCL). Uloga ove biblioteke je da izdvoji pojedinosti koje su temeljene na platformi za koju se aplikacija ili grafičko sučelje razvija, a zatim omogućuje dijeljenje zajedničke poslovne logike ostalim platformama. Iako u pravilu svaka platforma ima svoj vlastiti pristup i logiku kod kreiranja grafičkog sučelja, .NET MAUI nudi jedinstveni okvir koji objedinjuje više platformi, koristeći XAML kao zajednički jezik za deklarativno definiranje korisničkog sučelja.

Izvorni paketi aplikacija mogu se kompajlirati iz .NET MAUI koda napisanog na PC-u ili Mac-u:

- kada se Android aplikacija razvija s .NET MAUI, C# se prevodi u međujezik (IL), a tijekom izvođenja, međujezik se kompajliranjem u trenutku izvođenja (eng. „Just in time“) pretvara u izvorni asembler
- Aplikacije za iOS razvijene s .NET MAUI pretvaraju se iz C# u izvorni ARM asebmlerski kod
- Za macOS, ove MAUI aplikacije koriste Mac Catalyst, Appleovu tehnologiju koja prenosi vašu iOS aplikaciju temeljenu na UIKitu na radnu površinu i poboljšava je dodatnim AppKitom i platformskim API-jima.
- Izvorni Windows desktop programi razvijeni s .NET MAUI stvoreni su uz pomoć biblioteke Windows User Interface 3 (WinUI 3). (What Is .NET MAUI?, 2024.)

Najnovija verzija .NET tehnologije koja se koristi prilikom korištenja .NET MAUI je .NET 8. Ova tehnologija omogućuje dublju interakciju s modernim tehnologijama i alatima kao što su ML.NET ili Azure AI. Također je povećana podrška za cloud tehnologije i korištenje umjetne inteligencije čime se olakšava rad programerima tokom izrade aplikacija.

Bitno je spomenuti utjecaj i važnost Microsoftove platforme .NET 6 koja donosi niz značajki i poboljšanja koje čine .NET MAUI. Sam .NET MAUI je dizajniran i osmišljen tako da bude kompatibilan te da koristi značajke i tehnologiju .NET 6, kako bi se poboljšala performansa i produktivnost kod razvoja grafičkih sučelja i aplikacija.

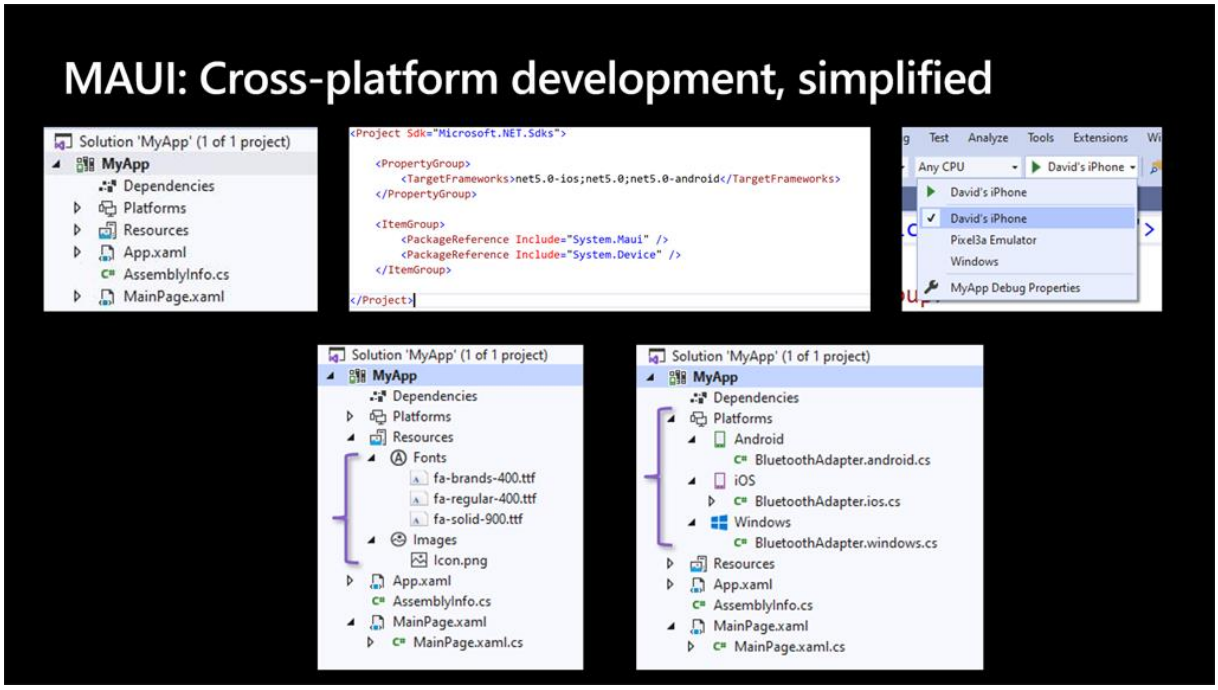
Prilikom razvoja .NET 6, fokus je bio na produktivnosti programera. Svi resursi koji se koriste prilikom izrade grafičkih sučelja potrebno je uključiti samo jednom unutar projekta, a zatim .NET MAUI uz svoje API-je povezuje resurse i na ostale platforme. Bitno je spomenuti kako .NET MAUI nudi programerima opciju između korištenja tradicionalnog MVVM (Model-View-ViewModel) arhitekturalnog uzorka ili MVU (Model-View-Update) uzorka.

MVVM model razdvaja poslovnu logiku od korisničkog sučelja. Model predstavlja podatke i poslovnu logiku, a View predstavlja korisničko sučelje. Ovakva arhitektura primjenjuje se u već prije spomenutoj, Windows Presentation Foundation tehnologiji.

MVU arhitektura često se koristi kod reaktivnog programiranja. Sastoji se od Modela koji predstavlja podatke i stanje i Viewa koji ima ulogu korisničkog sučelja. Update je funkcija čijim se pozivom prima prethodno stanje Modela te se generira novo stanje koje se prikazuje u Viewu.

Struktura projekta u .NET MAUI tehnologiji se sastoji od tri dijela.

1. Jedan projekt sa svim izvornim kodovima, uključujući grafička sučelja, poslovnu logiku, kod
2. Resursi za više platformi, kao što su slike, fontovi i ostali materijali
3. Mape specifične za određenu platformu koje sadrže resurse koji se specifično koriste za određenu platformu



Slika 14. Korištenje .NET MAUI-a za povezivanje različitih platforma (Hunter, 2020)

Poznato je kako se .NET MAUI fokusira na poboljšavanje produktivnosti tokom izrade grafičkih sučelja i aplikacija. Kako bi to i postigli, koriste se tehnologije poput Hot Reload i XAML Live Preview. Također, korištenjem njihovih značajki, ubrzava se optimizacija i iteracija ideja programera, ali i samih programa.

Hot Reload omogućuje mijenjanje izvornog koda u trenutku izvođenja programa, bez potrebe da se program prvo zaustavlja, naprave željene promjene te ponovno pokreće. Korištenjem Hot Reload-a promjene se mogu brzo primijeniti te su rezultati istih vidljivi odmah, što ubrzava sam ciklus razvoja aplikacije i grafičkog sučelja. Hot Reload podržava promjene u XAML datotekama, C# kodu, stilovima, resursima i drugim dijelovima aplikacije.

XAML Live Preview omogućuje uvid u promjene koje se naprave u XAML kodu u trenutku pisanja koda, bez potrebe za ponovnim pokretanjem programa. Korištenjem XAML Live Preview postiže se brzo i interaktivno uređivanje grafičkog sučelja u dizajnerskom načinu rada.

U nastavku slijedi primjer grafičkih sučelja kreiranih pomoću .NET MAUI tehnologije. Radi se o vremenskoj prognozi sa jednostavnom i privlačnom estetikom. Ovaj primjer prikazuje izgled sučelja na različitim uređajima kao što su mobilni uređaji i desktop računala, no u nastavku ovog rada fokus će biti na primjeni .NET MAUI tehnologije za izradu stolne aplikacije.

Slika 15. Grafička sučelja kreirana pomoću .NET MAUI tehnologije (Sučelje aplikacije, 2022)

3. Smjernice za rad s .NET tehnologijama

S obzirom na velik broj .NET tehnologija i različite načine na koje se one primjenjuju kod izrade aplikacija i grafičkih sučelja postoje različiti načini korištenja, odnosno različite smjernice koje je potrebno pratiti kako bi se .NET tehnologije pravilno upotrijebile te iskoristio njihov puni potencijal. Držanjem smjernica postiže se stvaranje visokokvalitetnih aplikacija i grafičkih sučelja.

Upravo zbog toga svaka .NET tehnologija ima svoje smjernice za rad koje bi bilo dobro pratiti kako bi se one što lakše i bolje implementirale. Kako bi korisničko iskustvo bilo što bolje, potrebno se odlučiti na kontinuiran i dosljedan dizajn kroz cijeli program. Korištenje nekoliko pažljivo odabranih boja i fontova doprinosi kontinuitetu i visokoj kvaliteti vizualnog doživljaja.

Prilikom ključnih akcija, kao što su potvrda ili odustajanje, potrebno je posebno naglasiti vidljivost tih opcija kako bi korisniku bilo jasno da se radi o važnim gumbima ili događajima.

Prema stranici Better Solutions, ovo je petnaest stavki koje bi se trebalo držati prilikom kreiranja grafičkih sučelja pomoću .NET tehnologija:

1. Ne prikazujte obrasce prozora na programskoj traci.
2. Provjerite jesu li slične kontrole iste veličine.
3. Osigurajte da dijalozi ne ostaju skriveni iza aktivne datoteke/dokumenta.
4. Provjerite jesu li kontrole ravnomjerno raspoređene.
5. Provjerite jesu li dijaloški okviri previše neodjeljivi ako podijelite kontrole na nekoliko kartica.
6. Omogućite da se svakoj kontroli može pristupiti putem tipke prečaca.
7. Provjerite nije li neka od tipki prečaca duplicirana.
8. Osigurajte da je redoslijed kartica ispravno postavljen.
9. Osigurajte da obrazac poduzima potrebnu radnju ako se odbaci ili ako korisnik pritisne tipku Escape.
10. Provjerite ima li pravopisnih pogrešaka u tekstu.
11. Osigurajte da obrazac ima odgovarajući naslov.
12. Provjerite jesu li kontrole logično grupirane.
13. Osigurajte da sve kontrole dopuštaju unos samo valjanih vrijednosti.
14. Kada imate dva okvira s popisom koji omogućuju korisniku premještanje stavki između njih, dopustite "dvostruki klik".
15. Ne omogućavajte gumbe "Spremi" ili "Spremi i zatvori" dok se obrazac ne završi.

