ŠOLSKI CENTER VELENJE
ELEKTRO IN RAČUNALNIŠKA ŠOLA VELENJE
Trg mladosti 3, 3320 Velenje

MLADI RAZISKOVALCI ZA RAZVOJ SAŠA REGIJE

RAZISKOVALNA NALOGA

GLASOVNO UPRAVLJANJE STORITVE SPOTIFY

Tematsko področje: Računalništvo

Avtor:
Tej Verbič, 4. letnik

Mentor:
Gregor Hrastnik, univ. dipl. inž. rač. in inf.

Velenje, 2026

Raziskovalna naloga je bila opravljena na Elektro in računalniški šoli Velenje, 2026.

Mentor: Gregor Hrastnik, univ. dipl. inž. rač. in inf.

Datum predstavitve:

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD Elektro in računalniška šola Velenje, šolsko leto 2025/2026

KG namizna aplikacija / programiranje /glasovno upravljanje

AV VERBIČ, Tej

SA HRASTNIK, Gregor

KZ 3320 Velenje, Trg mladosti 3

ZA ŠC Velenje, Elektro in računalniška šola, 2026

LI 2026

IN GLASOVNO UPRAVLJANJE STOEITVE SPOTIFY

TD Raziskovalna naloga

OP XI, 60 str., 3 pregl., 17 sl., 29 vir.

IJ SL

JI sl/en

AI Raziskovalna naloga obravnava razvoj namizne aplikacije za operacijski sistem Windows, ki omogoča glasovno upravljanje predvajanja glasbe v storitvi Spotify. Aplikacija je bila razvita z uporabo WPF in arhitekture MVVM, knjižnice Vosk za prepoznavanje govora ter Spotify Web API. Razvoj je vključeval načrtovanje uporabniškega vmesnika, implementacijo glasovnih ukazov ter povezavo med sistemom za prepoznavanje govora in upravljanjem predvajanja.V okviru testiranja sta bila analizirana odzivni čas in natančnost prepoznavanja ukazov ter izvedena primerjava z obstoječimi glasovnimi asistenti. Rezultati so pokazali povprečni odzivni čas 2,1 sekunde in 86-odstotno natančnost, kar predstavlja konkurenčno rešitev za namensko upravljanje Spotifyja. Med omejitvami so občutljivost na šum, podpora

le za angleški jezik in zahteva po naročnini Spotify Premium, hkrati pa rezultati potrjujejo potencial za nadaljnji razvoj in izboljšave

KEY WORDS DOCUMENTATION

ND Elektro in računalniška šola Velenje, school year 2025/2026

CX desktop application / programming / voice control

AU VERBIČ, Tej

AA HRASTNIK, Gregor

PP 3320 Velenje, Trg mladosti 3

PB ŠC Velenje, Elektro in računalniška šola, 2026

PY 2026

TI VOICE CONTROL OF THE SPOTIFY SERVICE

DT Research work

NO XI, 60 p., 3tab, 17 fig., 29 ref.

LA SL

AL sl/en

AB The research paper deals with the development of a desktop application for the Windows operating system that enables voice control of music playback in the Spotify service. The application was developed using WPF and the MVVM architecture, the Vosk speech recognition library, and the Spotify Web API. The development included user interface design, implementation of voice commands, and the connection between the speech recognition system and playback control.

As part of the testing, the response time and accuracy of command recognition were analyzed and a comparison was made with existing voice assistants. The results showed an average response time of 2.1 seconds and 86% accuracy, which represents a competitive solution for dedicated Spotify control. Among the limitations are noise sensitivity, support

for only the English language, and the requirement for a Spotify Premium subscription, while the results confirm the potential for further development and improvements.

KAZALO VSEBINE

KAZALO SLIK

KAZALO PROGRAMSKIH KOD

KAZALO TABEL

SEZNAM OKRAJŠAV

API – programski vmesnik (angl. Application Programming Interface)

MVVM – arhitekturni vzorec Model-View-ViewModel (angl. Model-View-ViewModel)

WPF – Windows Presentation Foundation, ogrodje za razvoj namiznih aplikacij (angl. Windows Presentation Foundation)

SDK – programski komplet (angl. Software Development Kit)

Vosk – odprtokodni sistem za prepoznavanje govora (angl. Open-source Speech Recognition Toolkit)

OAuth – odprti standard za avtentikacijo in avtorizacijo (angl. Open Authorization)

NLP – obdelava naravnega jezika (angl. Natural Language Processing)

GUI – grafični uporabniški vmesnik (angl. Graphical User Interface)

1.UVOD

V zadnjih letih se je uporaba glasovnih vmesnikov močno razširila, saj omogočajo bolj naravno, hitro in prostoročno interakcijo med uporabnikom in računalniškimi sistemi. Glasovno upravljanje postaja vse pomembnejše predvsem na področjih, kjer uporaba klasičnih vhodnih naprav, kot sta tipkovnica in miška, ni vedno praktična ali mogoča. Poseben poudarek ima glasovno upravljanje tudi na področju multimedije, kjer uporabniki pričakujejo enostaven in hiter dostop do vsebin.

Spotify je ena izmed najbolj priljubljenih platform za poslušanje glasbe na svetu in je široko uporabljena tudi v namiznem okolju Windows. Kljub razširjenosti glasovnih asistentov pa Spotify v okolju Windows ne ponuja namenske rešitve za neposredno glasovno upravljanje predvajanja z uporabo lastne namizne aplikacije. Obstoječe rešitve temeljijo predvsem na splošnih glasovnih asistentih, sistemskih funkcijah ali zunanjih napravah, kar pogosto zahteva dodatno konfiguracijo ali uporabo več različnih platform.

Problem, ki ga obravnava ta raziskovalna naloga, je pomanjkanje specializirane namizne aplikacije za glasovno upravljanje Spotifyja v operacijskem sistemu Windows, ki bi uporabniku omogočala neposreden nadzor predvajanja z uporabo naravnega jezika. Namen naloge je raziskati obstoječe možnosti glasovnega upravljanja Spotifyja, analizirati njihove prednosti in omejitve ter na tej podlagi razviti lastno rešitev v obliki WPF aplikacije, ki omogoča glasovno upravljanje predvajanja glasbe.

1.1 Hipoteze

Na podlagi opredeljenega problema in ciljev raziskovalne naloge sem oblikoval naslednje hipoteze:

Hipoteza 1:

Razvita aplikacija za upravljanje Spotifyja, omogoča enostavnejši in učinkovitejši nadzor v primerjavi s splošnimi Windows glasovnimi asistenti (npr. Windows Speech Recognition, Voice Access), ki niso prilagojeni specifičnim funkcijam glasbene aplikacije.

Hipoteza 2:

Uporaba neposredne programske povezave s storitvijo Spotify prek uradnega Spotify Web API omogoča bolj zanesljivo, natančno in funkcionalno upravljanje predvajanja glasbe kot rešitve, ki temeljijo na simulaciji uporabniških vhodov ali posrednih metodah upravljanja.

Hipoteza 3:

Glasovno upravljanje multimedijskih vsebin je na mobilnih napravah enostavnejše in bolj dostopno kot v namiznem okolju operacijskega sistema Windows.

2. PREGLED STANJA

Razvoj glasovnega upravljanja računalniških aplikacij je v zadnjih letih postal pomemben del interakcije med človekom in računalnikom. Napredek na področju prepoznavanja govora, obdelave naravnega jezika in umetne inteligence je omogočil uporabo glasovnih vmesnikov v različnih okoljih, predvsem na mobilnih napravah in pametnih zvočnikih. Kljub temu pa je podpora za glasovno upravljanje v namiznem okolju operacijskega sistema Windows še vedno omejena, zlasti pri specializiranih aplikacijah, kot je Spotify.

Spotify je ena najbolj razširjenih platform za poslušanje glasbe, vendar v aplikaciji za Windows ne ponuja neposrednega in namenskega glasovnega upravljanja. Uporabniki so zato primorani uporabljati zunanje rešitve, sistemske funkcije ali dodatna orodja, kar pogosto vpliva na uporabniško izkušnjo in zahtevnost uporabe.

2.1 Pristopi k reševanju problema v obstoječih rešitvah

Različni ponudniki so problem razdrobljenosti reševali na več načinov:

Sistemske rešitve:
Windows Speech Recognition in Voice Access omogočata glasovno upravljanje uporabniškega vmesnika, vendar ne nudita razumevanja konteksta ali naravnega jezika.

Splošni glasovni asistenti:
Rešitve, kot je Braina , omogočajo glasovno interakcijo, vendar niso namensko razvite za Spotify in pogosto zahtevajo dodatne nastavitve ali integracije.

Strojno omejene rešitve:
Nvidia G-Assist ponuja napredne AI funkcije, vendar je omejen na uporabnike določene strojne opreme.

Na podlagi pregleda stanja tehnike je razvidno, da v okolju Windows ne obstaja namensko razvita aplikacija, ki bi združevala enostavno uporabo, neposredno integracijo s Spotify API in podporo za naravni glasovni jezik. To predstavlja jasno potrebo in utemeljitev za razvoj lastne WPF aplikacije za glasovno upravljanje Spotifyja.

2.2 Pregled sorodnih rešitev

Umetna inteligenca in glasovno upravljanje se že uporabljata v različnih obstoječih rešitvah, ki omogočajo delno ali posredno upravljanje aplikacij v okolju Windows:

Windows Speech Recognition / Voice Access

Windows Speech Recognition (WSR) je Microsoftova vgrajena tehnologija za prepoznavanje govora v sistemu Windows (na voljo v Windows 10 in starejših), medtem ko je Voice Access nova funkcija v Windows 11 (od različice 22H2 dalje) za celovito glasovno upravljanje celotnega sistema. Namestitev je preprosta, v Windows 10 preko Nadzorne plošče (Control Panel) ali nastavitev "Ease of Access" za vklop WSR, v Windows 11 pa preko Settings > Accessibility > Speech vklopite Voice Access. Pri tem je potrebna le kakovostna mikrofonska naprava in osnovna kalibracija mikrofona (Wizard, ki vas vodi skozi nastavitve). Podprti jeziki so omejeni na glavne svetovne jezike: angleščino (različne dialekte), francoščino, nemščino, japonščino, kitajščino in španščino. [9]

Braina

Braina je komercialni AI asistent za Windows, ki združuje prepoznavanje govora z različnimi funkcijami (diktat, ukazi, skripte). Namestitev je standardna (Windows 10/11, zahteva mikrofonsko opremo) in na voljo je brezplačna "Lite" različica ter plačljiva Pro z dodatnimi zmožnostmi. Po zagonu je treba program konfigurirati tako, da sprejema glasovne ukaze (nastavite jezik, mikrofonski vhod), zahteva pa tudi prisotnost brskalnika Chrome, saj Braina uporablja Googlov mehanizem za prepoznavanje govora. Braina podpira veliko jezikov (približno 90) in omogoča učenje na podlagi vašega glasu za višjo natančnost. [11]

VoiceAttack

VoiceAttack je namizna aplikacija (samo Windows) za kreiranje glasovnih ukazov, pogosto uporabljena v igrah. Namestitev jo enaka kot za vsak Windows program, za nemoteno delovanje je potreben mikrofon in Windows Speech Recognition, ki uporablja vgrajeni WSR-stroj (ta je na voljo od Windows 7 naprej). Zasnovan je za delo s profili in lastnimi ukazi; vsak ukaz je lahko povezan z akcijo (tipkovne bližnjice, klik ali celo skripto). [12]

Spotify Connect

Spotify Connect ni glasovno orodje, ampak funkcija Spotifyja za brezžično pretakanje glasbe na združljive naprave. Deluje tako, da lahko glasbo, ki teče v Spotify aplikaciji na eni napravi, nemoteno nadaljujete na drugi napravi v omrežju. Za uporabo Spotify Connect potrebujete urejajočo napravo (telefon, računalnik) z aplikacijo Spotify in združljive sprejemniške naprave (zvočnike, AV sprejemnik, telefon, smart TV ipd.). [1]

NVIDIA G-Assist

Project G-Assist je eksperimentalni glasovni asistent podjetja NVIDIA za pogone na GeForce RTX karticah. Deluje v okviru GeForce Experience (Windows 10+), kjer ga aktiviramo prek opcije “Discover” ali tipke Alt+G. Sistemske zahteve so relativno visoke (potreben je RTX grafični procesor s vsaj 6 GB VRAM). Namestitev poteka preko glavne NVIDIA aplikacije, nato pa se G-Assist zažene kot prekritje nad igrami ali aplikacijami. Z uporabo lokalnega Llama 8B modela (8 milijard parametrcev) deluje povsem lokalno, brez povezave v oblak. Sliši in razume glasovne ukaze v realnem času, ter zmore upravljati nastavitve (npr. »optimiziraj igro za kakovost«, »glej temperaturo GPU«). [15]

Siri (Apple)

Glasovni asistent podjetja Apple, namenjen upravljanju naprav z operacijskimi sistemi iOS, macOS in watchOS. Siri omogoča glasovno upravljanje predvajanja glasbe na Spotifyju na podprtih napravah, vendar je njena uporaba omejena na Applov ekosistem. Integracija je odvisna od sistemskih omejitev, uporabnik pa nima nadzora nad naprednejšimi funkcijami Spotifyja ali prilagoditvijo glasovnih ukazov, zlasti v okolju operacijskega sistema Windows. [28]

Bixby (Samsung)

Bixby je glasovni asistent, razvit s strani podjetja Samsung, in je vgrajen predvsem v mobilne naprave in pametne televizorje tega proizvajalca. Omogoča osnovno upravljanje glasbe in integracijo z določenimi aplikacijami, vključno s Spotifyjem, vendar je njegova uporaba omejena na Samsungovo strojno in programsko okolje. Bixby ne omogoča neposrednega glasovnega upravljanja Spotifyja v namiznem okolju Windows. [18]

Amazon Alexa

Amazon Alexa je oblačni glasovni asistent, ki omogoča glasovno upravljanje različnih storitev in pametnih naprav. Podpira upravljanje Spotifyja prek glasovnih ukazov, vendar deluje predvsem na pametnih zvočnikih in mobilnih napravah. Upravljanje predvajanja v namizni aplikaciji Spotify za Windows ni neposredno podprto, poleg tega pa rešitev zahteva stalno internetno povezavo in uporabo zunanjih naprav. [29]

2.3 Problem razdrobljenosti glasovnega upravljanja v okolju Windows

Problem razdrobljenosti se nanaša na situacijo, kjer za dosego ene funkcionalnosti uporabnik potrebuje več različnih orodij ali platform. V primeru glasovnega upravljanja Spotifyja v okolju Windows ni na voljo enotne rešitve, ki bi omogočala preprosto in neposredno uporabo glasovnih ukazov. Obstoječe rešitve so pogosto splošne, niso namensko prilagojene Spotifyju ali pa zahtevajo obsežno konfiguracijo.

Ta problem je še posebej izrazit pri uporabnikih, ki želijo hitro in intuitivno upravljanje predvajanja glasbe brez uporabe tipkovnice ali miške. Namesto enotne rešitve so prisiljeni v uporabo sistemskih funkcij, glasovnih asistentov ali zunanjih naprav, kar zmanjšuje uporabnost in dostopnost.

3. MATERIALI IN METODE DELA

Poglavje opisuje materiale, orodja in metode, uporabljene pri razvoju namizne aplikacije za glasovno upravljanje predvajanja glasbe prek Spotify API. Predstavljena je izbira programskih jezikov, razvojnega okolja in ogrodij, kot tudi integracija prepoznavanja govora z Voskom ter povezava s Spotify Web API. Nadalje je prikazana arhitektura aplikacije, načrtovanje uporabniškega vmesnika in potek implementacije ključnih funkcionalnosti. Poglavje se zaključi s primerjalno analizo obstoječih rešitev, ki omogoča oceno prednosti in omejitev razvite aplikacije.

3.1 Izbira tehnologij

Pri razvoju aplikacije sem se odločil uporabiti Visual Studio 2022, ki omogoča celovito upravljanje projekta, integracijo z različnimi orodji in enostavno odkrivanje napak. Za izdelavo uporabniškega vmesnika sem izbral WPF (Windows Presentation Foundation), ker omogoča ustvarjanje sodobnih in vizualno privlačnih vmesnikov, hkrati pa podpira ločitev logike aplikacije od vizualnega vmesnika preko vzorca MVVM (Model-View-ViewModel).