(C# Windows Forms - Best Practices, 2024.)

3.1. Smjernice za rad s Windows Forms tehnologijom

Prilikom kreiranja Windows Forms aplikacije, vidljive su dvije datoteke, Form.cs i Form.Designer.cs. Form.Deisgner.cs datoteka sadrži kod s kontrolama forme te se ona automatski generira prilikom kreiranja aplikacije. U pravilu je uobičajeno da se kod unutar nje ne mijenja. Unutar Form.cs datoteke se piše kod, odnosno pišu se radnje i akcije koje se žele postići prilikom interakcije s formom, to jest vizualnim elementima forme. Osim toga, unutar Form.cs mogu se pisati i mijenjati sama svojstva i ponašanje vizualnih elemenata koje se nalaze na formi.

Iako ovo ne vrijedi samo kod Windows Forms tehnologiju, nego cijelu .NET razvojnu okolinu, potrebno je istaknuti standarde za imenovanje kako vizualnih elemenata, tako i elemenata programskog koda. Poznati standardi koji se često koriste jesu Pascal i Camel Case.

Prilikom izrade veće aplikacije, dobra praksa je razdvojiti različite elemente i kontrole na više različitih klasa. Posebna je važno odvojiti kod kojim se definira grafičko sučelje te njegovo ponašanje sa kodom koji pristupa bazi podataka ili mrežama. Ukoliko se ovi dijelovi ne odvoje, krši se pravilo načela jedinstvene odgovornosti (eng. „Single Responsibility Principle“). Kako bi se to izbjeglo, potrebno je kreirati različite klase, a prema potrebama se iz klase za pristupanje bazi podataka može naknadno pozvati u vizualne elemente grafičkog sučelja.

Kod dizajniranja samog sučelja pomoću Windows Forms tehnologiji, važnu ulogu u samom izgledu imaju položaji i veličine vizualnih elemenata. Položaj elemenata određeni su svojstvom Lokacija (eng. „Location“) koje imaju svoje koordinate (x0,y0). Osim lokacije, veličina je određena svojstom Veličina (eng. „Size“) i predstavlja širinu i visinu elementa.

Osim pozicioniranja i veličine elemenata, potrebno je precizno i kontinuirano postavljati elemente, a to se postiže korištenjem margina i paddinga.

Margin svojstvo određuje prostor oko kontrole, osiguravajući da su druge kontrole udaljene od njezinih granica za određeni iznos. Padding svojstvo određuje unutarnji prostor unutar kontrole, osiguravajući da je sadržaj kontrole (primjerice, tekst definiran svojstvom Text) na određenoj udaljenosti od njezinih unutarnjih granica. („Position and layout of controls (Windows Forms .NET)“,2024).

Iako se elementi mogu vlastoručno podešavati prema našim preferencijama, Windows Forms omogućuje automatsko dimenzioniranje elemenata korištenjem svojstva AutoSize. Korištenjem ovog svojstva, elementi automatski prilagođavaju svoje veličinu kada je to potrebno. U nastavku slijedi tablica preuzete sa Micorsoftove stranice koja detaljnije objašnjava primjenu AutoSize svojstva.

Tablica 2. Primjena Autosize svojstva („Position and layout of controls“,2024).

AutoSize postavljen na true	Opis
Automatsko određivanje veličine značajka je za vrijeme izvođenja.	Nikada ne raste ili smanjuje kontrolu i onda nema daljnjeg učinka.
Ako kontrola promijeni veličinu, vrijednost njenog svojstva Location uvijek ostaje konstantna.	Kada sadržaj kontrole uzrokuje njezino povećanje, kontrola raste prema desno i prema dolje. Kontrole ne rastu ulijevo.

Svojstva Dock i Sidro (eng. „Anchor“) poštuju se kada je AutoSize postavljen na true.	Vrijednost svojstva Location kontrole podešena je na ispravnu vrijednost. Kontrola oznake je iznimka od ovog pravila. Kada postavite vrijednost svojstva AutoSize usidrene kontrole Oznaka (eng.“Label“) na , elementu Oznaka (eng. „Label“) se neće rastezati.
Svojstva MaximumSize i MinimumSize kontrole uvijek se poštuju, bez obzira na vrijednost svojstva AutoSize.	Svojstvo AutoSize ne utječe na svojstva MaximumSize i MinimumSize .
Ne postoji minimalna veličina postavljena prema zadanim postavkama.	To znači da ako je kontrola postavljena da se skuplja pod AutoSize i nema sadržaja, vrijednost njenog svojstva Veličina je (0x,0y). U tom će se slučaju vaša kontrola smanjiti do određene točke i neće biti odmah vidljiva.
Ako kontrola ne implementira metodu GetPreferredSize , metoda GetPreferredSize vraća posljednju vrijednost dodijeljenu svojstvu Size .	Znači da postavljanje AutoSize na trueneće imati učinka.
Kontrola u ćeliji TableLayoutPanel uvijek se smanjuje kako bi stala u ćeliju dok se ne dosegne njezina minimalna veličina .	Ova se veličina primjenjuje kao maksimalna veličina. To nije slučaj kada je ćelija dio retka ili stupca AutoSize .

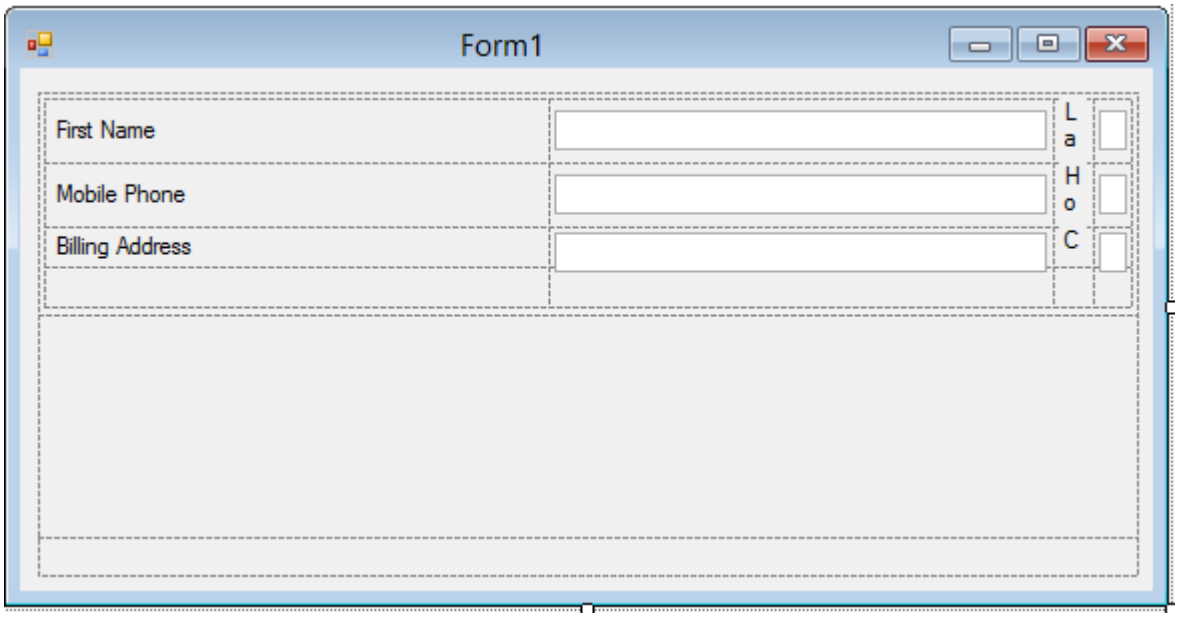
Prilikom kreiranja i dizajniranja forme u kojima se prikazuje puno podataka koriste se tehnike i kontrole poput TableLayoutPanel i svojstva sidrenja kako bi osigurali pregledan, organiziran i jasan način prikaza podataka.

„TableLayoutPanel predstavlja ploču koja dinamički postavlja svoj sadržaj u rešetku sastavljenu od redaka i stupaca.“ (TableLayoutPanel Class (System.Windows.Forms) | Microsoft Learn, 2024.)

Prilikom korištenja takve tehnike, potrebno se pridržavati pravila bez obzira na veličinu same forme. Neka od takvih pravila su:

- Veličina tekstnih okvira mora ostati jednaka za sve tekstne okvire i moraju ispuniti najveći mogući prostor
- Sve oznake moraju biti lijevo poravnate
- Mrežni prikaz mora ispuniti sav preostali prostor
- Mali gumb za osvježavanje mora postojati ispod rešetke, uvijek mora biti poravnat s desne strane obrasca. (*Designing the Layout of Windows Forms Using a TableLayoutPanel, with Auto-Expand Panels - CodeProject*, 2024.)

Na slici ispod nalazi se primjer grafičkog sučelja koje je dizajniran pomoću TableLayoutPanel tehnike.



Slika 16. Dizajniranje forme koristeći TableLayoutPanel (TableLayoutPanel, 2014)

Kod korištenja sidrenja, definiraju se rubovi na kojima se nalaze vizualni elementi koji imaju prepisanu određenu kontrolu. Kada se rubovi definiraju, njihova udaljenost između ruba i vizualnog elementa, odnosno kontrole, mora biti konstantan kroz cijelu takvu formu.

Što se tiče samog prikaza podataka, potrebno je odabrati odgovarajući vizualni element, odnosno kontrolu. Najčešće korištene takve kontrole su:

- Mrežni prikaz podataka (eng. Data Grid View)
- Prikaz stabla (eng. Tree View)
- Prikaz liste (eng. List View)

Mrežni prikaz podataka pogodan je za prikaz tabličnih podataka odnosno prikaz tablica. Prikaz stabla koristi se za hijerarhijske podatke te prikaz podataka u obliku stabla, bez obzira na

njihovu složenost. Prikaz liste prikazuje podatke u obliku popisa, odnosno liste, koje omogućuje korisnicima odabir jedne ili više stavki sa liste.

Kako bi se korisniku pružilo jasan pregled mogućnosti i podataka, potrebno je razdvojiti i napraviti obrasce, odnosno forme, po jasnoj, specifičnoj svrsi. U nastavku slijedi tablica sa smjernicama koje će olakšati rad sa Windows Forms tehnologijom.