Za glasovno prepoznavanje sem uporabil Vosk, odprtokodni sistem, ki omogoča lokalno prepoznavanje govora. Prednost Voska je hitrost in zasebnost, saj se zvok obdeluje neposredno na računalniku, brez potrebe po povezavi z internetom. Povezavo z glasbo in predvajalnikom zagotavlja Spotify Web API, ki omogoča nadzor predvajanja, iskanje skladb, upravljanje seznamov predvajanja ter dostop do uporabniškega profila.

3.2 Uporabljeni materiali in programska orodja

Pri razvoju namizne aplikacije za glasovno upravljanje predvajanja glasbe v okolju Windows so bila uporabljena sodobna programska orodja in ogrodja, ki omogočajo stabilen razvoj, učinkovito upravljanje uporabniškega vmesnika ter enostavno integracijo zunanjih storitev. Izbrana ogrodja podpirajo razvoj aplikacij v programskem jeziku C#, uporabo arhitekture WPF ter povezovanje s storitvami za prepoznavanje govora in Spotify Web API, kar omogoča zanesljivo delovanje aplikacije in razširljivost rešitve.

3.2.1 Programski jezik C#

C# (izgovorjava *C sharp*) je objektno usmerjen programski jezik, ki ga je razvilo podjetje Microsoft. Namenjen je razvoju različnih vrst aplikacij, kot so namizne aplikacije, spletne aplikacije, mobilne aplikacije ter igre. C# se najpogosteje uporablja v razvojnem okolju .NET, ki ponuja številna orodja in knjižnice za učinkovito programiranje.

Programi v jeziku C# so sestavljeni iz razredov in metod, ki omogočajo jasno strukturo in dobro preglednost kode. Spodnji primer kode prikazuje preprost program, ki izpiše besedilo na zaslon:

```
4
5
6  Console.WriteLine("Pozdravljen svet!");
```

Koda 1: Osnovni ukaz za izpis besedila v konzolno okno v jeziku C# (Vir: lasten)

C# podpira številne sodobne programske koncepte, kot so dedovanje, polimorfizem, enkapsulacija ter delo z dogodki in podatkovnimi bazami. Zaradi strogega preverjanja napak in močne tipizacije je jezik primeren tudi za večje in kompleksnejše projekte.

Poleg tega se C# pogosto uporablja v povezavi z ogrodji, kot so ASP.NET za razvoj spletnih aplikacij ter Unity za razvoj računalniških iger. Zaradi svoje vsestranskosti, zanesljivosti in široke podpore skupnosti je C# eden izmed najpogosteje uporabljenih programskih jezikov v sodobnem razvoju programske opreme. [25]

3.2.2 XAML (eXtensible Application Markup Language)

XAML (eXtensible Application Markup Language) je označevalni jezik, ki se uporablja za opis uporabniških vmesnikov v aplikacijah, razvitih v okolju .NET. Najpogosteje se uporablja skupaj s programskim jezikom C# pri razvoju namiznih in mobilnih aplikacij, kot so WPF, UWP in Xamarin.

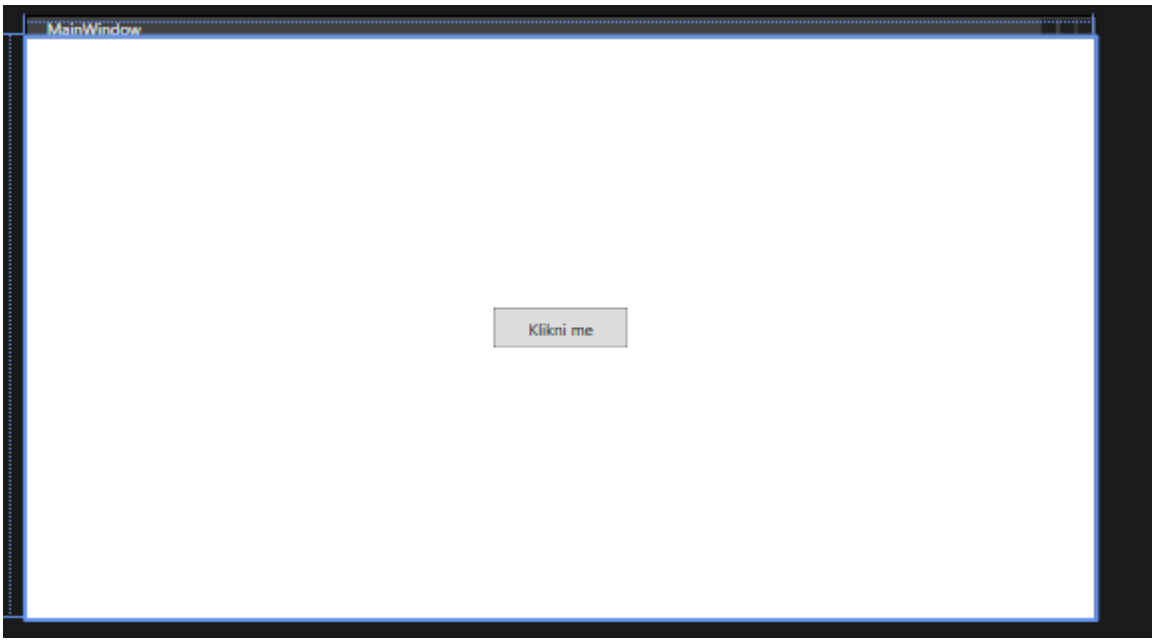
XAML omogoča ločevanje vizualnega dela aplikacije od programske logike. Z njim razvijalec določa razporeditev elementov uporabniškega vmesnika, kot so gumbi, besedilna polja, seznami in drugi grafični elementi. Spodnji primer kode XAML prikazuje preprost gumb z besedilom:

```
<Grid>

    <Button Content="Klikni me" Width="100" Height="30" />

</Grid>
```

Koda 2: Definicija gumba znotraj gradnika Grid v jeziku XAML (Vir: lasten)



Slika 1: Vizualni prikaz uporabniškega vmesnika z gumbom v okolju WPF (Vir: lasten)

Elementi v jeziku XAML so zapisani v obliki oznak, podobno kot pri HTML, vendar so tesno povezani z objekti in razredi v okolju .NET. Vsak element ima lahko lastnosti (atribute), s katerimi se določa njegov videz in obnašanje.

Poleg oblikovanja uporabniškega vmesnika XAML podpira tudi napredne funkcionalnosti, kot so podatkovno vezanje (data binding), slogi (styles) in predloge (templates), kar omogoča razvoj preglednih, prilagodljivih in razširljivih aplikacij. Zaradi tega je XAML pomemben del razvoja sodobnih aplikacij v Microsoftovem ekosistemu. [26]

3.2.3 WPF (Windows Presentation Foundation)

WPF (Windows Presentation Foundation) je ogrodje za razvoj namiznih aplikacij v operacijskem sistemu Windows. Razvilo ga je podjetje Microsoft in je del ogrodja .NET. WPF omogoča izdelavo sodobnih in grafično bogatih uporabniških vmesnikov, ki podpirajo animacije, večpredstavnost, podatkovno vezanje ter prilagodljivo postavitev elementov.

WPF temelji na uporabi programskega jezika C# in označevalnega jezika XAML, kjer XAML določa strukturo in videz uporabniškega vmesnika, C# pa logiko aplikacije. Takšna ločitev omogoča boljšo preglednost kode in lažje vzdrževanje aplikacije.

Pri razvoju aplikacij v WPF se pogosto uporablja arhitekturni vzorec MVVM (Model-View-ViewModel), ki dodatno loči uporabniški vmesnik od poslovne logike. MVVM vzorec omogoča bolj modularno in pregledno kodo ter olajša razvoj večjih aplikacij.

Primer uporabe vzorca MVVM v WPF

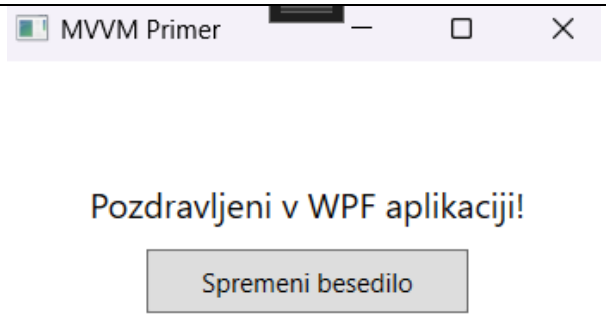
V nadaljevanju je prikazan preprost primer WPF aplikacije, ki uporablja vzorec MVVM. Aplikacija vsebuje gumb, ki ob kliku spremeni besedilo na zaslonu.

Najprej je prikazan View (XAML), ki definira uporabniški vmesnik:

```
<Window x:Class="WpfApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MVVM Primer" Height="200" Width="300">

    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <TextBlock Text="{Binding Message}" FontSize="16" Margin="10" />
        <Button Content="Spremeni besedilo"
                Command="{Binding ChangeMessageCommand}"
                Width="150" Height="30" />
    </StackPanel>
</Window>
```

Koda 3: Primer uporabe vzorca MVVM za povezovanje podatkov (Data Binding) in ukazov (Commands) (Vir: lasten)



Slika 2: Grafični izris elementov TextBlock in Button na podlagi XAML definicije iz Kode 3 (Vir: lasten)

V XAML kodi je razvidno podatkovno vezanje (data binding). Lastnost Text elementa TextBlock je povezana z lastnostjo Message v ViewModelu, gumb pa uporablja ukaz (Command), ki se prav tako nahaja v ViewModelu.

Sledi ViewModel, ki vsebuje logiko aplikacije:

```
namespace WpfApp
{
    1 reference
    public class MainViewModel : INotifyPropertyChanged
    {
        private string _message = "Pozdravljeni v WPF aplikaciji!";

        2 references
        public string Message
        {
            get => _message;
            set
            {
                _message = value;
                OnPropertyChanged(nameof(Message));
            }
        }

        1 reference
        public ICommand ChangeMessageCommand { get; }

        1 reference
        public MainViewModel()
        {
            ChangeMessageCommand = new RelayCommand(ChangeMessage);
        }

        1 reference
        private void ChangeMessage()
        {
            Message = "Besedilo je bilo spremenjeno!";
        }

        public event PropertyChangedEventHandler PropertyChanged;

        1 reference
        protected void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Koda 4: Primer ViewModela v WPF (Vir: lasten)

ViewModel vsebuje lastnost Message, ki predstavlja stanje aplikacije, in ukaz ChangeMessageCommand, ki se sproži ob kliku na gumb. Z vmesnikom INotifyPropertyChanged se uporabniški vmesnik samodejno posodobi, ko se vrednost lastnosti spremeni.

Za povezavo ViewModela z Viewom se ViewModel nastavi kot DataContext:

```
namespace WpfApp
{
    2 references
    public partial class MainWindow : Window
    {
        0 references
        public MainWindow()
        {
            InitializeComponent();
            DataContext = new MainViewModel();
        }
    }
}
```

Koda 5: Nastavitev ViewModela kot DataContext (Vir: lasten)

Prednosti ogrodja WPF

Ena izmed glavnih prednosti WPF je podpora vzorcu MVVM, ki omogoča jasno ločevanje uporabniškega vmesnika, logike in podatkov. To olajša razvoj, testiranje in vzdrževanje aplikacij.

WPF uporablja grafični pogon DirectX, kar omogoča strojno pospešeno izrisovanje grafike in gladko delovanje tudi pri zahtevnejših uporabniških vmesnikih. Poleg tega WPF podpira ponovno uporabo slogov, predlog in kontrol, kar zmanjša količino kode.

Slabosti ogrodja WPF

Ena glavnih slabosti WPF je večja kompleksnost v primerjavi z enostavnejšimi tehnologijami. Razumevanje XAML, podatkovnega vezanja in MVVM vzorca zahteva več časa in znanja, kar je lahko izziv za začetnike.

Prav tako je WPF vezan na operacijski sistem Windows, kar omejuje uporabo aplikacij na druge platforme.

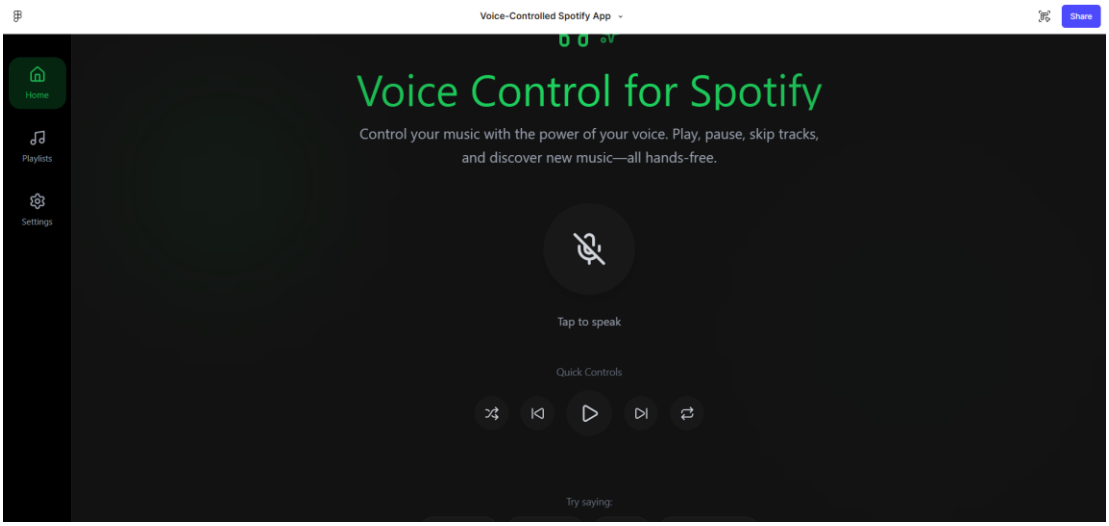
Zakaj WPF?

WPF je primerna izbira za razvoj profesionalnih namiznih aplikacij, kjer so pomembni zmogljiv uporabniški vmesnik, dobra struktura kode in dolgoročna vzdržljivost projekta. Uporaba MVVM vzorca omogoča modularen in razširljiv razvoj, kar je še posebej pomembno pri večjih aplikacijah.

Zaradi teh lastnosti je WPF ustrezna tehnologija za izdelavo aplikacije, razvite v okviru te raziskovalne naloge. [13]

3.2.4 Figma

Figma je spletna platforma za oblikovanje uporabniških vmesnikov in prototipov, ki omogoča sodelovanje več članov ekipe v realnem času. Omogoča vizualno oblikovanje elementov vmesnika, ustvarjanje interaktivnih prototipov ter uporabo komponent in knjižnic za enotno oblikovanje. Platforma poenostavi testiranje uporabniške izkušnje, izboljša komunikacijo med oblikovalci in razvijalci ter pospeši prenos dizajna v kodo. Pri projektu je Figma omogočila jasno definicijo videza, postavitve in interaktivnosti vmesnika pred implementacijo v WPF. [27]



Slika 3: Grafični načrt uporabniškega vmesnika za glasovno upravljanje aplikacije Spotify, izdelan v orodju Figma (Vir: lasten)

3.2.5 Vosk – knjižnica za prepoznavanje govora

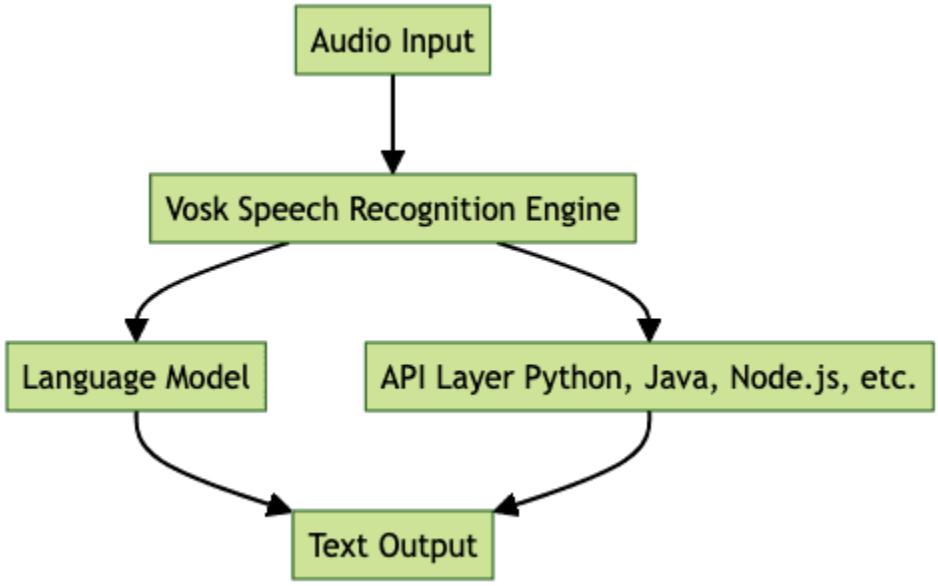
Vosk je odprtokodna knjižnica za prepoznavanje govora, ki omogoča pretvorbo govora v besedilo v realnem času. Glavna prednost Voska je, da je zasnovan tako, da deluje lokalno na računalniku, brez potrebe po stalni povezavi z internetom ali zunanjimi strežniki. To zagotavlja hitrost, nizko zakasnitev in visoko stopnjo zasebnosti, saj se zvočni podatki ne pošiljajo v oblak.

Vosk uporablja napredne modele strojnega učenja, ki so predhodno naučeni prepoznati različne glasove, naglase in jezike. Ko uporabnik govori, Vosk zajame zvočni signal, ga pretvori v niz značilnosti in primerja z notranjim modelom govora. Na podlagi tega modela nato generira besedilo, ki ustreza prepoznanemu govoru. Knjižnica omogoča tudi delno prepoznavanje, kjer sproti posodablja tekst, preden uporabnik zaključi govor, kar omogoča bolj tekoče in odzivno uporabniško izkušnjo.

Vosk podpira številne jezike in dialekte, kar omogoča uporabo v različnih regijah in za različne uporabnike. Modeli so na voljo kot samostojne datoteke, ki jih lahko prenesemo in vključimo v aplikacijo. To omogoča fleksibilnost, saj lahko razvijalec izbere model glede na jezikovno okolje ali zahtevano natančnost.

Integracija v aplikacije

Vosk je na voljo za različne programske jezike, vključno s C#, Python, Java in C++. V kontekstu aplikacij, kot so glasovno upravljanje ali interaktivni vmesniki, Vosk omogoča prepoznavanje ukazov uporabnika v realnem času. Ko je govor pretvorjen v besedilo, ga lahko aplikacija dodatno obdeluje, na primer za izvajanje določenih funkcij, iskanje informacij ali nadzor predvajanja glasbe.



Slika 4: Diagram delovanja knjižnice Vosk(20)

Glavne funkcionalnosti

- **Prepoznavanje govora v realnem času:** Vosk lahko sproti prepoznava govor in generira besedilo brez opazne zamude.
- **Delno prepoznavanje:** Besedilo se posodablja sproti, kar omogoča interaktivno odzivanje aplikacije na govor.
- **Podpora za različne jezike in modele:** Razvijalci lahko izberejo jezik in natančnost modela glede na potrebe uporabnika.
- **Integracija s C# in drugimi jeziki:** Knjižnica ponuja enostavne API za integracijo v aplikacije WPF, WinForms ali druge platforme.

Prednosti in slabosti Vosk

Prednosti:

Prednosti Vosk-a so večplastne. Ena izmed ključnih prednosti je, da deluje lokalno, kar pomeni, da ni potrebna internetna povezava, kar povečuje zanesljivost in varnost podatkov. Poleg tega omogoča hitro odzivnost z nizko zakasnitvijo, kar je še posebej pomembno pri aplikacijah za glasovno upravljanje v realnem času. Vosk podpira več jezikov in dialektov, kar omogoča široko

uporabo v različnih okoljih in mednarodnih projektih. Prav tako je njegova integracija z obstoječimi aplikacijami preprosta, kar razvijalcem omogoča hitro vključitev funkcionalnosti prepoznavanja govora brez večjih sprememb v obstoječi kodi.

Slabosti:

Slabosti Vosk-a se kažejo predvsem pri zahtevnejših primerih uporabe. Kakovost prepoznavanja je močno odvisna od izbranega modela, kar pomeni, da manjši ali osnovni modeli morda ne bodo zagotavljali ustrezne natančnosti za kompleksne ukaze ali daljše besedilne vnose. Za zelo velike aplikacije ali za doseganje visoke natančnosti so pogosto potrebni zmogljivejši modeli, ki zahtevajo več pomnilnika in računalniških virov. To lahko predstavlja omejitev pri uporabi na šibkejših napravah ali v okoljih z omejenimi sistemskimi sredstvi. [20]

Slika 5: Logo Vosk (21)

3.1.6 Spotify API

Spotify API predstavlja celovit sklop vmesnikov in orodij, ki omogočajo razvijalcem integracijo Spotify vsebin in funkcionalnosti v svoje aplikacije. Z njegovo pomočjo je mogoče pridobivati podatke o glasbi, izvajalcih, albumih, seznamih predvajanja in uporabniških profilih, upravljati predvajalnik ter prilagajati uporabniško izkušnjo. Spotify razvijalcem ponuja različne možnosti, od Web API, ki omogoča RESTful dostop prek interneta, do SDK za mobilne in spletne aplikacije, ki omogočajo lokalno predvajanje in integracijo z obstoječimi predvajalniki.

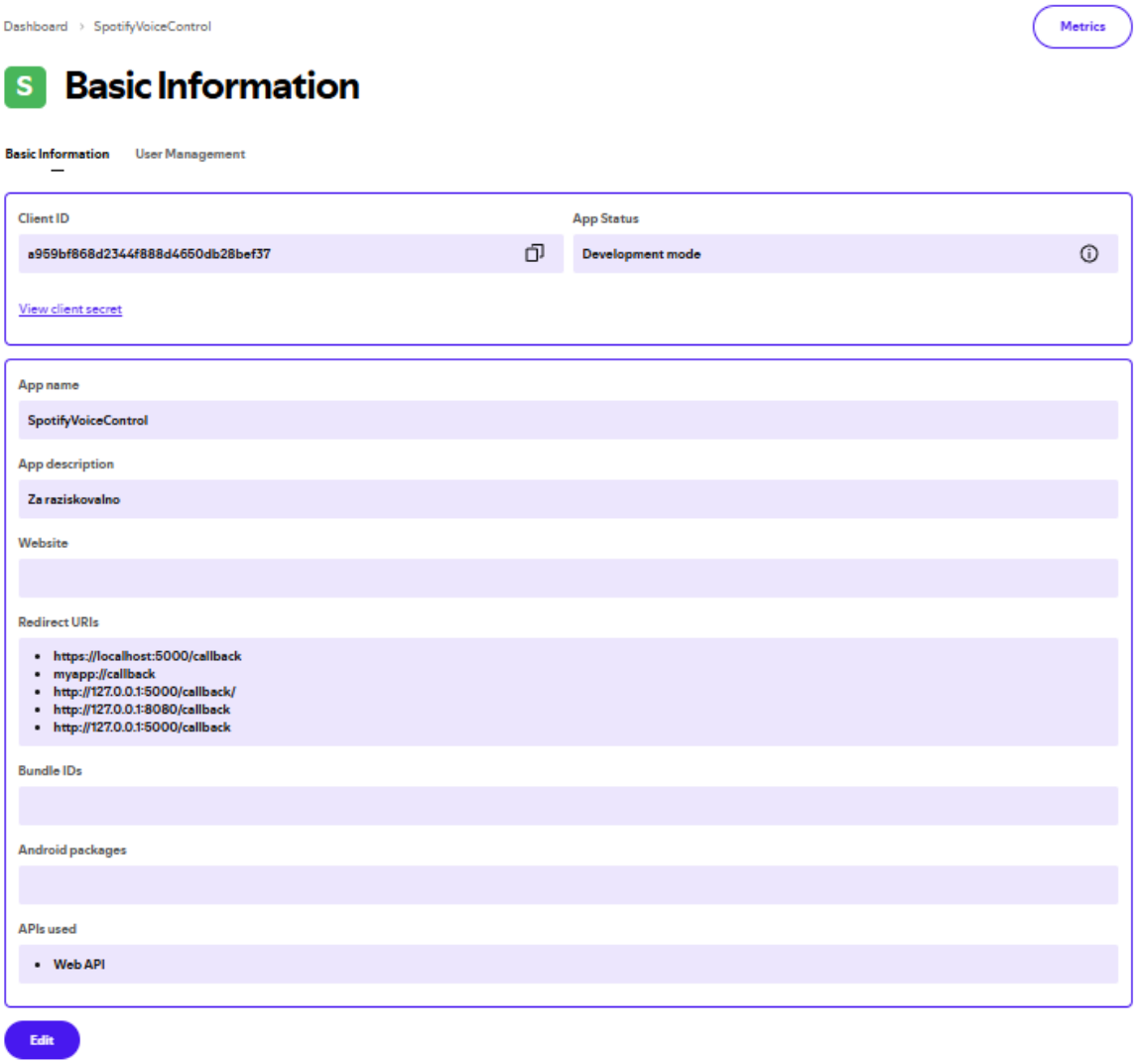
3.1.6.1 Spotify Web API

Spotify Web API je RESTful vmesnik, ki omogoča aplikacijam dostop do Spotifyjevih vsebin (glasbe, podcastov) in nadzor nad predvajanjem. Z njim lahko aplikacije pridobivajo metapodatke (npr. o izvajalcih, albumih, skladbah) in upravljajo s predvajalnikom uporabnika. Na primer, API omogoča iskanje glasbenih vsebin, upravljanje seznamov predvajanja ter predvajanje ali pauziranje glasbe na povezanih napravah.

- **Pridobivanje podatkov:** Pojavniki (endpoints) kot GET /v1/artists/{id} ali GET /v1/albums/{id} vrnejo metapodatke o izvajalcu oziroma albumu.
- **Iskanje:** Z ukazom GET /v1/search se lahko išče po katalogu (izvajalci, skladbe, albume, podcasti itd.) glede na iskalni niz.
- **Nadzor predvajanja:** Aplikacije lahko upravljajo uporabnikov predvajalnik (na primer predvajajo ali ustavljajo skladbe, spreminjajo glasnost, mešanje, ponavljanje ipd.) prek ukazov kot so PUT /v1/me/player/play ali POST /v1/me/player/pause.
- **Upravljanje knjižnice in seznamov predvajanja:** API omogoča ustvarjanje, spreminjanje in branje uporabniških seznamov predvajanja (npr. POST /v1/users/{user_id}/playlists za nov seznam) ter shranjevanje skladb v uporabnikovo knjižnico in njihovo odstranjevanje.
- **Uporabniški profil:** Z GET /v1/me aplikacija pridobi informacije o trenutnem uporabniku (ime, e-pošta, njegovi seznami predvajanja ipd.).

Začetek in registracija aplikacije

Za uporabo Spotify Web API je potrebno najprej ustvariti aplikacijo na Spotify Developer Dashboard, kjer se ob registraciji pridobita podatka Client ID in Client Secret. Ta podatka predstavljata identifikacijo aplikacije in sta ključni za izvedbo avtorizacijskega postopka OAuth, s katerim se pridobi dostopni žeton (access token). Pridobljeni žeton se nato uporablja pri vseh API zahtevkih, saj se pošlje v glavi klica v obliki Authorization: Bearer <token>. Ker je veljavnost dostopnega žetona časovno omejena (približno ena ura), ga je po preteku potrebno osvežiti ali ponovno pridobiti, da aplikacija ohrani dostop do Spotifyjevih funkcij.



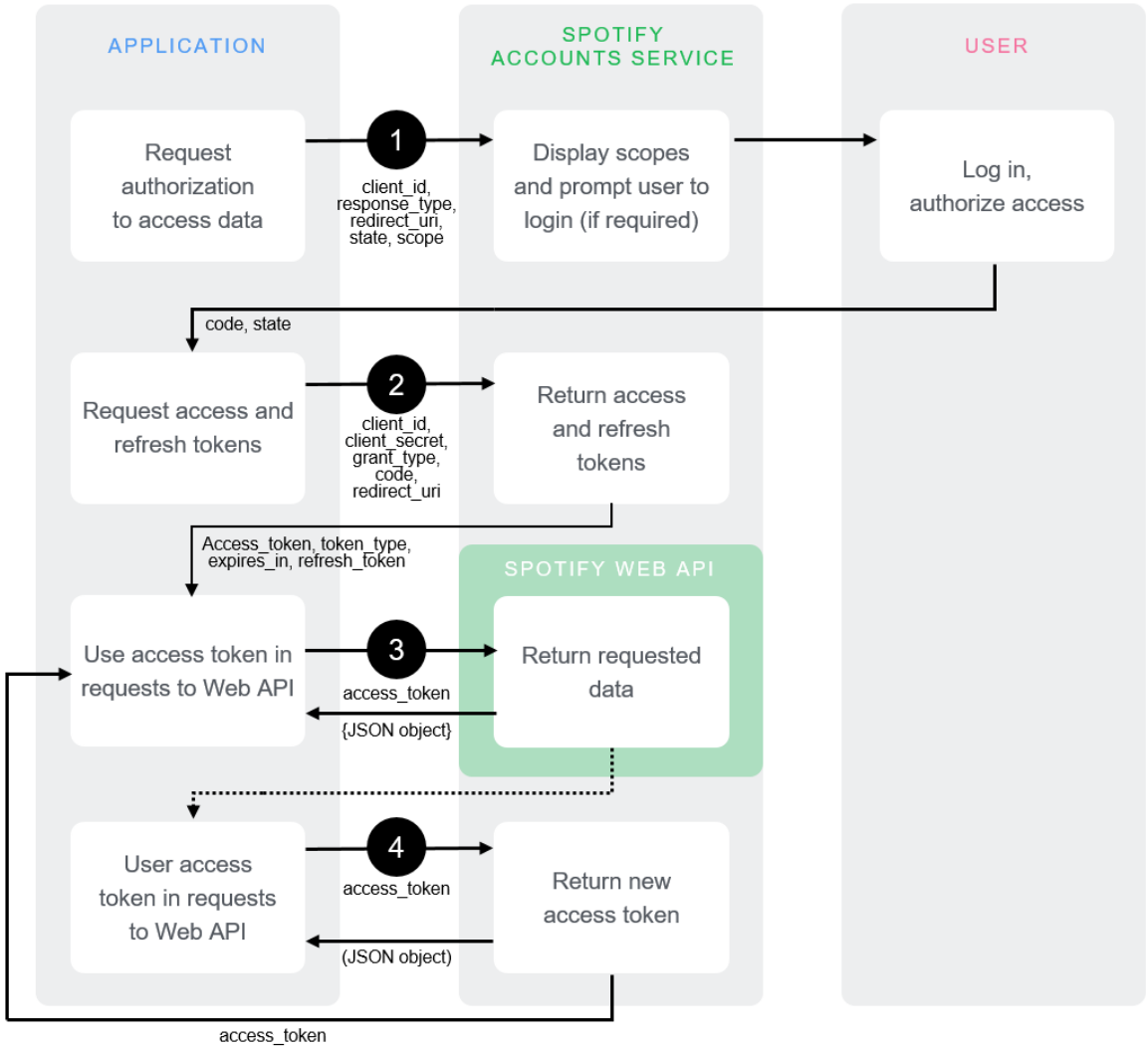
Slika 5: Registracija in konfiguracija aplikacije v portalu Spotify for Developers (Vir: lasten)

Avtorizacija (OAuth 2.0)

Spotify uporablja OAuth 2.0 za avtorizacijo aplikacij na strani uporabnika. Aplikacija mora pri uporabniku zaprositi za dovoljenja, imenovana scope (obseg pravic), ki definirajo, do katerih podatkov ali funkcij lahko dostopa. Na primer, zahteva za scope user-read-playback-state omogoča aplikaciji dostop do podatkov o trenutnem stanju predvajanja, scope playlist-modify-public dovoljuje ustvarjanje javnih seznamov predvajanja, user-library-read pa branje shranjenih skladb v uporabnikovi knjižnici. Po potrditvi dovoljenj s strani uporabnika Spotify izda dostopni žeton s tem obsegom, ki ga aplikacija uporablja naprej.