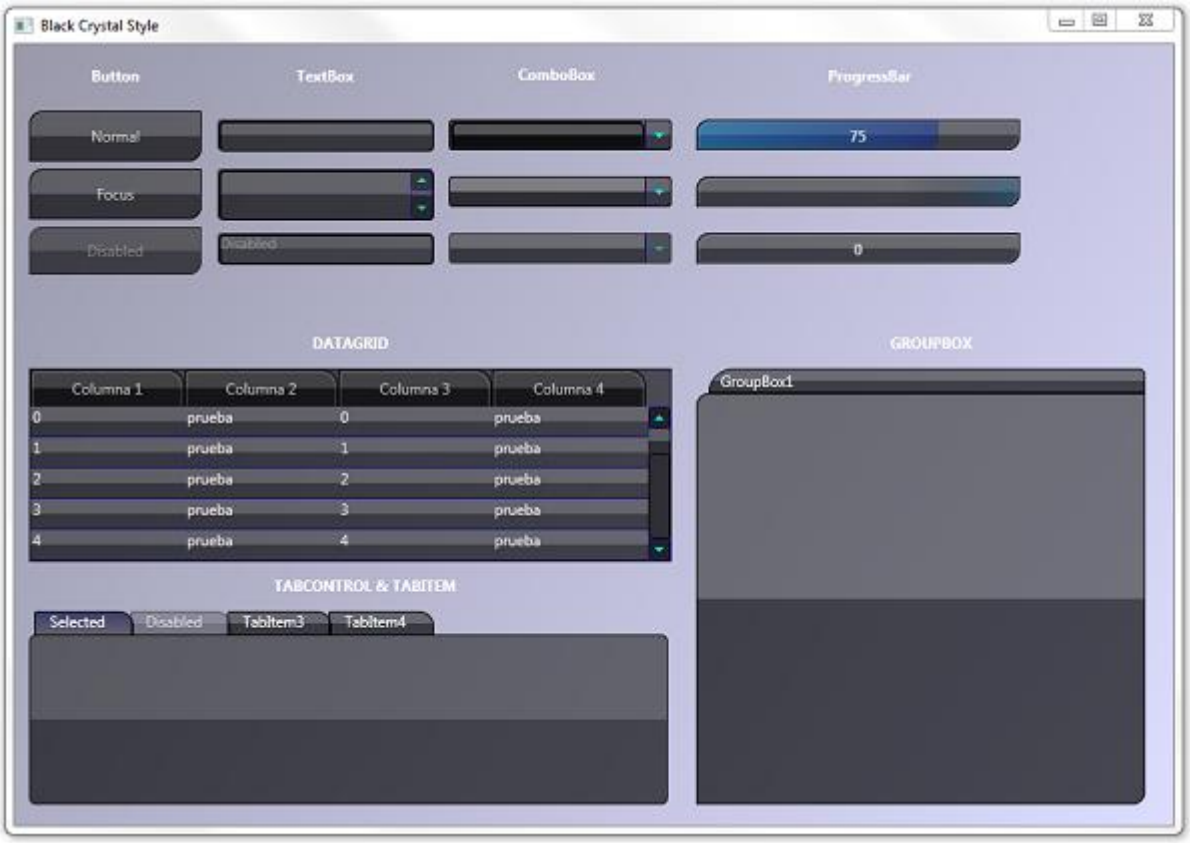
Tablica 3. Smjernice za korištenje Windows Forms tehnologije (autorski rad)

Kategorija	Element	Smjernice
Kodiranje	Form.Designer.cs	Ne mijenjati automatski generirani kod.
Kodiranje	Form.cs	Pisanje koda za interakciju s vizualnim elementima sučelja.
Dizajn	AutoSize, Margin, Padding, Size	AutoSize automatski prilagođava veličine elemenata. Margin određuje prostor oko elemenata. Padding određuje prostor unutar elemenata. Size definira širinu i visinu elemenata.
Struktura	Načelo jedinstvene odgovornosti(eng. „Single Responsibility Principle“	Razdvajanje koda za uređivanje grafičkih sučelja i koda za pristup bazama podataka.
Vizualni elementi	TableLayoutPanel, DataGridView, Tree View, List View	TableLayoutPanel za dinamičko pozicioniranje sadržaja i elemanata unutar rešetki. DataGridView za prikaz podataka u tablici. Tree View za prikaz podataka u hijerarhiji. List

		View za prikaz podataka u obliku liste.
Vizualni dizajn	Anchor	Anchor definira rubove koje održava rubove elemenata fiksnim.

3.2. Smjernice za rad s Windows Foundation Presentation tehnologijom

Bitna značajka koja pomaže kod samog kreiranja grafičkog sučelja korištenjem WPF tehnologije jest korištenje stilova i predložaka. Njihovim korištenjem može se lakše postići dosljednost jednakog stila i predložaka kroz cijelu aplikaciju, odnosno kroz sva grafička sučelja. Iako već postoji unaprijed definirani stilovi, WPF omogućuje programerima da pomoću klase UserControl sami kreiraju i izgrade vizualne elemente prilagođavajući svojstva i komponente po vlastitim preferencijama. Najčešći način izrade stilova jest pomoću svojstva Resursi (eng. „Resources“) unutar XAML datoteke. Također se, upotrebom XAML-a, omogućuje stvaranje dinamičkih korisničkih sučelja. U nastavku slijedi primjer grafičkog sučelja koje koristi vlastiti stil. Možemo primijetiti da su uređeni gumbi, tekstni okviri, traka za pomicanje, traka za napredak, čime je postignut dosljedan i atraktivan dizajn kroz cijelo grafičko sučelje.



Slika 17. Dizajnirani stil u WPF tehnologiji (Sučelje u WPF tehnologiji,2010)

Prilikom izrade vizuala za prikaz podataka u raznim oblicima (slike, tablice) poželjno je koristiti predložak podataka (eng. „DataTemplate“). Predložak podataka je klasa kako bi se vizualizirali podaci odnosno podatkovni objekti. Sve što se nalazi unutar predloška podataka dobiva definiran izgled.

Novost u WPF tehnologiji su zasigurno i okidači (eng. „Triggers). Postoji nekoliko okidača, a to su:

- Okidači svojstava (eng. „Property triggers“)
- Okidači događaja (eng. „Event triggers“)
- Više okidača (eng. „Multi triggers“)
- Okidači podataka (eng. „Data triggers“)
- Više okidača podataka (eng. „Multi data triggers“)

Okidače je veoma korisno koristiti kada se želi promijeniti određeno svojstvo ili pokrenuti neka radnja prilikom nekog događaja. Na primjer, na događaj klika gumba može se pokrenuti animacija koja zasigurno obogaćuje sami izgled, ali i korisničko iskustvo korištenja grafičkog sučelja.

Inovacija koju donosi WPF tehnologija jest animacije i tranzicije. Za kreiranje animacija koristi se scenarijska ploča (eng. „Storyboard“).

Tablica 4. Smjernice za korištenje WPF tehnologije (autorski rad)

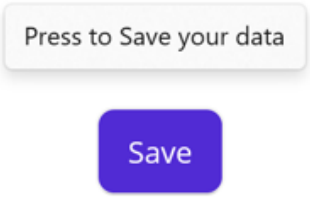
Kategorija	Element	Smjernice
Stilizacija	Styles, UserControl	Styles omogućuje dosljednost dizajna kroz cijelu aplikaciju pomoću definiranih stilova u XAML datotekama. UserControl omogućuje kreiranje prilagođenih Rvizualnih elemenata koji se mogu ponovno koristiti.
Resursi	Resources	Ključno za definiranje stilova i predložaka koji se mogu ponovno koristiti unutar cijele aplikacije.
Predlošci	DataTemplate	Definira prikaz podataka, omogućuje prilagodbu vizualizacije podataka i objekata.
Interakcija	Triggers, Property Triggers, Event Triggers, Multi Triggers, Dana Triggers	Triggersi omogućuju promjena svojstava ili pokretanja akcija na temelju događaja. Property Triggers mijenja izgled ili ponašanje elemenata na temelju događaja. Event Triggers omogućuje pokretanje animacija ili metoda na temelju događaja. Multi i Data Triggers omogućuju

		složenije uvjete za aktiviranje promjena baziranih na kombinaciji uvjeta ili podataka.
Animacije	Storyboard	Koristi se za definiranje i kontrolu animacija.

3.3. Smjernice za rad s .NET MAUI tehnologijom

Iako .NET MAUI nudi velike mogućnosti što se tiče kreiranja i prilagođavanja samih stilova i izgleda, preporuka je da se pridržava standardnih izgleda koje omogućuje .NET MAUI. S obzirom da je .NET MAUI tehnologija koja podržava izradu grafičkih sučelja za više platforma, treba biti oprezan prilikom kreiranja istog. Kao što je to primjer u WPF tehnologiji, .NET MAUI također koristi XAML za vizualni sadržaj grafičkog sučelja.

Dolaskom .NET 7 okvira implementirane su nove značajke, kao što je ToolTip koji može biti od velike koristi prilikom kreiranja grafičkog sučelja. ToolTip pruža korisniku dodatne informacije koja upućuje korisnika na kontrole u grafičkom sučelju. Kako bi takve informacije bile vidljive, korisnik mora mišem prelaziti preko kontrole.



Slika 18. Korištenje ToolTip-a u .NET MAUI (ToolTip u .NET MAUI, 2023)

Prilikom korištenja predložaka podataka poželjno je koristiti mrežu (eng. „grid“) s eksplicitnim veličinama umjesto automatskih veličina. Razlog je jednostavan, a to je da se korištenjem automatskih veličina neće dobiti željeni izgled i prostor kada se prikazuje veća količina podataka. Umjesto toga potrebno je koristiti eksplicitno dimenzioniranje.

Prilikom kreiranja navigacije .NET MAUI tehnologija naglašava navigaciju usmjerenu i intuitivnu korisniku. Nudi mogućnosti korištenja Shell i Tabbed/Flyout/Nav stranice. Poželjno je da se ta dva stila ne kombiniraju zbog toga što zapravo nisu kreirani i dizajnirani da rade zajedno.

Kod kreiranje aplikacije sa više grafičkih sučelja potrebno je iskoristiti Glavnu Stranicu (eng. „MainPage“) svojstvo samo jednom kako ne bi došlo do problema sa performansama i izvođenjem cjelokupne aplikacije, kao što su problemi s navigacijom ili gubitkom trenutnog stanja aplikacije prilikom promjene glavne stranice.

Kod kreiranja animacija dobra je praksa koristiti Lottie integraciju koja nudi brzu i laganu implementaciju animacija u grafičko sučelje. Lottie se smatra alternativa GIF-ovima te je format datoteke otvorenog koda za animacije.

Za dodavanje obruba oko različitih elemenata preporučljivo je koristiti granice (eng. „Border“) umjesto okvira (eng. „Frame“). Razlog korištenja granica je jednostavan, a to je da .NET MAUI tehnologija omogućuje veću fleksibilnost i kontrolu nad elementima njihovim korištenjem. Na primjer, korištenjem granica elementu je moguće dodati samo gornje rubove, bez dodavanja donjih rubova. Osim toga, moguće je dodati više složenijih oblika kao što su elipse ili poligoni, umjesto standardnih pravokutnika.