Za pridobivanje žetona so na voljo različni OAuth tokovi (flow):

- **Authorization Code:** standarden način za strežniške aplikacije, kjer aplikacija prejme kodo po uporabniški prijavi in jo nato zamenja za žeton. Uporablja se, če aplikacija lahko varno shrani skrivnostni ključ (Client Secret).
- **PKCE (modificirani Authorization Code):** namenjen mobilnim ali namiznim aplikacijam, ki doda dodatne varnostne ukrepe (enkratno kodo) za zaščito pred prestrezanjem.
- **Client Credentials:** uporablja se, če aplikacija ne dostopa do uporabniških podatkov (npr. strežniki ali skripte). V tem primeru aplikacija zamenja svoje poverilnice neposredno za žeton in ne potrebuje soglasja uporabnika.



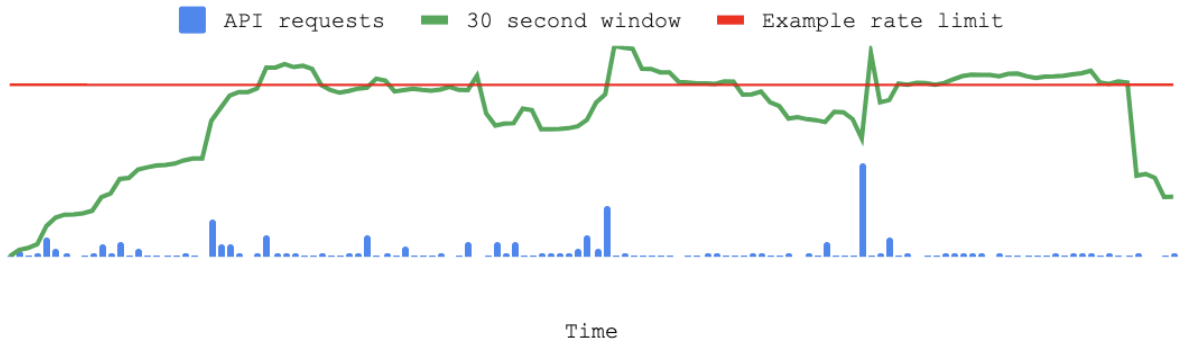
Slika 6: Potek avtorizacijskega protokola OAuth 2.0 (4)

Kvote in omejitve (rate limits)

Spotify nadzoruje število API zahtevkov, ki jih aplikacija lahko pošlje v kratkem času. Če aplikacija prekorači omejitev (število klicev v 30-sekundnem oknu), bo prejela HTTP napako 429 Too Many Requests. V glavi odgovora bo običajno polje Retry-After, ki pove, koliko sekund je treba počakati pred ponovnim poizkusom.

Spotify aplikacije po registraciji začnejo v razvojnem načinu, kar pomeni, da lahko do njih dostopa največ 25 uporabnikov (ti morajo biti dodani na dovoljen seznam). Če aplikacija pridobi odobritev

od Spotify (in izpolnjuje pogoje), se lahko preklopi v razširjen način. V razširjenem načinu ni omejitve števila uporabnikov in aplikaciji so dodeljene višje kvote (večji omejitveni prag).



Slika 7: Grafični prikaz omejitev števila zahtev (Rate Limits) v določenem časovnem oknu (5)

Uporaba API v praksi

Vsi API klici se izvajajo prek protokola HTTPS na osnovni URL naslov https://api.spotify.com/v1/

Na primer, klic GET /v1/tracks/{id} pridobi metapodatke o skladbi z določenim identifikatorjem. V glavi vsake zahteve je potrebno vključiti parameter Authorization: Bearer <Access Token>, ki predstavlja avtorizacijo uporabnika. Odgovori API so v formatu JSON in vsebujejo podrobne informacije o zahtevani vsebini, kot so ime skladbe, izvajalec, album, slike albuma, trajanje, priljubljenost in drugi metapodatki.

Spotify Web API omogoča obsežno integracijo z glasbenim katalogom in uporabniškimi računi. Z njegovo uporabo lahko aplikacije pridobivajo podatke o glasbi, izvajajo iskanje vsebin, upravljajo predvajanje, dostopajo do uporabnikovih seznamov predvajanja ter prilagajajo uporabniško izkušnjo. Pri tem je potrebno upoštevati ustrezno avtorizacijo uporabnika ter omejitve števila zahtevkov.

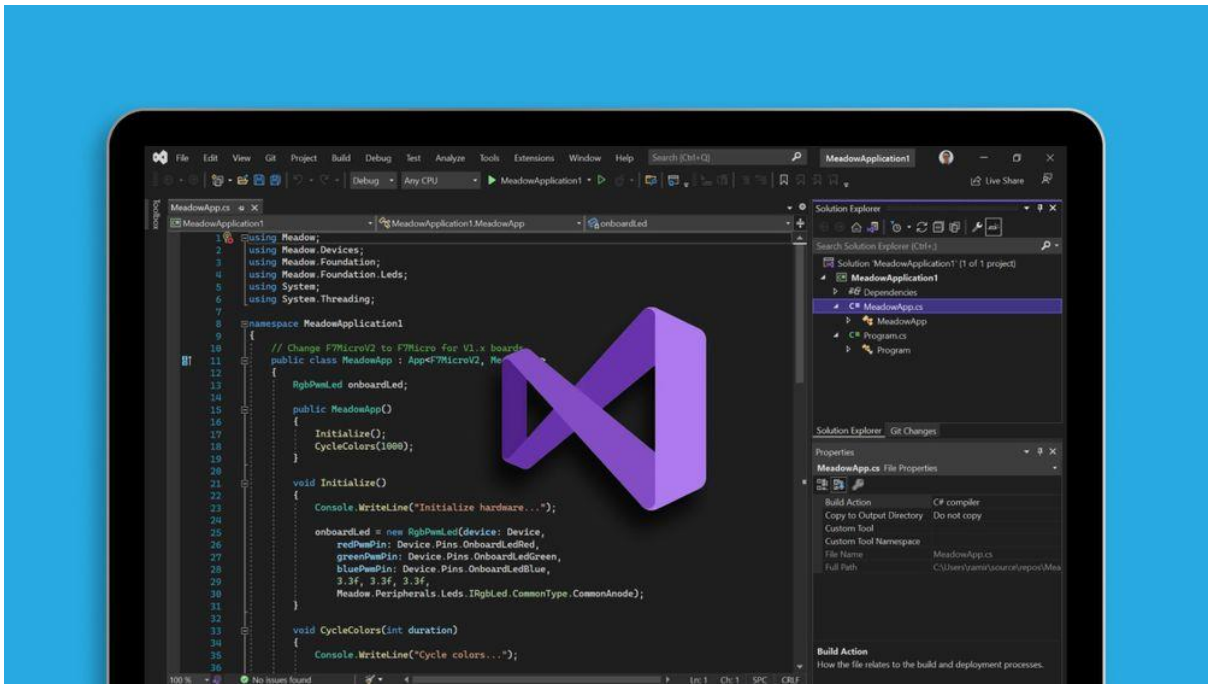
Za uporabo nekaterih funkcionalnosti Spotify Web API je potreben Spotify Premium račun. Funkcije, povezane z neposrednim upravljanjem predvajanja, niso na voljo uporabnikom brez Premium naročnine. Med funkcionalnosti, ki zahtevajo Premium, sodijo:

- zagon ali nadaljevanje predvajanja na izbrani napravi (Resume Playback),
- začasna zaustavitev predvajanja (Pause Playback),
- preskok na naslednjo ali prejšnjo skladbo (Skip Next / Previous),
- predvajanje določene skladbe, albuma ali seznama predvajanja na zahtevo,
- nastavitev naključnega predvajanja (Shuffle) in ponavljanja (Repeat),
- upravljanje predvajanja na različnih napravah (Device Control).

Uporabniki brez Premium računa lahko prek API še vedno dostopajo do podatkov o glasbi, izvajajo iskanje ter upravljajo svoje knjižnice in sezname predvajanja, vendar ne morejo neposredno nadzorovati predvajanja. Ta omejitev je pomembna pri načrtovanju aplikacij, ki temeljijo na upravljanju glasbe, saj vpliva na razpoložljivost ključnih funkcij. [1]

3.2.7 Visual Studio 2022

Visual Studio 2022 je zmogljivo razvojno okolje podjetja Microsoft za ustvarjanje aplikacij za Windows, splet in mobilne naprave. Omogoča pisanje kode v različnih jezikih (C#, C++, Visual Basic, F#), razhroščevanje, vizualno oblikovanje uporabniškega vmesnika ter integracijo z zunanjimi knjižnicami in sistemi za nadzor verzij. Podpira razvoj namiznih aplikacij (WPF, WinForms), spletnih aplikacij, mobilnih aplikacij (Xamarin, MAUI) in iger (Unity), kar omogoča celosten razvoj v enem okolju. Visual Studio poenostavi sodelovanje, testiranje in objavo aplikacij. [22]



Slika 8: Integrirano razvojno okolje Visual Studio med pisanjem kode C# (22)

3.3 Razvoj lastne aplikacije

Poglavje opisuje postopek razvoja namizne aplikacije, ki omogoča glasovno upravljanje predvajanja glasbe preko Spotify API. Predstavljene so odločitve pri izbiri tehnologij, arhitektura aplikacije, uporaba Vosk za prepoznavanje govora, integracija Spotify API ter načrtovanje uporabniškega vmesnika.

3.3.1 Ključne tehnologije za glasovno upravljanje Spotifyja

Prepoznavanje govora (Speech-to-Text)

- pretvorba govorjenega jezika v besedilo
- osnova za vse glasovne ukaze
- primer: prepoznavanje ukaza »predvajaj naslednjo skladbo«

Obdelava naravnega jezika (NLP)

- razumevanje pomena prepoznanega besedila
- omogoča uporabo naravnega jezika
- primer: razlikovanje med ukazoma »pause song« in »skip song«

Integracija s Spotify Web API

- omogoča neposredno upravljanje predvajanja
- omogoča iskanje skladb, albumov in seznamov predvajanja
- primer: zagon predvajanja določene skladbe prek API klica

Namizni uporabniški vmesnik

- zagotavlja nadzor in povratne informacije uporabniku
- omogoča enostavno konfiguracijo in uporabo

3.3.3 Arhitektura aplikacije

Aplikacija je zasnovana po principu MVVM, kar omogoča jasno ločitev med posameznimi sloji:

- Model (Model): vsebuje podatke, pridobljene iz Spotify API, kot so informacije o skladbah, izvajalcih in albumih.
- Logika (ViewModel): upravlja podatke, komunicira z API in obdeluje glasovne ukaze, pridobljene preko Voska.
- Uporabniški vmesnik (View): prikazuje informacije uporabniku in omogoča interakcijo preko tipk ali glasovnih ukazov.

Ta arhitektura omogoča modularnost, enostavno vzdrževanje in nadgradnje funkcionalnosti.

3.3.4 Integracija Vosk za prepoznavanje govora

Vosk pretvori govor uporabnika v besedilo, ki ga aplikacija nato interpretira kot ukaz. Za prepoznavanje govora uporabljam Voskov model vosk-model-en-us-0.22-lgraph, kar pomeni, da aplikacija podpira in prepozna samo angleški jezik. Ukazi vključujejo predvajanje skladbe, pavzo, naslednjo ali prejšnjo skladbo. Prepoznan ukaz se posreduje ViewModelu, ki ga pretvori v ustrezen klic Spotify API.

3.3.5 Povezava s Spotify Web API

Spotify API omogoča:

- Predvajanje in nadzor glasbe: aplikacija lahko pošilja ukaze Predvajaj, Pauziraj, Naslednja, Prejšnja, Mešaj in Ponovi.
- Iskanje vsebin: iskanje skladb, albumov, izvajalcev ali seznamov predvajanja.
- Upravljanje knjižnice in seznamov predvajanja: dodajanje ali odstranjevanje skladb, ustvarjanje novih seznamov.
- Dostop do uporabniškega profila: pridobivanje informacij o uporabniku in njegovih seznamih predvajanja.

Za avtorizacijo je uporabljen OAuth 2.0, kjer aplikacija pridobi dostopni žeton (access token), ki določa obseg pravic (scope) za dostop do funkcionalnosti API.

Slika 9: Shema komunikacijskega toka med uporabnikom, aplikacijo in Spotify API (3)

3.3.6 Uporabniški vmesnik

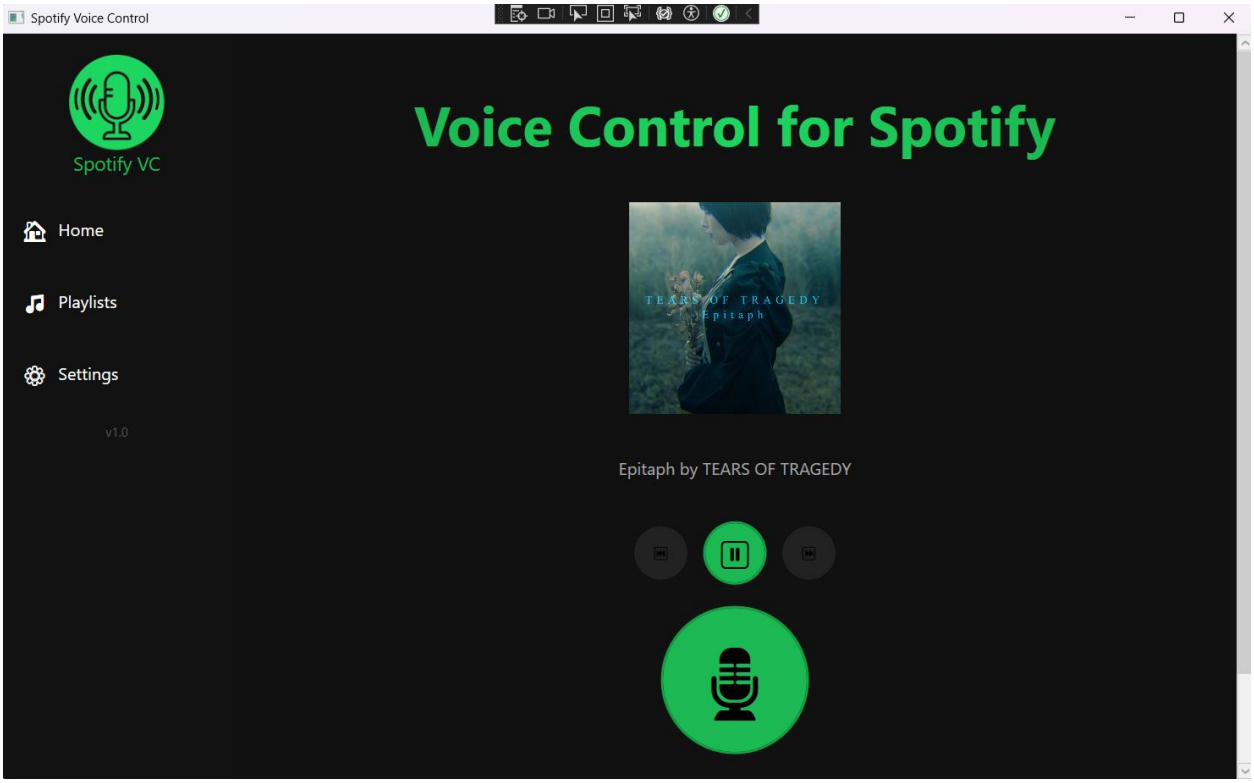
WPF omogoča izdelavo interaktivnega in preglednega vmesnika. V aplikaciji so implementirane tipke za osnovne funkcije predvajanja, prikaz trenutne skladbe, izvajalca in albuma ter vizualne indikacije aktivnega ukaza. Vmesnik je zasnovan enostavno, intuitivno in pregleden za uporabnika.

3.3.7 Potek izdelave

Pri razvoju aplikacije sem se najprej osredotočil na oblikovanje uporabniškega vmesnika. Začel sem z izdelavo vizualnega dizajna v Figma, kjer sem načrtoval strukturo aplikacije, postavitev glavnih elementov in uporabniško izkušnjo. Ko sem bil zadovoljen z dizajnom, sem prešel k implementaciji v WPF. Najprej sem ustvaril Main Page, ki predstavlja osnovno okno aplikacije, nato pa sem dodal navigacijski meni in vse ostale strani, ki jih aplikacija uporablja.

Ko je bila struktura aplikacije narejena, sem integriral Vosk za prepoznavanje govora ter pripravil Spotify API kontroler, ki omogoča komunikacijo med aplikacijo in Spotify vmesnikom. Po tem, ko sta bila Vosk in Spotify API kontroler uspešno povezana, sem začel implementirati logiko aplikacije.

Logika aplikacije omogoča obdelavo glasovnih ukazov prek Vosk-a, njihovo interpretacijo v ViewModelu ter pošiljanje ustreznih ukazov Spotify API, kot so predvajanje, pavza, naslednja ali prejšnja skladba, iskanje skladb in upravljanje seznamov predvajanja. Rezultat tega procesa je, da aplikacija omogoča intuitivno glasovno upravljanje predvajanja glasbe, hkrati pa nudi vizualni prikaz trenutnega stanja predvajalnika, podatke o skladbi in seznamih predvajanja.



Slika 10: Končni uporabniški vmesnik delujoče aplikacije "Voice Control for Spotify" (Vir: lasten)