Slika 19. Korištenje svojstva Border u .NET MAUI (Border u .NET MAUI, 2023)

Kako bi se što bolje poboljšao raspored u grafičkom sučelju, poželjno je koristiti horizontalni raspored (eng. „HorizontalStackLayout“) i vertikalni raspored (eng. „VerticalStackLayout“) umjesto StackLayout svojstava. Prednost korištenja horizontalnog i vertikalnog rasporeda je u jasnijoj orijentaciji, s obzirom da je njihovim korištenjem unaprijed jasno kako će se elementi rasporediti. Osim toga, smanjuje se mogućnost pogrešaka i nepravilnog rasporeda koje se mogu pojaviti zbog dodavanja novih elemenata.

Tablica 5. Smjernice za korištenje .NET MAUI tehnologije (autorski rad)

Kategorija	Element	Smjernice
Općenito	Standardni izgledi	Preporuka je korištenje standardnih izgleda koje .NET MAUI nudi.
Komponente	ToolTip	Pomaže korisnicima pružanjem dodatnih

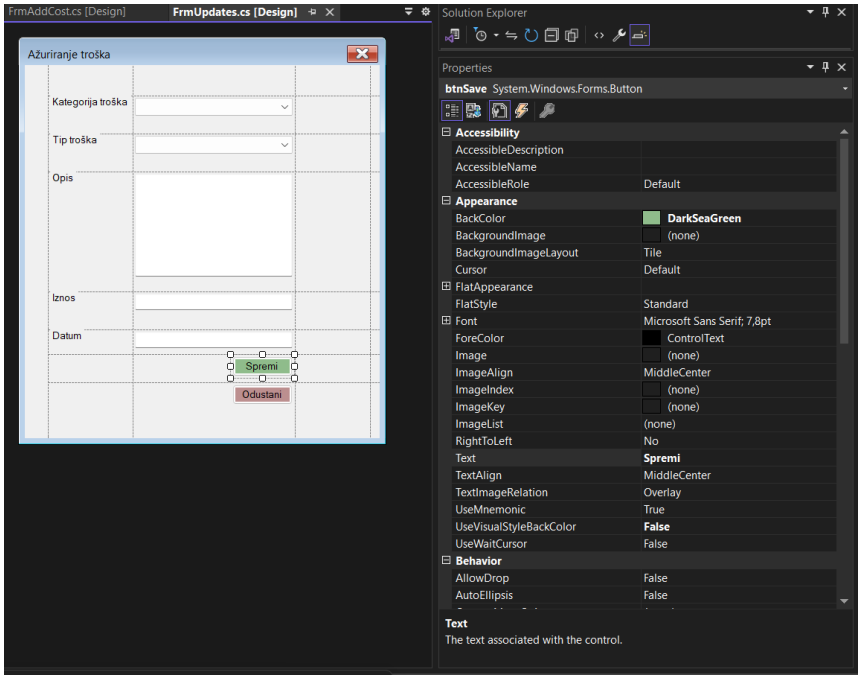
		informacija o elementima prilikom prelaska mišem preko njih.
Layout	Grid	Preporučuje se koristiti eksplicitne veličine umjesto automatskih kako bi se osigurao željeni izgled.
Layout organizacija	HorizontalStackLayout, VerticalStackLayout	Koristiti specifične layoute za jasniju orijentaciju elementa, smanjuje mogućnost grešaka prilikom dodavanja novih elemenata.
Navigacija	Shell, Tabbed/Flyout/Nav	Koristiti jedan stil navigacije po aplikaciji kako bi se izbjegli konflikti u navigaciji.
Animacije	Lottie	Koristiti Lottie za brze i kvalitetne animacije.
Stilizacija	Border	Korištenjem bordera postiže se veća fleksibilnost. Pruža dodavanje složenijih oblika.

4. Implementacija .NET tehnologija

Nakon teorijskog dijela o Windows Forms, Windows Presentation Foundation i .NET MAUI tehnologijama i njihovim smjernicama o korištenju, slijedi njihova implementacija u izradi grafičkih sučelja za desktop aplikaciju pod imenom Financijski Voditelj. Svrha same aplikacije jest praćenje i bilježenje vlastitih financija odnosno troškova. Sastoji se od 3 grafička sučelja. Prvo sučelje koje prikazuje sve troškove, drugo sučelje koje omogućuje unos novih troškova i treće sučelje koje omogućuje ažuriranje postojećih troškova. Aplikacija osim što nudi pregled troškova, unos i njihovo ažuriranje također nudi i njihovo brisanje. Aplikacija nudi filtriranje troškova prema njihovim kategorijama te nudi pretraživanje prema tipu troška. Kod izrade samih sučelja za sve tri tehnologije držao sam se općenitih pravila i smjernica o izradi sučelja kao što je preglednost svih elemenata, intuitivnost prilikom korištenja mogućnosti, struktura i pozicija elemenata.

4.1.1. Implementacija aplikacije pomoću Windows Forms Tehnologije

Na slici ispod prikazan je Designer za uređivanje grafičkog sučelja u Windows Forms tehnologiji. Također su prikazana i svojstva gumba „Spremi“. Kod dizajna ovog sučelja dosljedno prema smjernicama koristio sam TableLayoutPanel za raspored vizualnih elemenata.

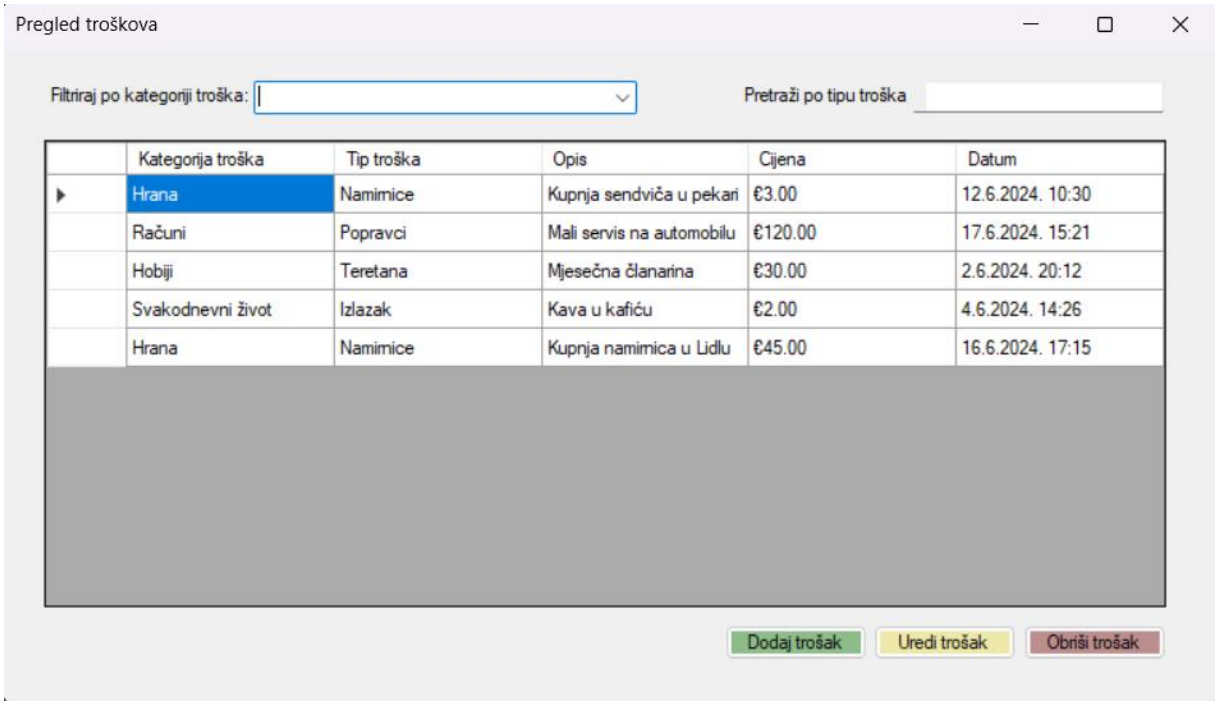


Slika 20. Designer okruženje u Windows Forms tehnologiji (autorski rad)

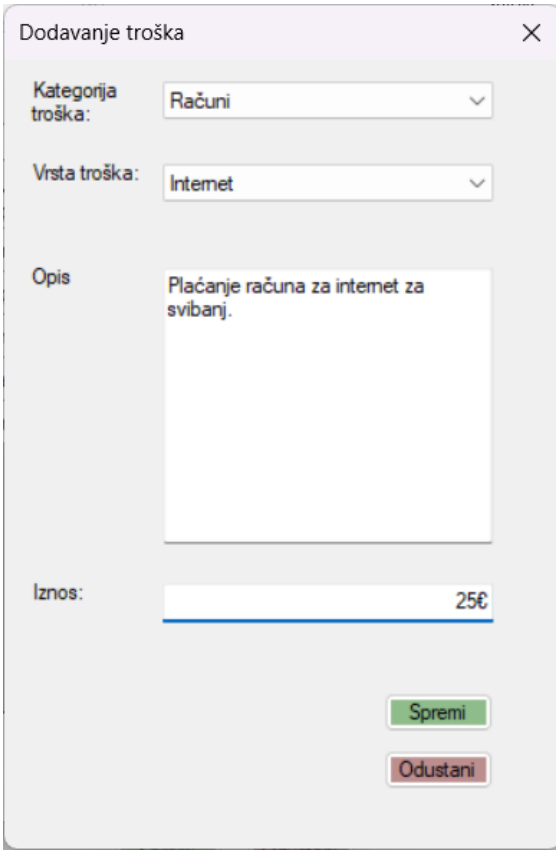
Na slici ispod prikazan je isječak koda za gumb koji dodaje trošak. Prilikom pritiska na gumb „btnAddSaveCost“ stvaraju se 4 string varijable koje spremaju odabrane kategorije troška i tipove troška iz padajućih izbornika. Također se spremaju opis i cijena troška iz tekstualnih okvira. Nakon spremanja novog troška korisniku se prikazuje obavijest odnosno poruka o uspješno unesenom trošku.

```
private void btnAddSaveCost_Click(object sender, EventArgs e)
{
    string categoryCost = (string)cboAddCategoryCost.SelectedItem;
    string typeCost = (string)cboAddTypeCost.SelectedItem;
    string description = txtAddDescriptionCost.Text;
    var price = int.Parse(txtAddPriceCost.Text);

    NewCost = new Cost
    {
        CategoryCost = categoryCost,
        TypeCost = typeCost,
        Description = description,
        Price = price,
        Date = DateTime.Now // You can set this to a different date if required
    };

    AddedCost = true;
    DialogResult = DialogResult.OK;
    MessageBox.Show("Uspješno ste unijeli trošak!", "Obavijest", MessageBoxButtons.OK, MessageBoxIcon.Information);
    Close();
}

1 reference
private void btnAddCancelCost_Click(object sender, EventArgs e)
{
    Close();
}
```

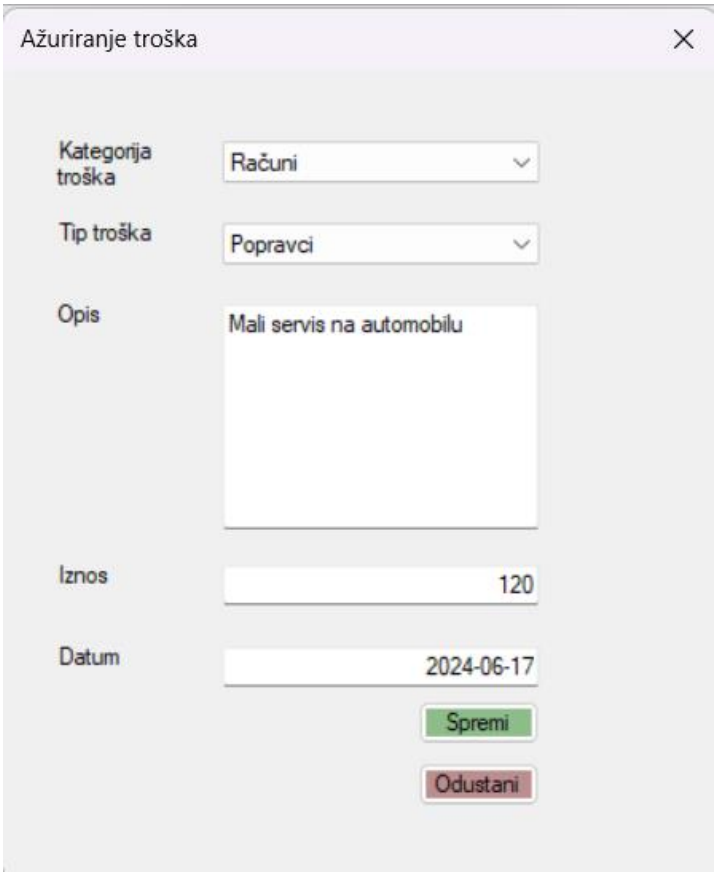
Slika 21. Isječak koda u Windows Forms tehnologiji (autorski rad)



Slika 22. Pregled troškova u Windows Forms tehnologiji (autorski rad)



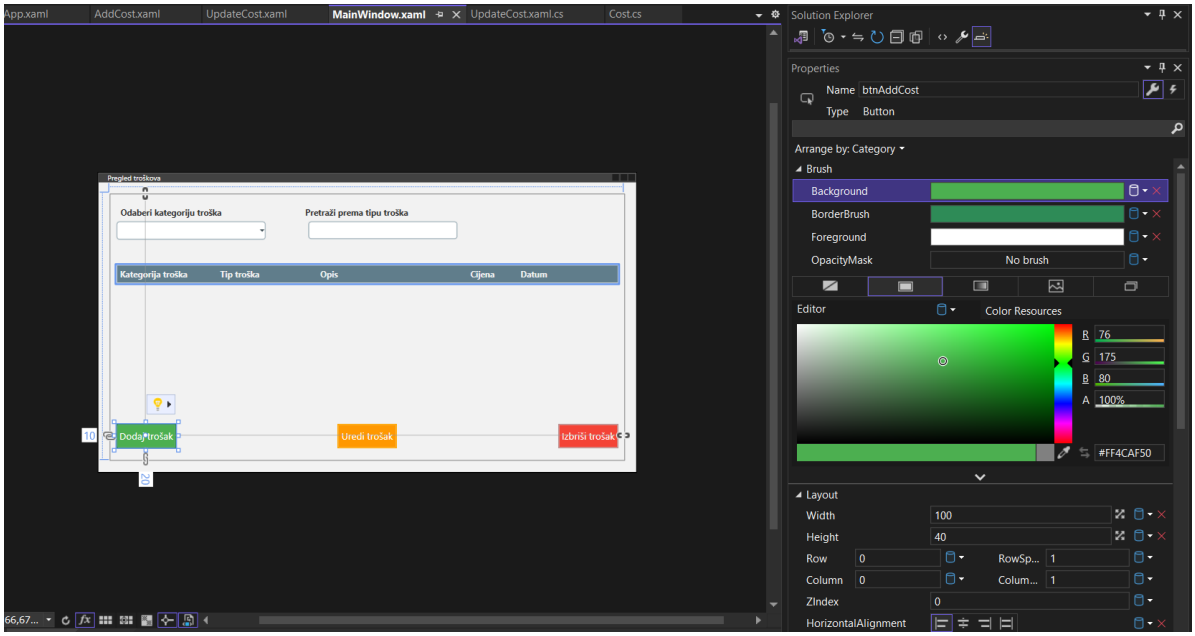
Slika 23. Dodavanje troška u Windows Forms tehnologiji (autorski rad)



Slika 24. Ažuriranje troška u Windows forms tehnologiji (autorski rad)

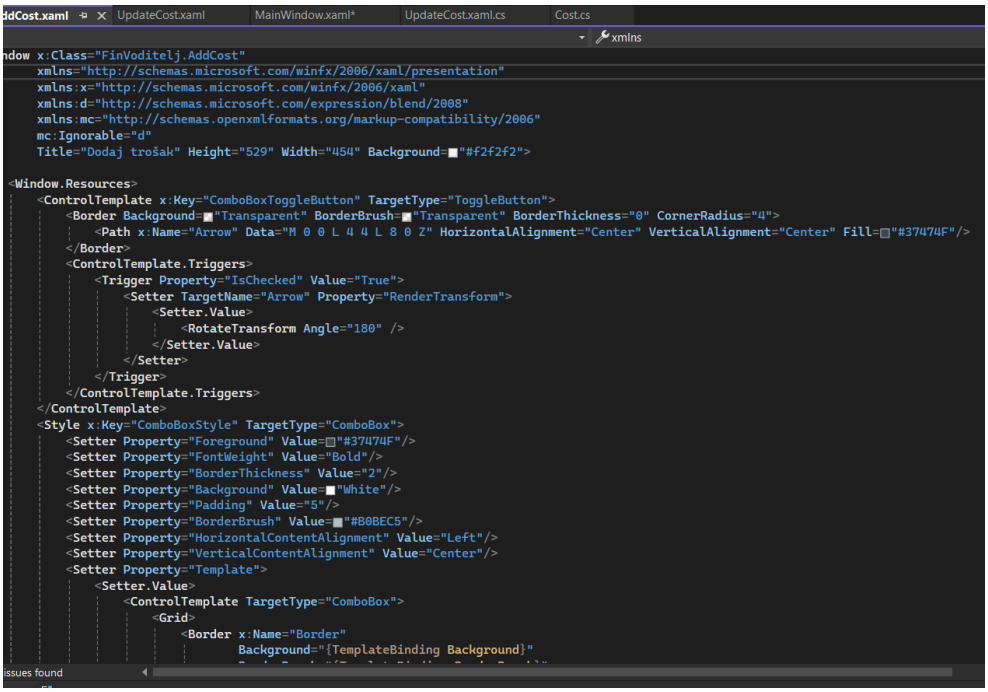
4.1.2. Implementacija aplikacije pomoću Windows Foundation Presentation tehnologije

Kod kreiranje sučelja u WPF tehnologiji uglavnom sam se oslanjao na XAML kako bi uredio grafičko sučelje. Osim toga koristio sam i Designer kako bih prilagodio elemente i njihove pozicije te promijenio određena svojstva (boja, naziv).

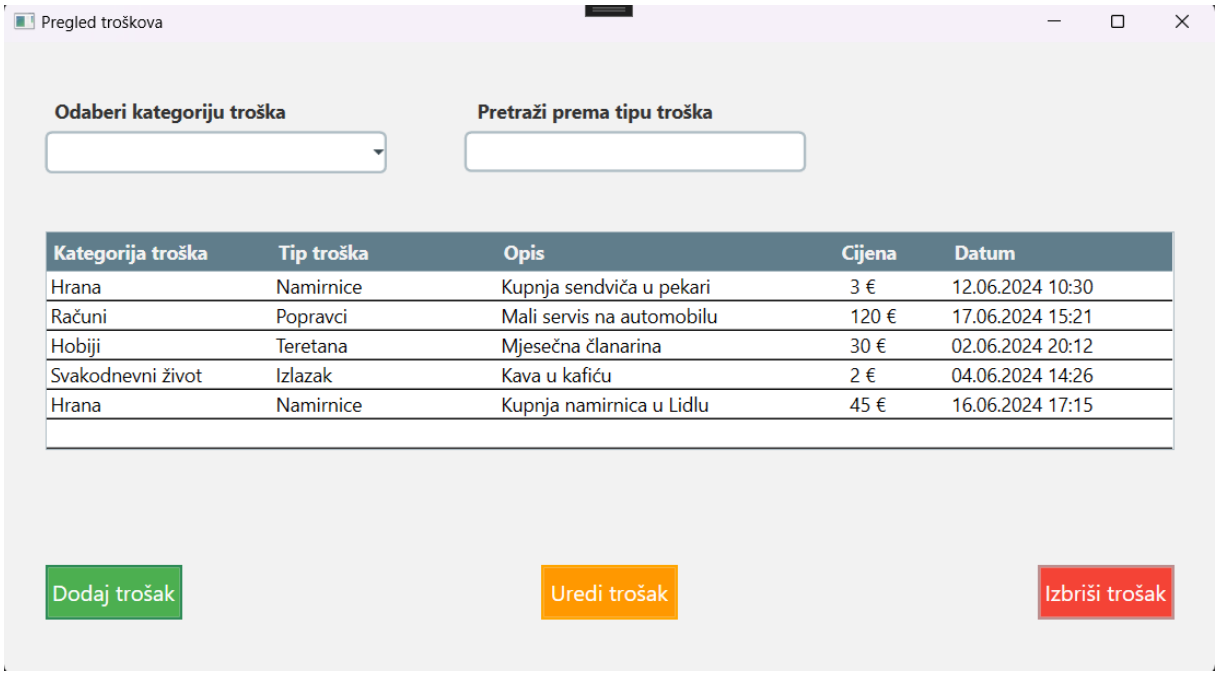


Slika 25. Designer okruženje u WPF tehnologiji (autorski rad)

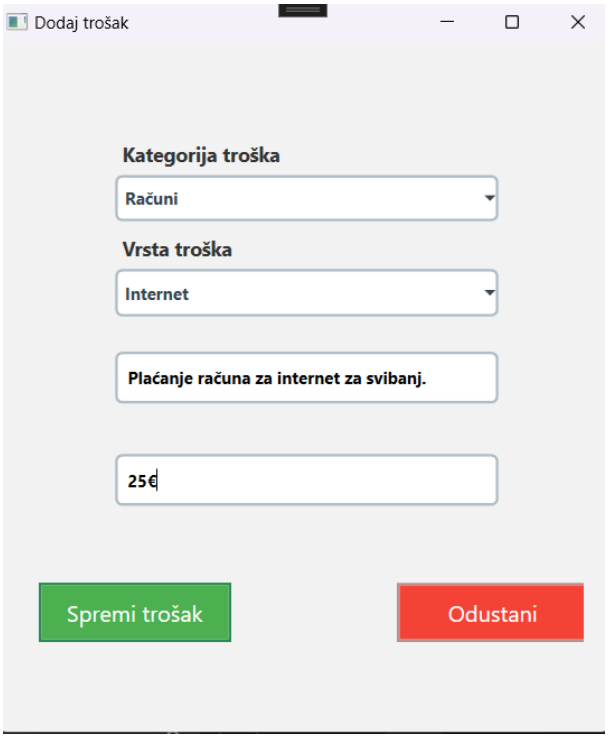
Ovaj isječak koda prikazuje definicija stilova i predložaka kontrola (eng. „ControlTemplate“) za prilagođavanje izgleda elemenata sučelja.