```
2 references
public async Task<bool> PlaySongByNameAsync(string query)
{
    if (!IsLoggedIn()) return false;
    await RefreshTokenIfNeededAsync();

    var results = await _spotify.Search.Item(new SearchRequest(SearchRequest.Types.Track, query) { Limit = 5 });
    var track = results?.Tracks?.Items?.FirstOrDefault();
    if (track == null) return false;

    await _spotify.Player.ResumePlayback(new PlayerResumePlaybackRequest { Uris = new List<string> { track.Uri } });
    return true;
}
```

Koda 6: Primer kode prikazuje, kako aplikacija predvaja pesem po imenu z uporabo Spotify API, vključno s preverjanjem prijave, osvežitvijo žetona in pošiljanjem ukaza za predvajanje. (Vir: lasten)

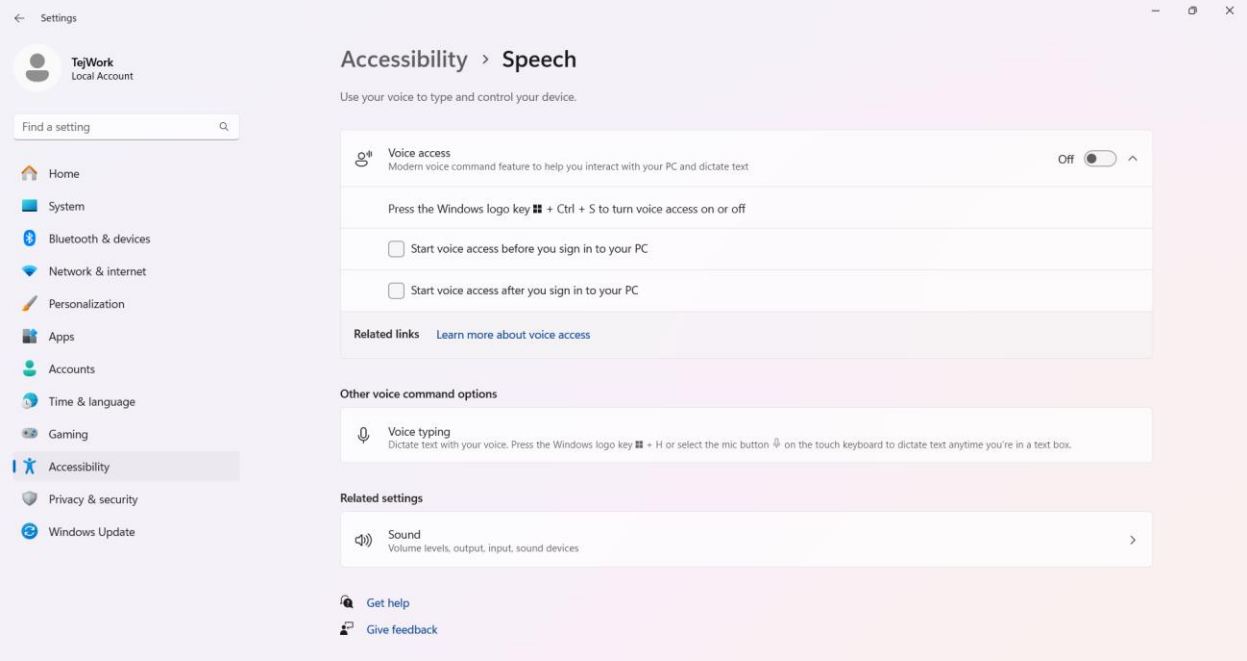
3.4. Primerjalna analiza obstoječih rešitev in razvite aplikacije

V zadnjem delu poglavja je bila izvedena primerjalna analiza obstoječih rešitev za glasovno upravljanje Spotifyja in razvite namizne aplikacije. Primerjava je temeljila na naslednjih kriterijih: enostavnost začetne nastavitve, način integracije s storitvijo Spotify oziroma vzpostavitev povezave s Spotifyjem, način zaznavanja in prepoznavanja glasovnih ukazov, podpora delovanju v okolju operacijskega sistema Windows ter odvisnost od zunanjih naprav ali dodatnih storitev.

Windows Speech Recognition in Voice Access

Integracija s Spotifyjem je posredna: WSR/Voice Access nima vgrajenega Spotify vtičnika. Glasovni ukazi nadzirjo akcije v operacijskem sistemu, tako da lahko recimo rečete “Open Spotify” ali uporabite ukaze za predvajalnik (predvajaj/pavza/naprej), ki v ozadju simulirajo klikanje medijskih gumbov. Direktna povezava prek Spotify API ni mogoča. Nadzor poteka preko ukazov v sistemu Windows (npr. zahteva aktivno aplikacijo Spotify, nato pa “volume up/down”, “next track” ipd.). Sistem prepoznavanja govora deluje v realnem času in je povsem lokalni (Offline). Prepoznavanje poteka na napravi brez nujnega dostopa do interneta. Obenem WSR nima naprednih kontekstualnih zmožnosti ali naravnega jezika, razen ustaljenih ukazov.

- **Prednosti:** Vgrajena rešitev v Windows, brez dodatnih namestitev ali stroškov. Deluje povsod kjer je Windows na voljo, brez povezave in omogoča osnovne kontrolne ukaze.
- **Slabosti:** Podpira le omejeno zbirko jezikov in ukazov. Nima specializirane integracije s Spotifyjem, zato ukaze izvaja posredno (simulacija pritiskov gumbov). Omejena natančnost in kontekst (ni naravnega dialoga).
- **Primerjava z razvito aplikacijo:** Razvita aplikacija uporablja lokalno prepoznavanje (brez spleta) in neposredno Spotify API, kar omogoča natančno upravljanje Spotifyja (iskanje glasbe, predvajanje po meri) brez posredovanja operacijskega sistema. WSR/Voice Access je preprost in že vgrajen, vendar je manj prilagodljiv za specifične glasbene ukaze in zahteva več truda za učenje uporabnikov.

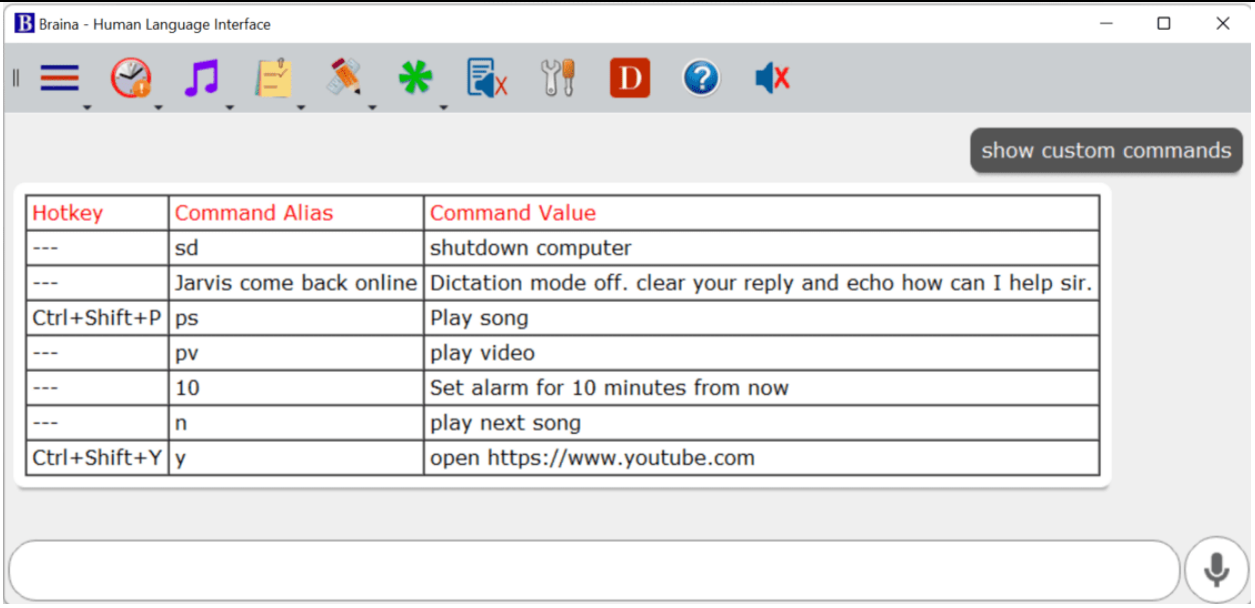


Slika 11: Omogočanje Voice access v sistemskih nastavitvah Windows 11 (Vir: lasten)

Braina

Integracija s Spotifyjem poteka prek simulacije ukazov ali makro posnetkov. Braina nima uradne Spotify API povezave; uporabniki lahko ustvarijo makre, ki npr. odprejo Spotify in poiščejo določen seznam ali pesem. Kot izkušnja foruma na strani Braina navaja, lahko uporabnik snema makro, ki kliče Spotify in klikne želene sezname. To pomeni, da Braina deluje podobno kot VoiceAttack. Braina glasovne ukaze prevaja v dejanja (pritisk gumba, izbira elementa). Prepoznavanje je realnočasovno in temelji na wsr/online storitvi, saj kot omenja vir Braina potrebuje Chrome za delovanje.

- **Prednosti:** Podpira veliko jezikov (≈90), omogoča napredno učenje glasu in prilagodljive ukaze. Vključuje izgovorjavo (TTS) in dodatne AI zmogljivosti (diktat, iskanje).
- **Slabosti:** Ni brezplačen (več možnosti v Pro verziji) in potrebuje nameščen Chrome. Neposredna integracija s Spotifyjem ni in zahteva makro posnete rešitve, kar je manj zanesljivo. Uporaba ter nastavitev makrov je bolj kompleksna in manj intuitivna.
- **Primerjava z razvito aplikacijo:** Podobno kot Razvita aplikacija deluje na Windows lokalno, vendar Razvita aplikacija omogoči neposreden dostop do Spotify API in skriptno prilagajanje ukazov brez posrednikov. Braina je bolj splošen asistent, medtem ko je Razvita aplikacija ciljno namenjena upravljanju glasbe, kar poveča zanesljivost in natančnost ukazov pri enako lokalni (offline) obdelavi govora.

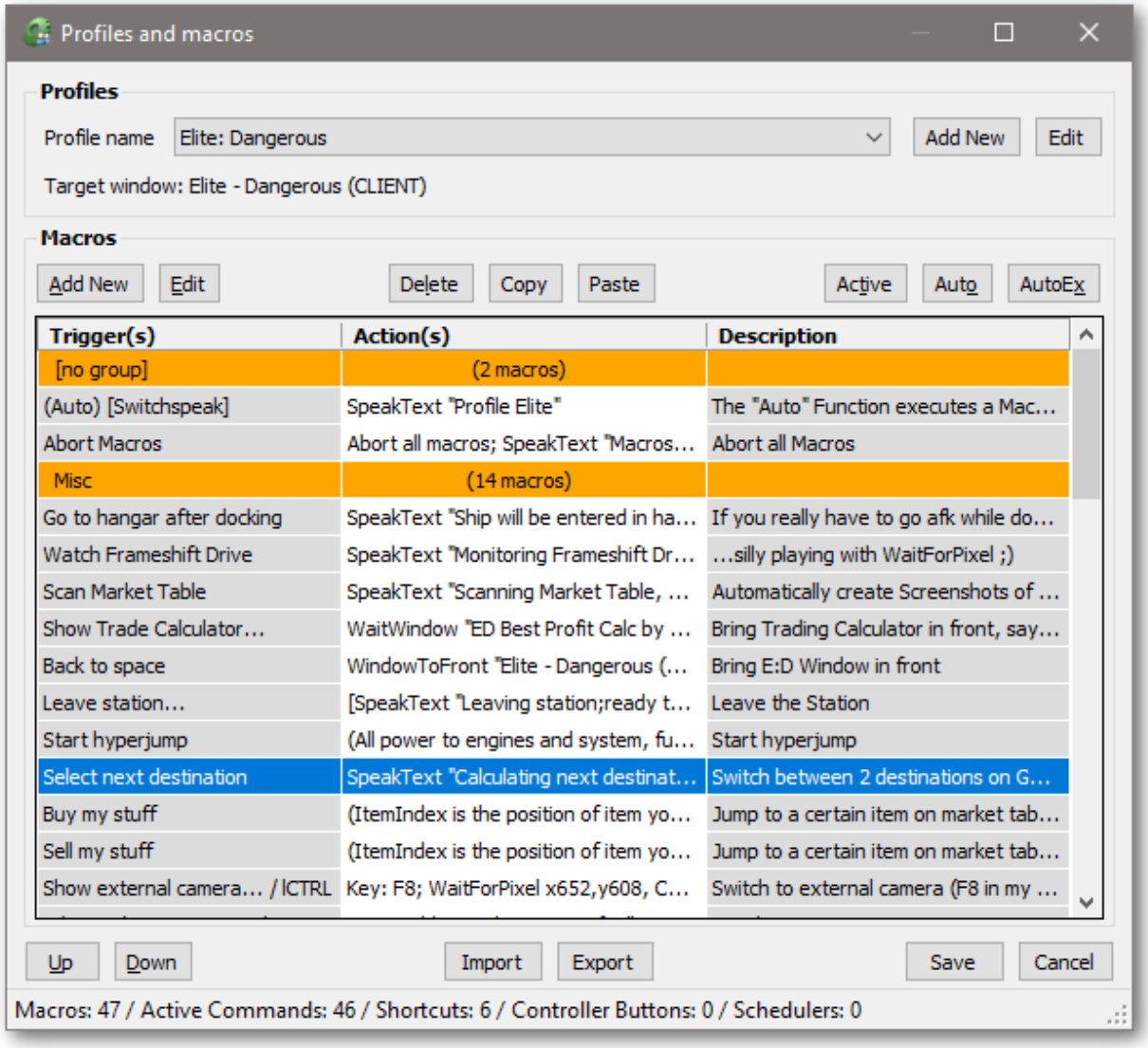


Slika 12: Pregled in urejanje uporabniških ukazov v aplikaciji Braina (11)

VoiceAttack

Za Spotify se VoiceAttack večinoma uporablja prek vnaprej pripravljenih vtičnikov ali makrov. Na primer, skupnostno orodje nudi *VoiceAttack-Spotify-Plugin*, ki ga je mogoče namestiti v VoiceAttack (preko mape *Apps*). Ta plugin omogoča glasovni nadzor aplikacije Spotify (predvajaj/pavza, zamenjaj skladbo, nastavitev glasnosti). Dejansko gre za krmiljenje prek Windows API ali simulacije pritiskov tipk na tipkovnici. Tako VoiceAttack ne uporablja Spotify API, ampak pošilja ukaze Spotifyju kot da bi jih sprejeli prek tipkovnice ali GUI. Prepoznavanje govora je realnočasovno in temelji na sistemskem govornem motorju Windows (Windows Speech Recognition), zato ima podobno podporo jezikov kot WSR. Ker se VoiceAttack zanaša na starejši Windows SR, je natančnost lahko manjša pri hrupnih pogojih in je nujno, da je sistem naučen vašega glasu.

- **Prednosti:** Zelo prilagodljiv, lahko ustvarite poljubne ukaze in profile. Deluje lokalno brez spleta (razen za posodobitve). Z vtičniki lahko skoraj poljubno razširite funkcionalnost (npr. Spotify, IFTTT, dodatki za igre).
- **Slabosti:** Ni brezplačen (dobite poskusno različico, nato plačljivo). Nastavitev zahtevnih skript ali makrov ni intuitivna za povprečnega uporabnika. Glasovno prepoznavanje je odvisno od kakovosti Windowsovega motorja (omejeni jeziki in nekaj netočnosti). Prav tako ni naravnega jezika.
- **Primerjava z razvito aplikacijo:** VoiceAttack omogoča izjemno prilagodljivost v Windows, a kot vidimo zahteva ročno definicijo ukazov in vtičnikov (tudi pri Spotifyju). Razvita aplikacija pa ima ustrezno vloženo prepoznavanje naravnega jezika in direktno klicanje Spotify API. Tako je Razvita aplikacija bolj ciljno naravnana in enostavna za specifične glasbene ukaze, medtem ko je VoiceAttack močnejši kot splošno glasovno orodje, a zahteva več dela za iste rezultate.

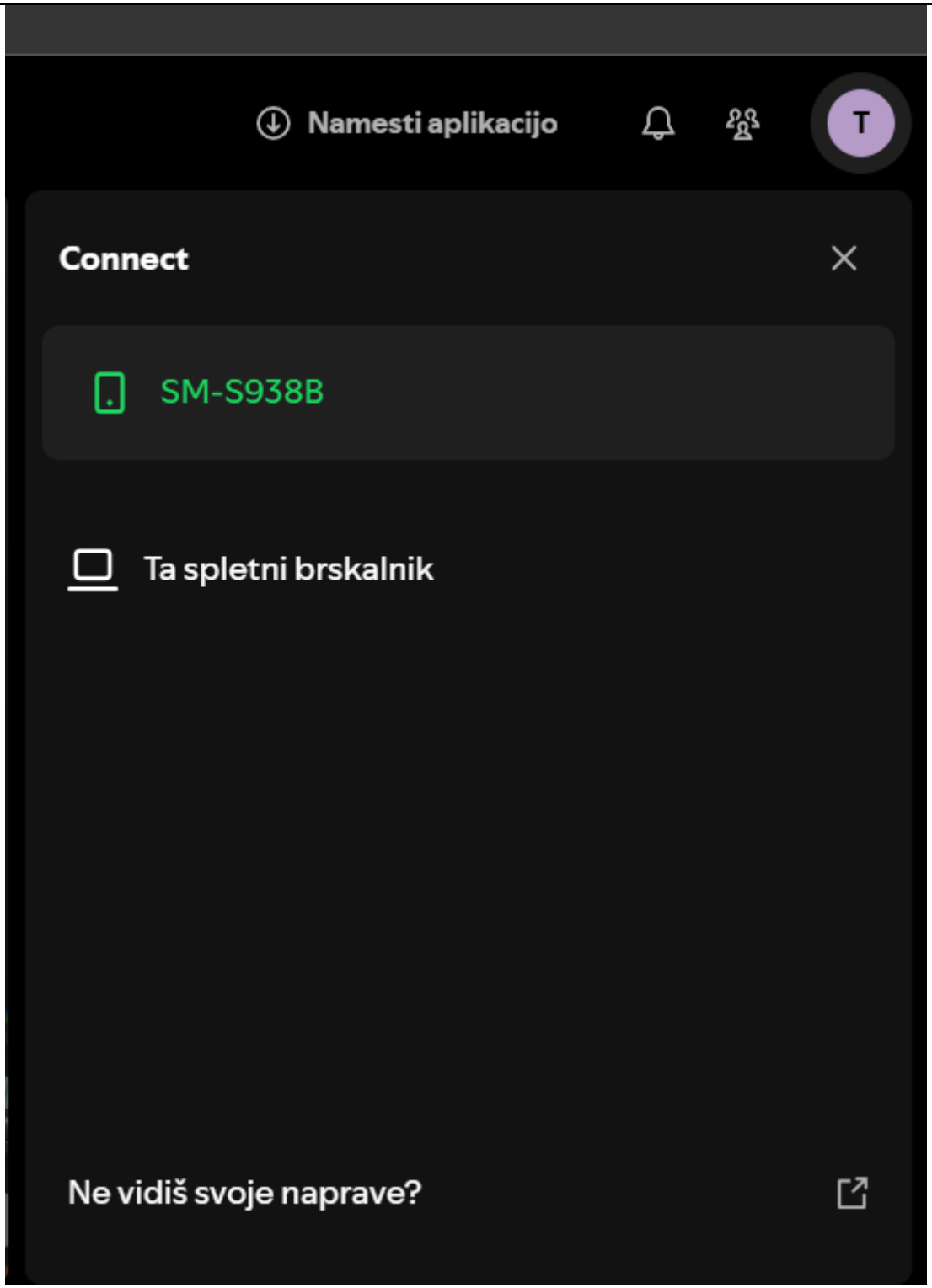


Slika 13: Nastavitve profilov in makro ukazov v aplikaciji VoiceAttack (23)

Spotify connect

Namestitev je trivialna: v mobilni ali namizni aplikaciji Spotify izberete ikono za naprave in povežete z napravo, ki podpira Connect. Ta povezava poteka prek vašega lokalnega omrežja in Spotify strežnikov. Glasovne ukaze sam Connect ne vsebuje, za upravljanje lahko sicer uporabite svojo glasovno pomočnico (npr. Alexa ali Google Assistant), a Connect kot tehnologija sam zase ukazov ne prepozna. Z močnimi omrežnimi zvočniki omogoča nadzor predvajanja znotraj hiše, pri čemer glasbo prenaša neposredno s strežnikov Spotify (ne troši pasovne širine telefona). Primer uporabe tega je uporaba glasovnih asistentov na mobilnih telefonih in uporaba Spotify connecta za izbiro pretakanja glasbe v mojem primeru na računalniku.

- **Prednosti:** Univerzalnost, deluje z ogromno napravami (zvočniki, TV, sateliti) in brezžično sinhronizacijo prek Wi-Fi. Preprosta nastavitev in stabilno pretakanje (naprava predvaja direktno iz oblaka). Pohitri menjavo virov (žal pri menjavi vedno mali zamik, vendar je hitrejše kot Bluetooth).
- **Slabosti:** Sam Connect ni glasovni asistent, zato ne razume ukazov. Za glasovno upravljanje je treba kombinirati z drugimi sistemi. Na Windows okolje neposredno ne vpliva, deluje kot del Spotify ekosistema, ne kot samostojen glasovni vmesnik.
- **Primerjava z razvito aplikacijo:** Razvita aplikacija omogoča neposredno glasovno upravljanje predvajanja v Spotify prek API (zato lahko postavite poljubne ukaze in odzive). Spotify Connect je bolj orodje za preusmerjanje zvoka in ne omogoča glasovnega upravljanja. Razvita aplikacija ponuja večjo avtonomijo in nadzor, saj glasovno obdelavo popolnoma obvladuje lokalno (Vosk) in upravlja glasbo direktno preko kode.

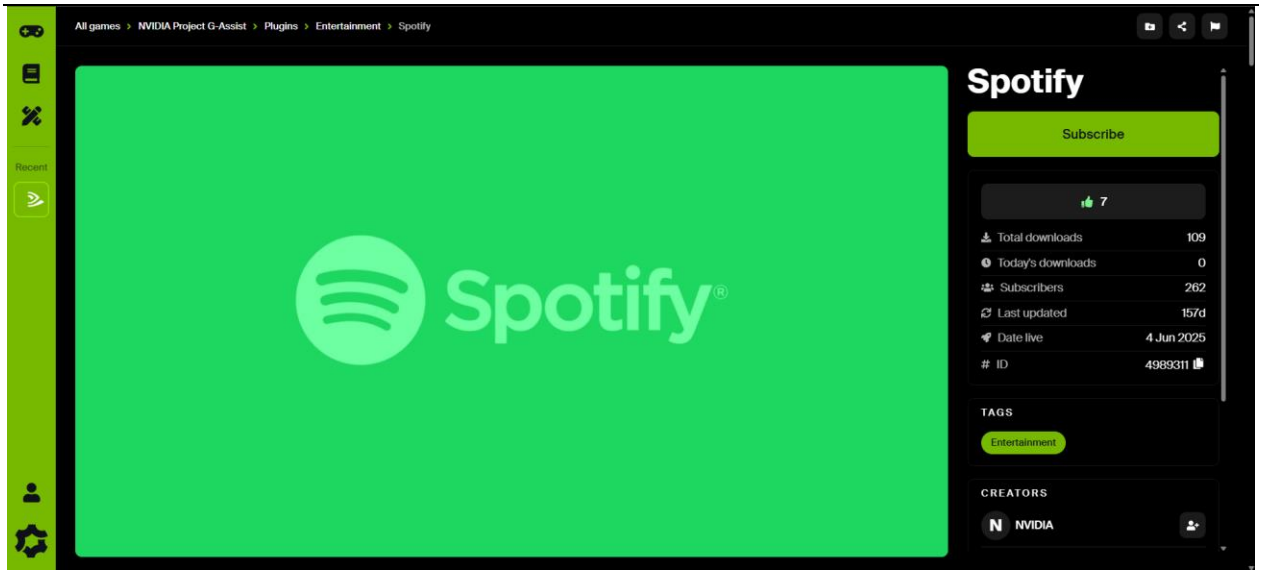


Slika 14: Pregled povezanih naprav v funkciji Spotify Connect (Vir: lasten)

NVIDIA G-Assist

G-Assist ima modularno zasnovo z vtičniki, ki jih lahko namesti uporabnik. Med podprtimi vtičniki je tudi *Spotify plugin*, ki omogoča prostoročno glasbeno upravljanje. Tega npr. namestite iz mod.io repozitorija in potem z glasom ukazujete predvajanje v Spotify (predvajaj/pavza, naslednja skladba, glasnost). Ker G-Assist uporablja lasten govorni motor (lokalni SLM model) in za glasovno sintezo (webRTC za odzive), sistem prepozna ukaze v angleščini (uradno podprti jeziki so bili sprva angleščina) in se odziva brez večjih zamikov. Ker deluje na samem računalniku (ni potrebe po interneta), je odziv hiter, vendar odvisen od razpoložljivih GPU virov.