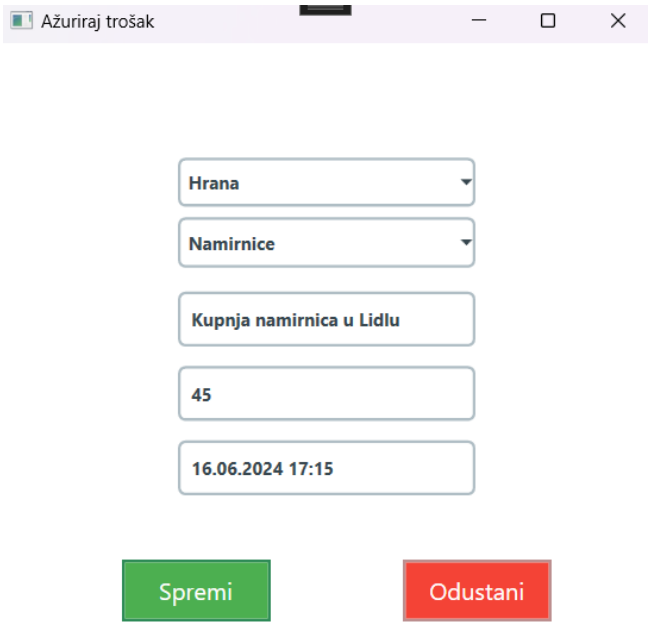
Slika 26. Isječak XAML koda u WPF tehnologiji (autorski rad)



Slika 27. Pregled troškova u WPF tehnologiji (autorski rad)



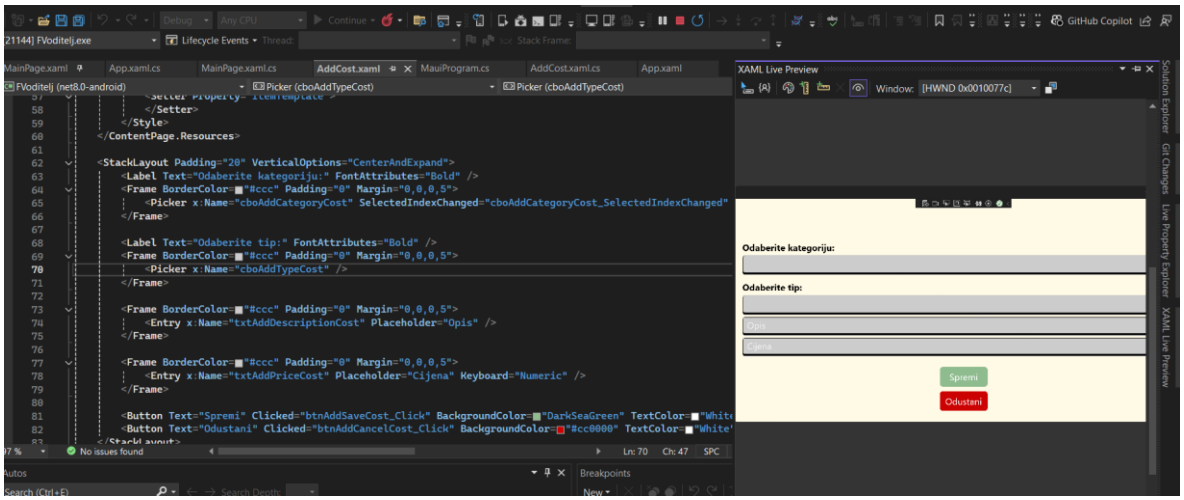
Slika 28. Dodavanje troška u WPF tehnologiji (autorski rad)



Slika 29. Ažuriranje o troška u WPF tehnologiji (autorski rad)

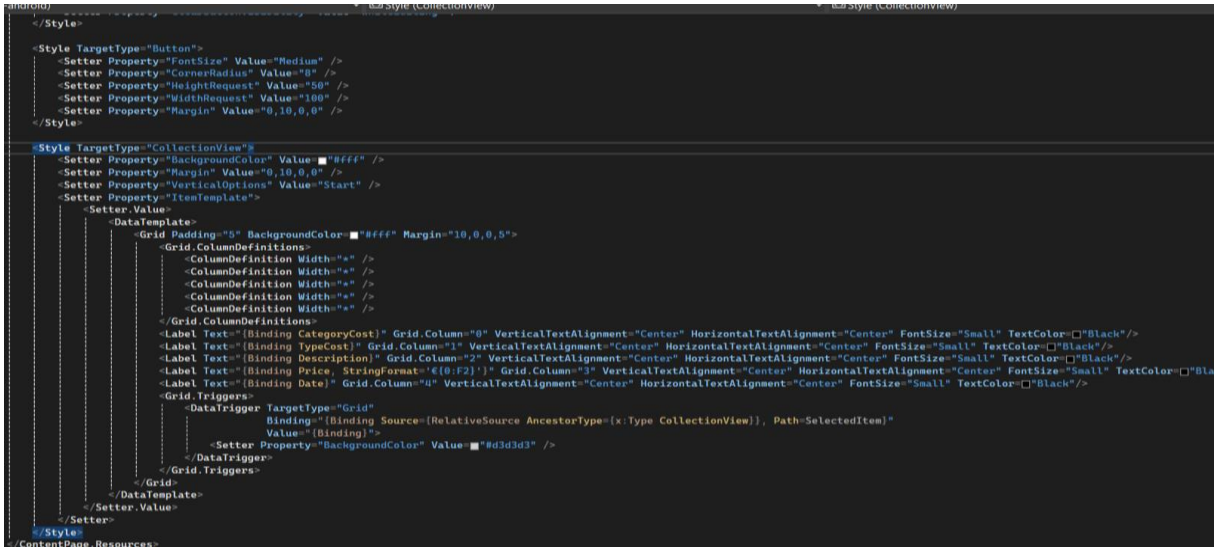
4.1.3. Implementacija aplikacije u .NET MAUI tehnologiji

Kod implementacije aplikacije u .NET MAUI tehnologiji koristio sam XAML te „hot reload“ koje omogućuje da se promjene u kodu vide uživo u stvarno vrijeme. Kao što je to bio slučaj u WPF tehnologiji i ovdje sam se oslanjao na sam XAML te sam „hot reload“ koristio kako bih ispravio pozicioniranje i raspored elemenata.



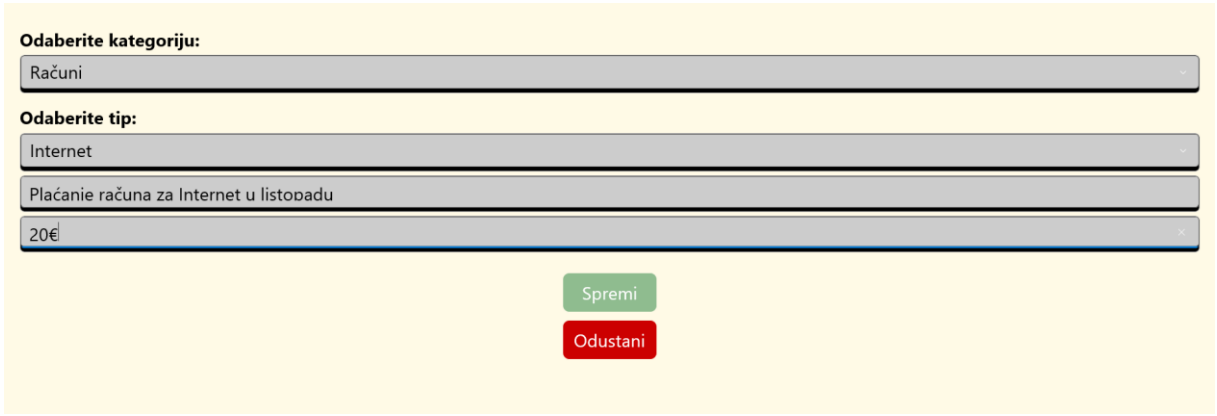
Slika 30. XAML Live Preview u .NET MAUI tehnologiji (autorski rad)

Ovaj isječak XAML koda definira stil za `CollectionView` u .NET MAUI aplikaciji. Stil postavlja pozadinu, margine i vertikalne opcije za `CollectionView`, te definira `ItemTemplate` koji prikazuje podatke svakog elementa u `CollectionView`-u u obliku `Grid`-a s pet stupaca. Također uključuje `DataTrigger` koji mijenja pozadinu `Grid`-a kada se odabere određeni element u `CollectionView`-u.