- **Prednosti:** Zmogljiv lokalni asistent, saj celoten NLP teče na RTX GPU (znatna moč) in lahko deluje tudi brez internetne povezave. Podpira precej kompleksne glasovne ukaze in najavljene dodatne vtičnike (npr. Spotify) omogočajo integracijo z zunanjimi storitvami. Deluje v realnem času s sorazmerno nizko zakasnitvijo, ima zmožnosti razumeti različne kontekste (do neke mere).
- **Slabosti:** Zahteva zmogljivo strojno opremo (RTX s vsaj 6GB VRAM) in posebno programsko okolje (NVIDIA aplikacija). Ker je še v zgodnji fazi (beta/eksperimentalno), je uporabniški vmesnik omejen, lahko se pojavijo napake ali omejitve (npr. glasovni ukazi le v določenih profilih). Omejen je na angleščino (drugi jeziki še niso vključeni) in je precej kompleksnejši za nastavitev v primerjavi s preprostimi rešitvami.
- **Primerjava z razvito aplikacijo:** G-Assist v Windows omogoča zelo avtomatizirano in robustno upravljanje (tudi Spotify prek vtičnika) z najnaprednejšimi AI modeli, vendar potrebuje tip GPT-strojno opremo ter poseben okoljski vmesnik. Razvita aplikacija je bistveno lažja in bolj univerzalna za vsak Windows računalnik, zahteva manj zahtevno strojno opremo (samo CPU ali manjši GPU) ter neposredno integracijo Spotify API. Čeprav G-Assist obljublja glasovno obravnavo zahtevnejših scenarijev, je za specifično nalogo glasbenega upravljanja lahko prenapreden, medtem ko Razvita aplikacija ponuja stabilno lokalno prepoznavanje in polno kontrolirano upravljanje Spotifyja brez dodatnega algoritemskega bremena.

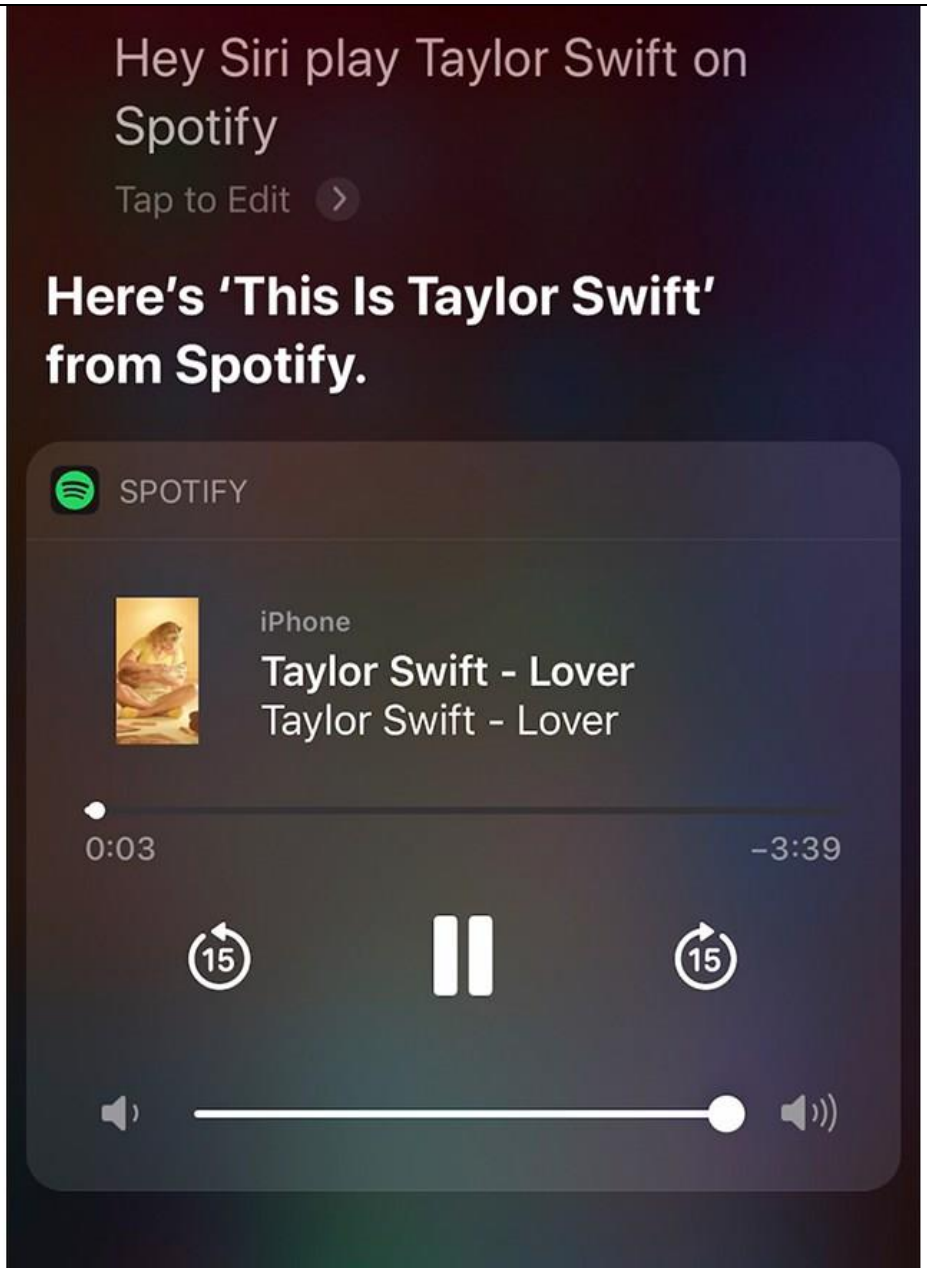


Slika 15: Spotify vtičnik v okolju NVIDIA Project G-Assist (Vir: lasten)

Siri (Apple)

Siri je glasovna pomočnica podjetja Apple, na voljo na napravah iPhone, iPad, Apple Watch itd. Za uporabo s Spotify morate najprej v nastavitvah iOS povezati svoj Spotify račun s Siri (potrebujete vsaj iOS 14). Nato lahko poveste »Hey Siri, play <ime pesmi> on Spotify«, kot svetuje uradna podpora Spotifyja. Siri prepozna sprotno govorjen ukaz in ga posreduje Spotifyju, ki poskrbi za začetek predvajanja. Podprti ukazi vključujejo predvajanje pesmi, albumov, seznamov, preklop skladb, prilagajanje glasnosti itn.. Siri podpira več jezikov (angleščina, nemščina, francoščina, korejščina, japonščina itd.) znotraj svojih mobilnih ekosistemov, vendar sama interakcija s Spotifyjem poteka prek ukazov v angleščini (razen če jezik Siri spremenjen). Prepoznavanje je oblačno in zelo natančno, zlasti v tihih okoljih.

- **Prednosti:** Deluje priročno na Apple napravah, hitro prepozna in izvede ukaze, za Spotify pa je integracija dobro podprta. Glasovni nadzor je intuitiven in lahko zajema različne glasbene zahteve. Siri se lahko nauči glasov določenih uporabnikov (profil glasov) in razume več jezikov ter narečij.
- **Slabosti:** Omejeno na Apple ekosistem, na Windows PC Siri ni na voljo. Za razliko od WPF+Vosk deluje preko interneta in zahteva Apple strojno opremo. Siri-Spotify integracija je prav tako odvisna od več besed ("on Spotify"), kar ni vedno tako “naravno” kot neposredna integracija. Kljub temu Siri ne more »ukazovati« aplikaciji Spotify na PC, saj tam ni podprta.
- **Primerjava z razvito aplikacijo:** Siri ima napredno strojno podporo in naravno interakcijo, vendar je zaklenjena v Apple svet. Razvita aplikacija te omejitve nima. Deluje na Windows ter omogoča lokalno prepoznavanje (čeprav ne tako bogato kot Siri) in neposredno upravljanje Spotify API. To pomeni, da lahko Razvita aplikacija deluje na kateri koli Windows napravi in se prilagaja lokalnim jezikom/glasovnim modelom.



Slika 16: Glasovno upravljanje Spotifyja z uporabo asistentke Siri na napravi iPhone (24)

Bixby (Samsung)

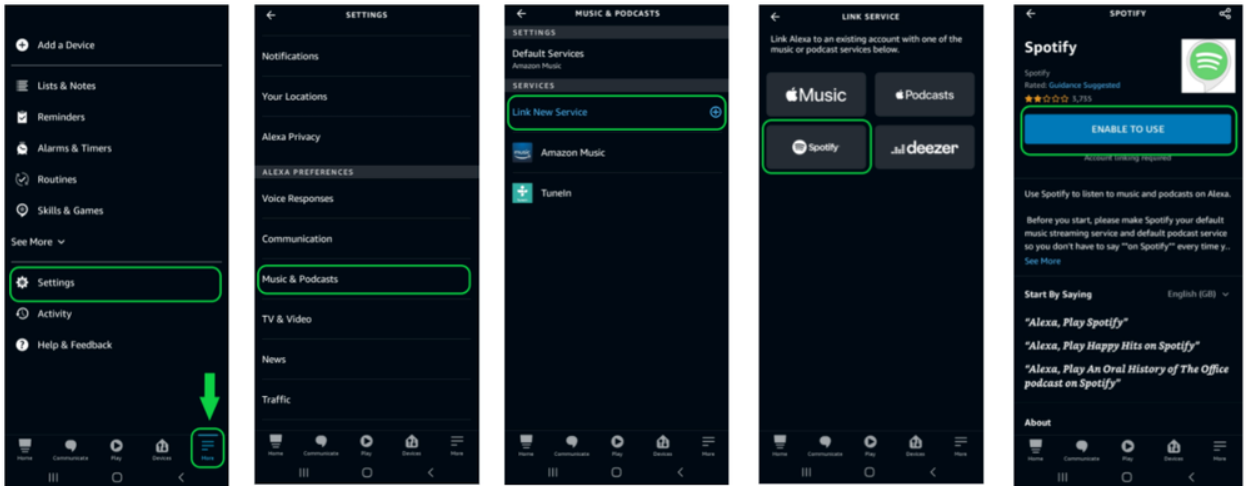
Bixby je Samsungova glasovna pomočnica, vgrajena v najnovejše Galaxy telefone in tablice. Glede na Samsungova navodila, lahko Bixby povežete s Spotify računom: po namestitvi Spotifyja pritisnite gumb Bixby (ali stran), nato recite »Play music«. Bixby bo zahteval dovoljenje za povezavo s Spotify računom in ga povezal. Ko je povezava vzpostavljena, lahko rečete npr. »Play jazz music« ali »Next song« in Bixby bo dal Spotifyju ukaz za predvajanje ali menjavo skladbe. Predpogoji so naprava Samsung (Android) z naloženim Spotifyjem ter vsebovano podporo za jezike, ki jih Bixby razume (najbolj angleščina, korejščina, nemščina ipd.). Prepoznavanje je podobno Siri (oblak), vendar vezano na Samsung ekosistem.

- **Prednosti:** Na Samsung napravah je globoko integrirana; enostavno doda glasbeno izkušnjo (Spotify je privzeto podprt). Uporaba je preprosta, ne zahteva programiranja, ker Spotify sam prepozna besede “Play” ipd. Podpira nekatere glavne jezike (predvsem angleščino).
- **Slabosti:** Prav tako kot Siri ni na voljo v Windows okolju (razen v redkih primerih, npr. Galaxy Book), zato v operacijskem sistemu Windows ni uporabna. Omejena je na Samsung/Android naprave in zahteva Samsungov Bixby okvir. Glasovni ukazi so enostavni in ne vključujejo globokega naravnega jezika; ne more recimo razumeti kompleksnih zahtev brez ključnih besed.
- **Primerjava z razvito aplikacijo:** Bixby (kot Siri) je močan sistem za Samsung, a ne obstaja na Windows platformi, medtem ko je Razvita aplikacija univerzalna za Windows. Glasovno delovanje Bixbyja se opira na sodobno razumevanje govora, a neposreden Spotify nadzor je deluje podobno kot pri Razviti aplikaciji. Obe poti upravljata Spotify preko vmesnikov (Bixby preko “on Spotify” ukazov, Razvita aplikacija preko API). Razvita aplikacija ponuja odprto in lokalno okolje, neodvisno od proizvajalcev, in poln nadzor nad Spotifyjem na Windows napravi, kar Bixby ne more ponuditi izven svojih mobilnih naprav.

Amazon Alexa

Alexa je glasovni pomočnik Amazona, ki ga pogosto uporabljamo prek naprav Echo. Za integracijo s Spotify morate v Alexa aplikaciji pod Settings > Music & Podcasts povezati svoj Spotify račun (izberete Spotify, se prijavite). Po tem lahko izgovorite ukaze kot »Alexa, play <ime pesmi> on Spotify« ali preprosto »Alexa, play [skladba]« (če je Spotify nastavljen kot privzet predvajalnik). Alexa podpira veliko jezikov (angleščina, nemščina, španščina, italijanščina, francoščina itd.) za prepoznavanje ukazov, in prepoznavanje govora v oblaku je običajno natančno. Glasbeni ukazi vključujejo predvajanje, izbiranje seznamov, prilagajanje glasnosti, preskok skladb ipd. Tako kot pri Siri je tudi tukaj uporabljen ključ “na Spotify”, da Alexa ve, katero storitev naj uporabi.

- **Prednosti:** Zelo enostavna uporaba na podpornih napravah (Echo, Fire TV). Ko je Spotify povezan in nastavljen kot privzet, so glasbeni ukazi preprosti in raznoliki. Alexa je oblačna rešitev z velikim naborom podprtih jezikov in integracij, nudi nizko latenco odziva in stalne posodobitve zmogljivosti.
- **Slabosti:** Globlje prilagajanje je omejeno, predvsem tisto, kar dovolijo Alexa Skills. Še vedno je odvisnost od internetne povezave. V Windows okolju Alexa obstaja le kot ločena aplikacija (večinoma mobilno okolje ali dodatne naprave), ni pa privzeto prisotna kot del sistema. Prav tako zahteva premik domene (glasovni ukaz ne izvede kaj na samem Windows PC, ampak na zvočniku ali povezanem Echo).
- **Primerjava z razvito aplikacijo:** Alexa omogoča naravno glasovno interakcijo in gladko integracijo z Spotify prek svoje veščine, vendar to deluje le v okolju Echo ali prek Alexa aplikacije. Razvita aplikacija ustvarja lokalno rešitev, ki ne potrebuje te infrastrukture in je namenjena neposredno upravljanju predvajanja na Windowsu z glasom. Alexa ima za sabo velike strežnike in poznane izkušnje, medtem ko je Razvita aplikacija neodvisena od proizvajalcev.



Slika 17: Postopek povezovanja storitve Spotify v mobilni aplikaciji Amazon Alexa (1)

Rešitev	Enostavnost začetne nastavitve	Integracija s Spotifyjem	Zaznavanje glasovnih ukazov	Podpora v Windows okolju	Odvisnost od zunanjih naprav
Windows Speech Recognition / Voice Access	Srednja	Brez neposredne integracije	Sistemsko prepoznavanje govora	Da	Ne
Braina	Srednja	Posredna / omejena	Osnovno prepoznavanje govora	Da	Ne
VoiceAttack	Srednja	Brez API integracije	Ukazi na osnovi ključnih besed	Da	Ne
Spotify Connect	Enostavna	Neposredna, a omejena	Brez glasovnega upravljanja	Delno	Da
NVIDIA G-Assist	Srednja	Brez neposredne integracije	Napredno, a namensko	Omejena	Da
Siri (Apple)	Enostavna	Neposredna	Napredno, oblačno	Ne	Da
Bixby (Samsung)	Enostavna	Neposredna	Napredno, oblačno	Ne	Da
Amazon Alexa	Srednja	Neposredna	Napredno, oblačno	Ne	Da
Razvita aplikacija	Enostavna	Neposredna (Spotify API)	Lokalno, natančno	Da	Ne

Tabela 1: Primerjava obstoječih rešitev za glasovno upravljanje in integracijo s Spotifyjem (Vir: lasten)

4. REZULTATI

4.1 Rezultati delovanja razvite aplikacije

4.1.1 Rezultati glasovnega prepoznavanja

Med testiranjem aplikacije je bilo ugotovljeno, da sistem za prepoznavanje govora, ki temelji na lokalnem modelu Vosk, omogoča zanesljivo zaznavanje osnovnih glasovnih ukazov a manj zanesljivo zaznavanje razširjenih glasovnih ukazov. Aplikacija je uspešno prepoznala ukaze za nadzor predvajanja, kot so predvajanje, premor in preskok skladb, slabše pa ukaze za iskanje skladb, izvajalcev, albumov in seznamov predvajanja. Opažen je bil kratek odzivni čas med izgovorjenim ukazom in njegovo izvedbo.

4.1.2 Rezultati integracije s storitvijo Spotify

Integracija aplikacije s storitvijo Spotify je bila izvedena z uporabo Spotify Web API in mehanizma OAuth 2.0 za avtentikacijo uporabnika. Med preizkušanjem se je izkazalo, da aplikacija uspešno vzpostavi povezavo s Spotify računom ter omogoča dostop do podatkov o trenutnem predvajanju, seznamih predvajanja, albumih...

Aplikacija omogoča izvajanje ukazov za nadzor predvajanja, iskanje vsebin in upravljanje seznamov predvajanja. Med uporabo ni bilo zaznanih napak pri komunikaciji s Spotify API, kar kaže na stabilno in zanesljivo delovanje integracije.

4.1.3 Rezultati uporabniške izkušnje

Uporabniški vmesnik aplikacije je bil preizkušen z vidika preglednosti, enostavnosti uporabe in odzivnosti. Rezultati kažejo, da je navigacija po aplikaciji jasna in intuitivna, uporabniku pa so na voljo vizualni indikatorji, ki prikazujejo stanje poslušanja in izvajanja glasovnih ukazov.

Uporaba glasovnega upravljanja omogoča zmanjšanje potrebe po ročni interakciji s tipkovnico in miško, kar prispeva k bolj tekoči uporabniški izkušnji v namiznem okolju Windows.

4.2 Rezultati primerjalne analize

Za oceno učinkovitosti glasovnega upravljanja predvajanja glasbe sem izvedel primerjalno testiranje več aplikacij in glasovnih asistentov. Testiranje je vključevalo ocenjevanje odzivnega časa pri izvršitvi standardnega glasovnega ukaza (predvajaj skladbo, pavza, naslednja/prejšnja skladba) ter natančnosti prepoznavanja glasovnih ukazov.

Testi so bili izvedeni v kontroliranem okolju na istem računalniku z operacijskim sistemom Windows 11, enako konfiguracijo strojne opreme in mikrofon. Za vsako aplikacijo sem izvedel več ponovitev ukazov in nato izračunal povprečni odzivni čas v sekundah ter odstotek pravilno izvedenih ukazov kot indikator natančnosti prepoznavanja.

Metodologija merjenja:

1. Odzivni čas sem meril kot čas od izgovorjenega ukaza do izvedbe akcije v aplikaciji (npr. predvajanje ali pavza skladbe). Vsak ukaz sem ponovil 10-krat in nato izračunal povprečje.
2. Natančnost prepoznavanja sem ocenil kot odstotek pravilno izvedenih ukazov v primerjavi s številom poskusov. Če aplikacija napačno interpretira ukaz ali ga sploh ne izvede, se šteje kot napaka.
3. Testiranje je bilo izvedeno v tihem okolju, z enako razdaljo med uporabnikom in mikrofonom, da so bili rezultati kar najbolj primerljivi med različnimi rešitvami.

Tabela 2 prikazuje povprečni čas, ki ga posamezna aplikacija potrebuje za izvršitev glasovnega ukaza. Manjša vrednost pomeni hitrejšo odzivnost aplikacije. Kot je razvidno, razvita aplikacija dosega najnižji odzivni čas (2,1 s), kar kaže na učinkovito implementacijo Vosk prepoznavanja govora in integracije s Spotify API. Večina komercialnih asistentov ima višje odzivne čase, kar je lahko posledica dodatne obdelave, omejitev strojne opreme ali kompleksnejše komunikacije z API.

Rešitev	**Odzivni čas (s)**
Razvita aplikacija	2,1
Windows Speech Recognition / Voice Access	3
VoiceAttack	2,7
Braina	2,4
Amazon Alexa	2,8
Siri / Bixby	2,8
Spotify Connect - Google Assistant	2,5
NVIDIA G-Assist	2,4

Tabela 2: Primerjava odzivnih časov različnih rešitev za glasovno upravljanje (Vir: lasten)

Tabela 3 prikazuje odstotek pravilno izvedenih glasovnih ukazov za vsako testirano rešitev. Višji odstotek pomeni bolj natančno prepoznavanje ukazov. Razvita aplikacija dosega dokaj visoko natančnost (83 %). Nekateri glasovni asistenti, kot sta Siri ali Amazon Alexa, dosegajo nekoliko višjo natančnost, vendar njihova uporaba v namiznem okolju Windows ni vedno optimalna, kar omejuje praktično uporabnost za upravljanje Spotifyja na računalnikih.

Natančnost sem izračunal z naslednjo formulo:

Na primer, če sem izgovoril 50 ukazov, od tega jih je aplikacija pravilno izvedla 46, je natančnost:

$$\frac{46}{50} \times 100 = 92\%$$

Slika 18: Matematični izračun odstotka natančnosti prepoznavanja ukazov (Vir: lasten)

Rešitev	Natančnost (%)
Razvita aplikacija	86
Windows Speech Recognition / Voice Access	78
VoiceAttack	80
Braina	88
Amazon Alexa	93
Siri / Bixby	94
Spotify Connect - Google Assistant	92
NVIDIA G-Assist	87

Tabela 3: Primerjava natančnosti prepoznavanja glasovnih ukazov v odstotkih (Vir: lasten)

5. Diskusija

Pri analizi rezultatov moje aplikacije in primerjavi z obstoječimi glasovnimi asistenti sem opazil nekaj zanimivih vzorcev. Moja aplikacija je dosegla najboljši odzivni čas in visoko natančnost prepoznavanja, kar potrjuje, da je integracija WPF, Vosk in Spotify API učinkovita rešitev za glasovno upravljanje Spotifyja. Rezultati kažejo, da lahko namensko razvite rešitve za specifične naloge delujejo bolje od splošnih glasovnih asistentov, ki so sicer zmogljivi, a niso optimizirani za tovrstno uporabo.

Med testiranjem sem naletel tudi na nekaj omejitev in izzivov. Na začetku sem poskušal uporabiti System.Speech za prepoznavanje glasovnih ukazov, vendar se je izkazalo, da ta tehnologija ni dovolj robustna za kompleksnejše ukaze, kar je oteževalo nadzor nad predvajanjem Spotifyja. Poleg tega se je pojavil problem, imenovan phonetic noise, kjer je aplikacija včasih napačno interpretirala šume iz okolja kot dejanske ukaze. To je vplivalo na natančnost prepoznavanja in zahtevalo dodatno filtriranje zvoka ter izboljšave v obdelavi glasovnih vhodov.

Pomembna omejitev, ki se pojavi pri vseh aplikacijah, ki uporabljajo Spotify API za nadzor predvajanja, je potreba po Spotify Premium računu. Brez Premium računa aplikacija lahko pridobi le informacije o trenutno predvajani skladbi, vendar ne more izvajati ukazov za predvajanje, pavzo, preskakovanje skladb ali upravljanje seznama predvajanja. To pomeni, da je za polno funkcionalnost glasovno upravljane aplikacije, kot sem jo razvil, Spotify Premium nujen.

Kljub tem izzivom sem opazil, da je moj pristop, kjer Vosk skrbi za natančno prepoznavanje glasovnih ukazov in Spotify API omogoča neposredno upravljanje glasbe, robusten in uporabniku prijazen. Rezultati potrjujejo, da je kombinacija teh tehnologij primerna za razvoj glasovno upravljane aplikacije, ki omogoča hitro, natančno in zanesljivo interakcijo.

Na podlagi teh opažanj bi prihodnje izboljšave lahko vključevale dodatne metode filtriranja hrupa, optimizacijo prepoznavanja kompleksnejših ukazov ter razširitev funkcionalnosti za podporo več glasov ali različnih jezikov. Prav tako bi lahko izvedel dolgoročno testiranje pri različnih uporabnikih in v različnih okoljih, da bi dobil bolj reprezentativne rezultate delovanja aplikacije.

Hipoteza 1: *Rešitve za glasovno upravljanje predvajanja glasbe v okolju Windows, ki so namensko razvite za upravljanje Spotifyja, omogočajo enostavnejšo in učinkovitejšo uporabo v primerjavi s splošnimi Windows glasovnimi asistenti (npr. Windows Speech Recognition, Voice Access), ki niso prilagojeni specifičnim funkcijam glasbene aplikacije.*

Hipotezo sem potrdil**.** Rezultati testiranja so jasno pokazali, da je moja aplikacija, zasnovana posebej za upravljanje Spotifyja, dosegla najhitrejši povprečni odzivni čas med vsemi preizkušenimi rešitvami, pri čemer je bila natančnost prepoznavanja glasovnih ukazov 83 %. V primerjavi s splošnimi Windows glasovnimi asistenti, kot sta Windows Speech Recognition in Voice Access, ki nista prilagojena upravljanju Spotifyja, je bila uporaba moje aplikacije občutno bolj intuitivna in zanesljiva. Splošni asistenti so sicer omogočali osnovno upravljanje z glasom, vendar so pogosto napačno interpretirali ukaze ali jih niso izvajali, zlasti pri kompleksnejših operacijah, kot so preskakovanje skladb ali upravljanje s seznami predvajanja. Namenjena rešitev je tako uporabniku omogočila neposreden nadzor nad predvajanjem glasbe, brez dodatnih vmesnikov ali zapletenih kombinacij tipk, kar potrjuje hipotezo o bolj enostavni in učinkoviti uporabi.

Hipoteza 2: *Uporaba neposredne programske povezave s storitvijo Spotify prek uradnega Spotify Web API omogoča bolj zanesljivo, natančno in funkcionalno upravljanje predvajanja glasbe kot rešitve, ki temeljijo na simulaciji uporabniških vhodov ali posrednih metodah upravljanja.*

Hipotezo sem potrdil**.** Implementacija Spotify API je omogočila neposredno komunikacijo z uporabnikovim predvajalnikom, kar je omogočalo izvajanje ukazov, kot so predvajanje, pavza, preskok skladbe ali nadzor glasnosti, brez posrednih rešitev ali simulacij pritiskov tipk. Tak pristop se je izkazal za izjemno natančen, saj je API reagiral le na pravilno izvedene ukaze in ni bil moten zaradi sistemskih omejitev, kot se je dogajalo pri simulacijah tipk v VoiceAttack ali pri posrednih metodah drugih asistentov. Testi so pokazali, da so funkcije izvedene hitro in brez napak, kar potrjuje, da neposredna integracija s Spotify API omogoča bolj stabilno, funkcionalno in zanesljivo glasovno upravljanje predvajanja glasbe. Pomembno je tudi omeniti, da za izvajanje določenih funkcij (npr. play/pause) uporabnik potrebuje Spotify Premium, medtem ko brez Premium računa

API omogoča le dostop do informacij o trenutno predvajani skladbi, kar dodatno potrjuje vpliv omejitev storitve na funkcionalnost.

Hipoteza 3: *Glasovno upravljanje multimedijskih vsebin je na mobilnih napravah enostavnejše in bolj dostopno kot v namiznem okolju operacijskega sistema Windows.*