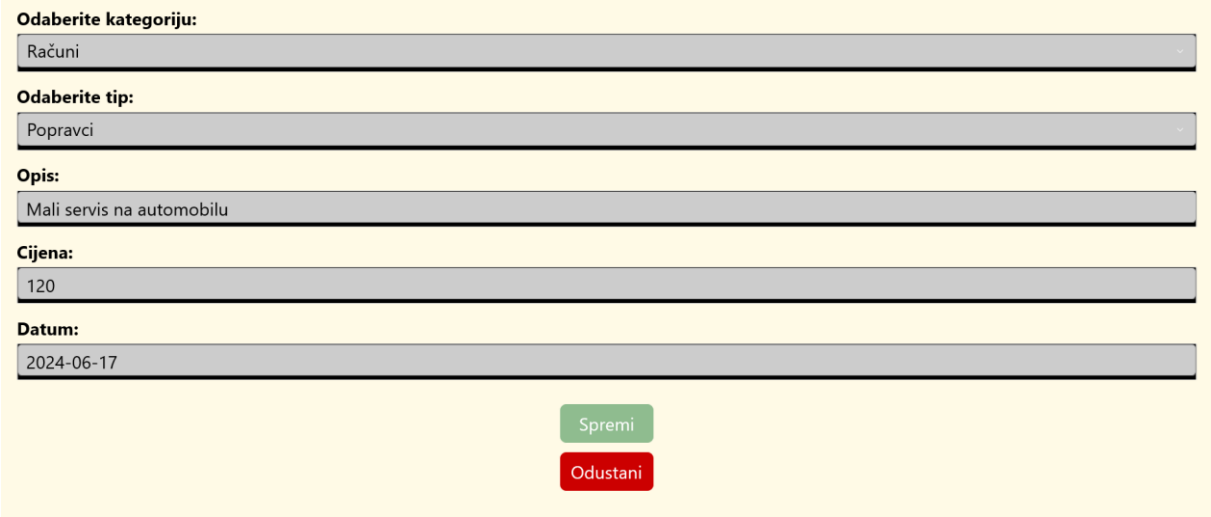


Slika 31. Isječak XAML koda u .NET MAUI tehnologiji(autorski rad)

Slika 32. Pregled troškova u .NET MAUI tehnologiji (autorski rad)

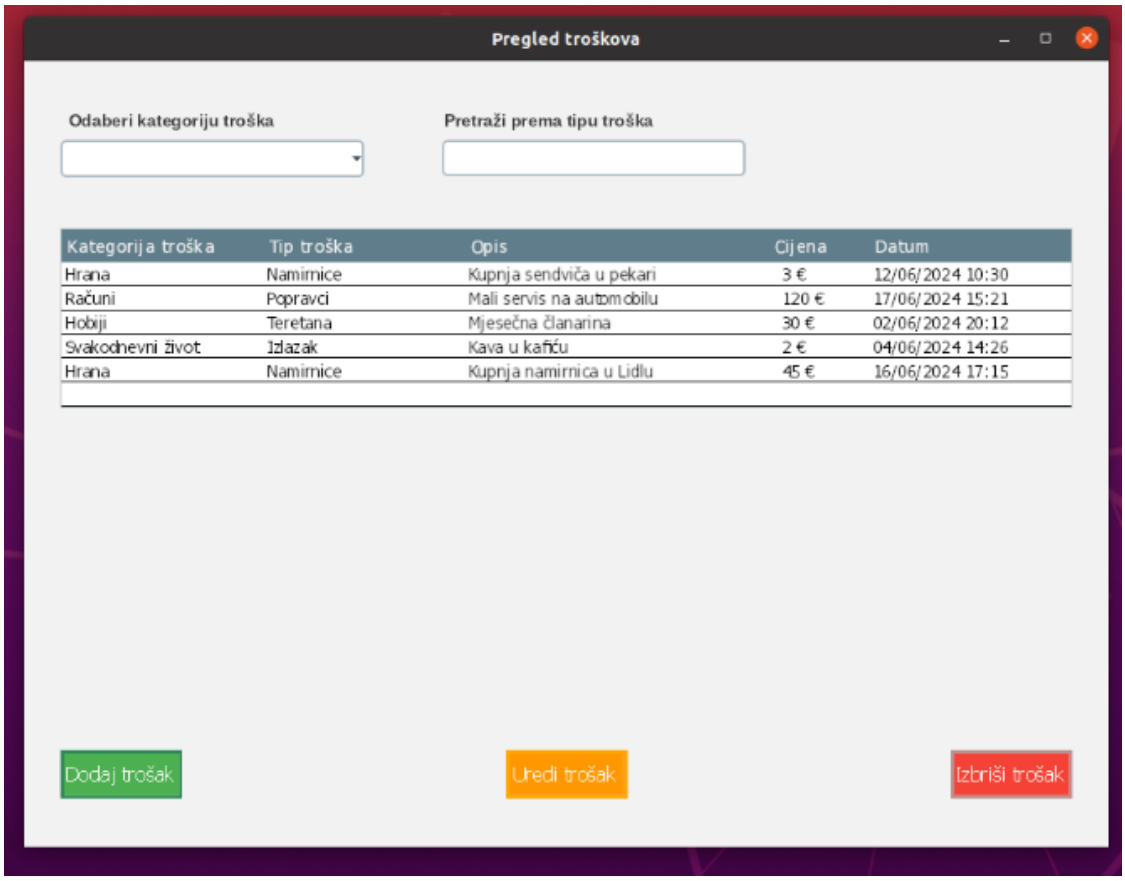


Slika 33. Dodavanje troška u .NET MAUI tehnologiji (autorski rad)

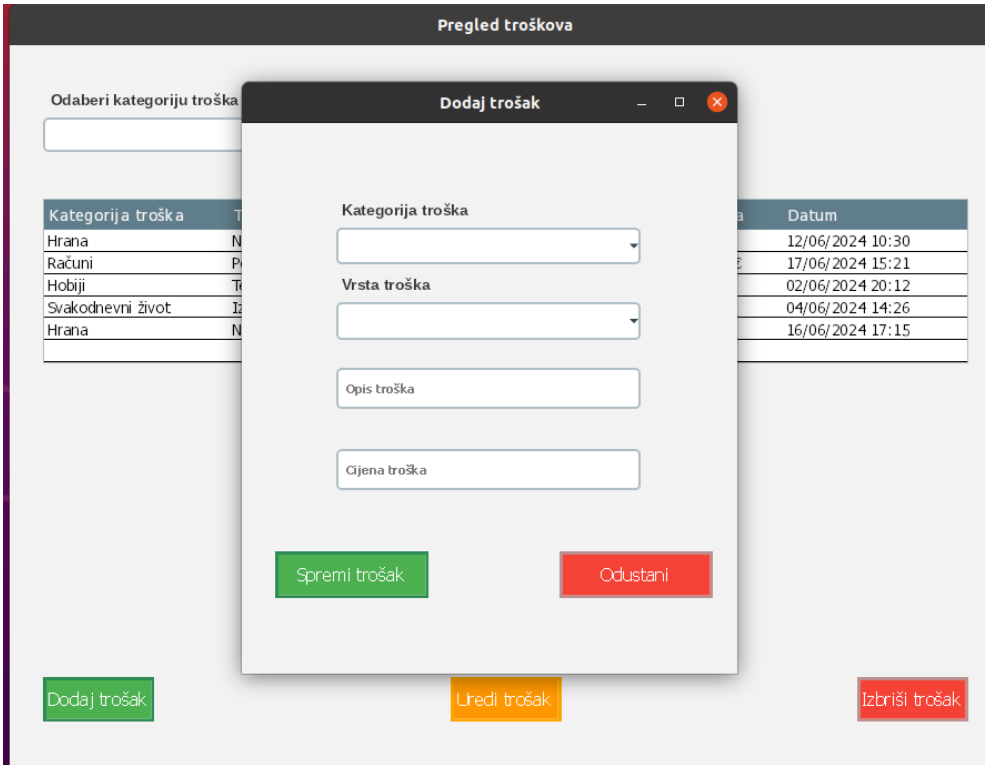


Slika 34. Ažuriranje troška u .NET MAUI tehnologiji (autorski rad)

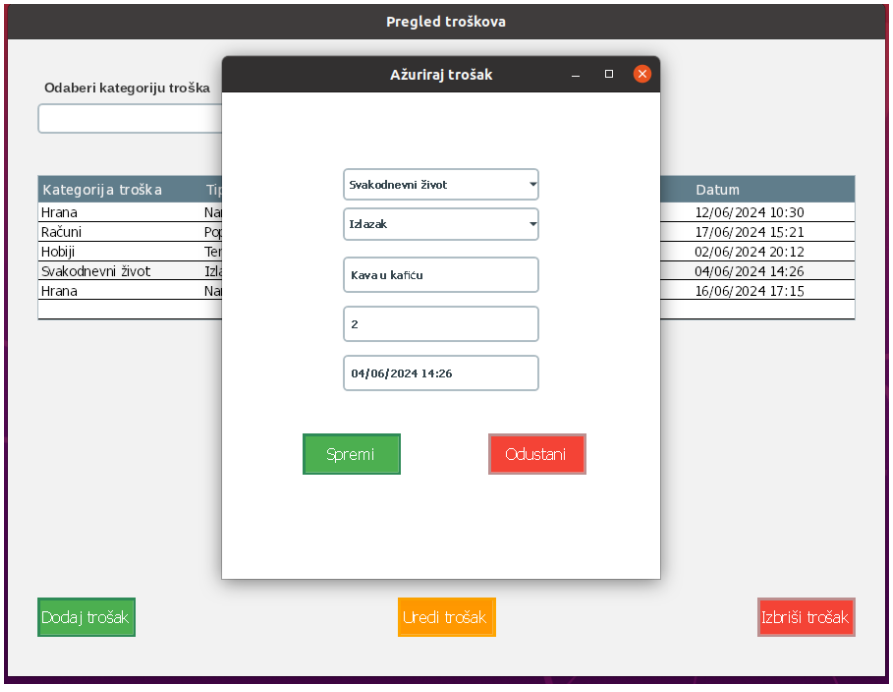
Kako bi vidjeli višeplatformnost koju omogućuje .NET MAUI pokrenuo sam ovu aplikaciju na operacijskom sustavu Ubuntu. U nastavku slijedi izgled MAUI aplikacije.



Slika 35. Pregled troškova u .NET MAUI tehnologiji na Ubuntu (autorski rad)



Slika 36. Dodavanje troškova u .NET MAUI tehnologiji na Ubuntu (autorski rad)



Slika 37. Ažuriranje troškova u .NET MAUI tehnologiji na Ubuntu (autorski rad)

5. **Zaključak**

Korištenje .NET tehnologija uveliko olakšava posao prilikom izrade grafičkih sučelja i desktop, ali i mobilnih aplikacija. S obzirom na međusobnu povezanost i integraciju može se reći da je .NET tehnologija postala standard prilikom kreiranja aplikacija. U ovom radu istražene su Windows Forms, Windows Presentation Foundation i .NET MAUI tehnologije. Windows Forms tehnologija usprkos svojoj starosti i dalje pruža široke mogućnosti za kreiranje aplikacija te može zadovoljiti sve potrebe korisnika. Ipak, s obzirom na naprednije i moćnije tehnologije jasno je kako se Windows Forms sve manje koristi. Windows Presentation Foundation ide korak dalje u odnosu na Windows Forms te nudi XAML koji omogućuje razvoj i izradu moćnijih i bolje prilagođenih sučelja i aplikacija. Iako se u početku teže snalaziti kod same izrade, nakon nekog vremena je veoma intuitivno i lako upravljati ovom tehnologijom. .NET MAUI tehnologija osim što nudi XAML za potpunu prilagodbu sučelja također nudi i razvoj za višeplatformske aplikacije bez potrebe za ispravljanjem samog koda ili izgleda sučelja. Prilikom izrade sučelja u ovim tehnologijama najlakše mi se bilo snalaziti u Windows Forms tehnologiji zbog samog iskustva u njoj, ali i samoj jednostavnosti. Windows Forms ne pruža toliko mogućnosti kao ostale dvije tehnologije pa nije ni bilo problema prilikom same izrade sučelja. Kod WPF i .NET MAUI tehnologija situacija je drugačija. Kod izrade sučelja pomoću WPF tehnologije trebalo mi je više vremena kako bi uspio izraditi izgled sučelja preko XAML jezika. Pomoć kod izrade u WPF tehnologiji jest i dizajner koji omogućuje vizualno uređivanje sučelja. Kod samog početka izrade sučelja više sam koristio dizajner te sam vizualno oblikovao sučelje, a zatim sam pomoću koda precizirao i dovršavao uređivanje(dodavanje određenih boja, margine, razmaci između elemenata). S obzirom da nudi više mogućnosti potrebno je više vremena za istražiti i odabrati odgovarajuće elemente za sučelje. Kod izrade sučelja uz pomoć .NET MAUI tehnologije smatram da sam se lakše i bolje snašao nego kod izrade sučelja u WPF tehnologiji zbog toga što .NET MAUI također koristi XAML za izradu sučelja. .NET MAUI tehnologija nudi brojne mogućnosti za svaku od platformi. Kod izrade sučelja u .NET MAUI tehnologiji dobrim dijelom sam se oslanjao na obrube i margine kako bih oblikovao i dizajnirao elemente u sučelju. Smatram kako se i danas svaka od ovih tehnologija itekako mogu dobro iskoristiti, ali je potrebno odabrati onu koja najbolje zadovoljava potrebe korisnika. Na primjer, ukoliko se aplikacije rade za različite timove unutar samog poduzeća (za dohvaćanje podataka iz baze podataka koji su potrebni marketinškom timu) dovoljno je

koristiti Windows Forms zbog svoje jednostavnosti. Ukoliko je namjera napraviti komercijalna sučelja te sučelja koja će koristiti šira publika onda je odabir WPF ili .NET MAUI tehnologije bolje odabir zbog više mogućnosti što se tiče samog dizajna sučelja, ali i izradi sučelja za više platformi (.NET MAUI tehnologija). S obzirom na sve svoje mogućnosti možemo zaključiti kako je .NET tehnologija prošlost, sadašnjost i budućnost prilikom izrade grafičkih sučelja za stolne, ali i mobilne platforme.

Popis literature

Beyond the Basics: Best Practices—UI Handling in .NET MAUI. (2024). Preuzeto Lipanj 7, 2024, iz https://www.telerik.com/blogs/beyond-basics-best-practices-ui-handling-net-maui

C# (CSharp) - Windows Forms Best Practices - LearningKoala. (2021). Preuzeto Lipanj 2, 2024, iz https://www.learningkoala.com/csharp-windows-forms-best-practices.html#mcetoc_1f2p8avhg3cv

Control layout options - Windows Forms .NET | Microsoft Learn. (2021). Preuzeto Svibanj 12, 2024, iz https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/layout?view=netdesktop-8.0

Control layout options - Windows Forms .NET | Microsoft Learn. (2021.). Preuzeto Svibanj 25, 2024, iz https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/layout?view=netdesktop-8.0

Designing the Layout of Windows Forms using a TableLayoutPanel, with auto-expand panels - CodeProject. (2014.). Preuzeto Svibanj 28, 2024, iz https://www.codeproject.com/Tips/842418/Designing-the-Layout-of-Windows-Forms-using-a

Display tooltips - .NET MAUI | Microsoft Learn. (2023.). Preuzeto Lipanj 12, 2024, iz https://learn.microsoft.com/en-us/dotnet/maui/user-interface/tooltips

Graphics and Multimedia - WPF .NET Framework | Microsoft Learn. (2022.). Preuzeto Svibanj 12, 2024, iz https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/?view=netframeworkdesktop-4.8

.NET Framework Class Library [Slika] (2021.) Preuzeto Travanj 12, 2024 iz https://www.geeksforgeeks.org/net-framework-class-library-fcl/

Guidelines for Designing Stylable Controls - WPF .NET Framework | Microsoft Learn. (2022). Preuzeto Svibanj 12, 2024, iz https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls/guidelines-for-designing-stylable-controls?view=netframeworkdesktop-4.8

Introducing .NET Multi-platform App UI - .NET Blog. (n.d.). Preuzeto Lipanj 12, 2024, iz https://devblogs.microsoft.com/dotnet/introducing-net-multi-platform-app-ui/

Introducing DirectX to WPF - CodeProject. (2020.). Preuzeto Svibanj 5, 2024, iz https://www.codeproject.com/Articles/200908/Introducing-DirectX-to-WPF

Is Xamarin really dead?. MAUI—Multi-Platform App UI rise from… | by Umair Hassan | BITLogix | Medium. (2021.). Preuzeto Svibanj 20, 2024, iz https://medium.com/bitlogix/is-xamarin-really-dead-c6cd79ca4ecf

Overview - Windows Forms .NET Framework | Microsoft Learn. (2023.). Preuzeto Svibanj 25, 2024, iz https://learn.microsoft.com/en-us/dotnet/desktop/winforms/windows-forms-overview?view=netframeworkdesktop-4.8

Sells, Chris., Griffiths, I., & Sells, Chris. (2007), Programming WPF. Preuzeto Lipanj 6, iz https://books.google.hr/books?id=558i6t1dKEAC&printsec=frontcover&hl=hr#v=onepage&q&f=false

Sells, Chris., & Weinhardt, Michael. (2006), Windows Forms 2.0 programming. Preuzeto Svibanj 13, iz https://books.google.hr/books?id=Woa6WJyyvqIC&printsec=frontcover&hl=hr#v=onepage&q&f=false

.NET MAUI in a Nutshell — Everything You Need to Know in 2024. (2024.). Preuzeto Lipanj 4, 2024, iz https://brainhub.eu/library/net-maui-in-nutshell

Binding mode - .NET MAUI | Microsoft Learn (2024.). Preuzeto Lipanj 4, 2024, iz https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/data-binding/binding-mode?view=net-maui-8.0

Social Club: Sample application using WinForms, C#.NET, ADO.NET and MS Access - CodeProject. (2014.). Preuzeto Svibanj 17, 2024, iz https://www.codeproject.com/Articles/607868/Social-Club-Sample-application-using-WinForms-Csha

State of the Windows Forms Designer for .NET Applications - .NET Blog. (2022.). Preuzeto Lipanj 5, 2024, iz https://devblogs.microsoft.com/dotnet/state-of-the-windows-forms-designer-for-net-applications/

What is .NET MAUI? - .NET MAUI | Microsoft Learn. (2024.). Preuzeto Lipanj, 2024, iz https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-8.0

WPF: Customize your Application with Styles and Control Templates (Part 2 of 2) - CodeProject. (2010). Preuzeto Lipanj 2, 2024, iz https://www.codeproject.com/Articles/85896/WPF-Customize-your-Application-with-Styles-and-C-2

Data binding with Windows Forms 2.0: programming smart client data applications with .NET. Noyes, Brian. (2006). Preuzeto Svibanj 7, 2024, iz https://www.wikiwand.com/en/Windows_Forms

maui-samples/8.0/Apps/WeatherTwentyOne at main · dotnet/maui-samples · GitHub. (2024.). Preuzeto Lipanj 17, 2024, iz https://github.com/dotnet/maui-samples/tree/main/8.0/Apps/WeatherTwentyOne

WPF .NET Framework | Microsoft Learn. (2023). Preuzeto Svibanj 12, 2024, iz https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls/styles-templates-overview?view=netframeworkdesktop-4.8

Windows Form arhitektura [Slika] (2014.) Preuzeto Travanj 12, 2024 iz https://www.codeproject.com/Articles/607868/Social-Club-Sample-application-using-WinForms-Csha
Sučelje u Windows Forms [Slika] (2014.) Preuzeto Travanj 12, 2024 iz https://www.codeproject.com/Articles/607868/Social-Club-Sample-application-using-WinForms-Csha

3D element u WPF [Slika] (2011.) Preuzeto Svibanj 05, 2024 iz https://www.codeproject.com/Articles/200908/Introducing-DirectX-to-WPF

WPF - Overview. (n.d.). Preuzeto Svibanj 5, 2024, iz https://www.tutorialspoint.com/wpf/wpf_overview.htm

WPF arhitektura [Slika] (n.d.) Preuzeto Svibanj 05, 2024 iz https://www.tutorialspoint.com/wpf/wpf_overview.htm

C# - the .Net Framework. (2015.). Preuzeto Svibanj 9, 2024, iz https://www.google.hr/books/edition/C_the_Net_Framework/TYVGCgAAQBAJ?hl=hr&gbpv=1&dq=.net+framework&pg=PT13&printsec=frontcover

3-Tier Architecture in C# - javatpoint. (n.d.). Preuzeto Svibanj 12, 2024, iz https://www.javatpoint.com/3-tier-architecture-in-c-sharp

Animacije u WPF [Slika] (2022.) Preuzeto Svibanj 12, 2024 iz https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/?view=netframeworkdesktop-4.8

Arhitektura .NET MAUI [Slika] (2021.) Preuzeto Svibanj 20, 2024 iz https://medium.com/bitlogix/is-xamarin-really-dead-c6cd79ca4ecf

Introducing .NET Multi-platform App [Slika] (2020 Preuzeto Svibanj 20, 2024 iz https://devblogs.microsoft.com/dotnet/introducing-net-multi-platform-app-ui/
WeatherTwentyOne [Slika] (2024.) Preuzeto Svibanj 05, 2024 iz https://github.com/davidortinau/WeatherTwentyOne

C# Windows Forms - Best Practices. (n.d.). Preuzeto Svibanj 22, 2024, iz https://bettersolutions.com/csharp/windows-forms/best-practies.htm

TableLayoutPanel [Slika] (2014.) Preuzeto Svibanj 28,2024 iz https://www.codeproject.com/Tips/842418/Designing-the-Layout-of-Windows-Forms-using-a

Sučelje u WPF tehnologiji [Slika] (2010.) Preuzeto Lipanj 06, 2024 iz https://www.codeproject.com/Articles/85896/WPF-Customize-your-Application-with-Styles-and-C-2

Border - .NET MAUI | Microsoft Learn. (2023.). Preuzeto Lipanj 7, 2024, iz https://learn.microsoft.com/en-us/dotnet/maui/user-interface/controls/border?view=net-maui-8.0

Border u .NET MAUI [Slika] (2023.) Preuzeto Lipanj 10, 2024 iz https://learn.microsoft.com/en-us/dotnet/maui/user-interface/controls/border?view=net-maui-8.0

ToolTip u .NET MAUI [Slika] (2023.) Preuzeto Lipanj 10, 2024 iz https://learn.microsoft.com/en-us/dotnet/maui/user-interface/tooltips?view=net-maui-8.0

What is .NET MAUI?. (2022.). Preuzeto Lipanj 12, 2024, iz https://www.c-sharpcorner.com/article/what-is-net-maui/

Popis slika

Popis tablica