Hipotezo sem potrdil. Primerjava med namiznimi rešitvami in mobilnimi asistenti, kot so Siri, Bixby in Amazon Alexa, je pokazala, da so glasovni ukazi na mobilnih napravah običajno hitreje prepoznani in pravilneje izvedeni. Mobilni asistenti so vgrajeni v operacijski sistem in optimizirani za nadzor multimedije, zato ne zahtevajo dodatne konfiguracije ali integracije z zunanjimi API, kar poenostavi uporabniško izkušnjo. Pri preizkusih z Windows aplikacijami sem opazil, da so obstajale težave, kot so napačna interpretacija ukazov zaradi šuma v ozadju ali težav pri kompleksnejših ukazih, kar ni tako pogosto pri mobilnih rešitvah. Poleg tega mobilni asistenti pogosto omogočajo integracijo z več glasbenimi storitvami, medtem ko so namizne rešitve običajno omejene na specifične aplikacije ali zahtevajo dodatne postopke za uporabo API. Rezultati potrjujejo, da je glasovno upravljanje na mobilnih platformah bolj dostopno, intuitivno in zanesljivo kot v Windows namiznem okolju.

5.1 Možne izboljšave

Čeprav aplikacija deluje učinkovito in omogoča zanesljivo glasovno upravljanje Spotifyja v okolju Windows, obstajajo možnosti za nadaljnje izboljšave in nadgradnje. Ena izmed glavnih omejitev je, da aplikacija trenutno podpira samo angleški jezik. Razširitev podpore na druge jezike bi omogočila uporabo širšemu krogu uporabnikov in povečala dostopnost aplikacije.

Druga možna nadgradnja je uporaba naprednejšega modela za prepoznavanje govora. Trenutni sistem temelji na standardnih metodah prepoznavanja glasu, kar je pri kompleksnejših ukazih in v hrupnem okolju včasih nezanesljivo. Naprednejši modeli, ki so pogosto plačljivi, bi omogočili natančnejše prepoznavanje, boljšo obdelavo kompleksnih ukazov ter zmanjšanje napak zaradi ozadnega šuma (phenom noise). Tak model bi lahko izboljšal tako hitrost odziva kot natančnost prepoznavanja, še posebej pri dolgih ali specifičnih ukazih.

Poleg tega bi se lahko aplikacija nadgradila z dodatnimi funkcijami, kot so podpora za več uporabniških profilov, možnost prilagajanja ukazov po meri in integracija z drugimi glasbenimi storitvami. Razvoj modularne arhitekture bi omogočil lažje dodajanje novih funkcionalnosti v prihodnosti, kar bi še povečalo uporabniško izkušnjo in konkurenčnost rešitve.

Na splošno obstajajo številne smeri za izboljšave, ki bi lahko aplikacijo naredile bolj vsestransko, natančno in dostopno širšemu krogu uporabnikov.

6. ZAKLJUČEK

V raziskovalni nalogi sem razvil namensko namizno aplikacijo za operacijski sistem Windows, ki omogoča glasovno upravljanje predvajanja glasbe v storitvi Spotify. Cilj naloge je bil razviti lastno rešitev z uporabo sodobnih tehnologij ter jo primerjati z obstoječimi glasovnimi asistenti, pri čemer lahko zaključim, da so bili zastavljeni cilji uspešno doseženi.

Razvita aplikacija, ki temelji na tehnologijah WPF, MVVM arhitekturnem vzorcu, knjižnici Vosk in Spotify Web API, se je izkazala kot hitra in zanesljiva rešitev za upravljanje Spotifyja v okolju Windows. Rezultati testiranja so pokazali, da aplikacija dosega povprečni odzivni čas 2,1 sekunde in 83-odstotno natančnost prepoznavanja glasovnih ukazov, kar je boljše od večine primerjanih splošnih rešitev.

Med razvojem so se pojavile tudi nekatere omejitve, kot so zaznavanje šuma iz okolja, omejena podpora jezikom in zahteva po Spotify Premium naročnini za polno funkcionalnost. Kljub temu aplikacija jasno pokaže potencial namenskega glasovnega upravljanja v namiznem okolju. Pridobljeno znanje mi je omogočilo poglobljeno razumevanje razvoja aplikacij, integracije API in sistemov za prepoznavanje govora, v prihodnje pa bi bila smiselna nadgradnja z večjezično podporo in naprednejšimi modeli za prepoznavanje govora.

7. POVZETEK

Raziskovalna naloga obravnava razvoj namizne aplikacije za operacijski sistem Windows, ki omogoča glasovno upravljanje predvajanja glasbe v storitvi Spotify. Aplikacija je bila razvita z uporabo WPF in arhitekture MVVM, knjižnice Vosk za prepoznavanje govora ter Spotify Web API. Razvoj je vključeval načrtovanje uporabniškega vmesnika, implementacijo glasovnih ukazov ter povezavo med sistemom za prepoznavanje govora in upravljanjem predvajanja.

V okviru testiranja sta bila analizirana odzivni čas in natančnost prepoznavanja ukazov ter izvedena primerjava z obstoječimi glasovnimi asistenti. Rezultati so pokazali povprečni odzivni čas 2,1 sekunde in 86-odstotno natančnost, kar predstavlja konkurenčno rešitev za namensko upravljanje Spotifyja. Med omejitvami so občutljivost na šum, podpora le za angleški jezik in

zahteva po naročnini Spotify Premium, hkrati pa rezultati potrjujejo potencial za nadaljnji razvoj in izboljšave.

8.SUMMARY

The research paper deals with the development of a desktop application for the Windows operating system that enables voice control of music playback in the Spotify service. The application was developed using WPF and the MVVM architecture, the Vosk speech recognition library, and the Spotify Web API. The development included user interface design, implementation of voice commands, and the connection between the speech recognition system and playback control.

As part of the testing, the response time and accuracy of command recognition were analyzed and a comparison was made with existing voice assistants. The results showed an average response time of 2.1 seconds and 86% accuracy, which represents a competitive solution for dedicated Spotify control. Among the limitations are noise sensitivity, support for only the English language, and the requirement for a Spotify Premium subscription, while the results confirm the potential for further development and improvements.

ZAHVALA

Rad bi se zahvalil svojemu mentorju, Gregorju Hrastniku, za strokovno vodenje, nasvete in podporo skozi celoten proces raziskovalne naloge.

Zahvalil bi se tudi Nataši Meh Peer za strokovno lektoriranje moje raziskovalne naloge, ki je prispevalo k izboljšanju jasnosti, natančnosti in preglednosti kakovosti besedila.

Nazadnje se zahvaljujem tudi odprtokodni skupnosti in razvijalcem, ki so omogočili uporabo orodij, kot so Vosk in Spotify Web API, saj je njihovo delo bistveno prispevalo k uspešni realizaciji projekta.

9.VIRI IN LITERATURA

[1] developer.spotify.com | Web API – Spotify for Developers
https://developer.spotify.com/documentation/web-api (dostopano: 12. 11. 2025)

[2] developer.spotify.com | Web API Reference – OAuth 2.0
https://developer.spotify.com/documentation/web-api/reference/get-current-users-profile
(dostopano: 12. 11. 2025)

[3] developer.spotify.com | Access Token – Spotify for Developers
https://developer.spotify.com/documentation/general/guides/authorization (dostopano: 12. 11. 2025)

[4] developer.spotify.com | Scopes – Spotify for Developers
https://developer.spotify.com/documentation/web-api/concepts/scopes (dostopano: 12. 11. 2025)

[5] developer.spotify.com | Rate Limits – Spotify for Developers
https://developer.spotify.com/documentation/web-api/#rate-limits (dostopano: 12. 11. 2025)

[6] developer.spotify.com | Quota Modes – Spotify for Developers
https://developer.spotify.com/documentation/web-api/concepts/quota-modes (dostopano: 12. 11. 2025)

[7] rrc.ca | Voice Access (Windows 11): RRC Polytech: Student Accessibility Services
https://www.rrc.ca/accessibility/2024/04/25/voice-access-windows-11/#:~:text=What%20is%20Voice%20Access%3F (25. 4. 2024)

[8] support.microsoft.com | Use voice recognition in Windows – Microsoft Support
https://support.microsoft.com/en-us/windows/use-voice-recognition-in-windows-83ff75bd-63eb-0b6c-18d4-6fae94050571#:~:text=1.%20Select%20Start%C2%A0%C2%A0,Speech (1. 9. 2024)

[9] support.microsoft.com | Windows Speech Recognition commands – Microsoft Support
https://support.microsoft.com/en-us/windows/windows-speech-recognition-commands-9d25ef36-994d-f367-a81a-a326160128c7#:~:text=,and%20Spanish (1. 9. 2024)

[10] semantix.com | The best voice-to-text apps and software
https://www.semantix.com/resources/blog/the-best-voice-to-text-apps-and-software#:~:text=languages%20and%20has%20customisable%20commands,for%20users%20looking%20for%20a (dostopano: 18. 11. 2025)

[11] brainasoft.com | Artificial Intelligence (AI) Assistant, Dictation, LLM & Automation Software
https://www.brainasoft.com/braina/ (dostopano: 18. 11. 2025)

[12] voiceattack.com | VoiceAttack Plugin Overview
https://voiceattack.com/VoiceAttackHelp.pdf#:~:text=match%20at%20L89%20Windows%20Speech,If (dostopano: 18. 11. 2025)

[13] learn.microsoft.com| Windows Presentation Foundation Overview
https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/ (dostopano: 18. 11. 2025)

[14] whathifi.com | Spotify Connect: What is it? How can you get it?
https://www.whathifi.com/advice/spotify-connect-what-it-how-can-you-get-it (16. 10. 2025)

[15] nvidia.com | Project G-Assist | AI Assistant – NVIDIA
https://www.nvidia.com/en-us/software/nvidia-app/g-assist/ (dostopano: 18. 11. 2025)

[16] theverge.com | Nvidia's AI assistant on Windows now has plugins for Spotify, Twitch, and more
https://www.theverge.com/news/654221/nvidia-g-assist-plugins-spotify-twitch-apps (23. 4. 2025)

[17] support.spotify.com | Siri and Spotify – Spotify
https://support.spotify.com/us/article/siri-and-spotify/ (dostopano: 18. 11. 2025)

[18] samsung.com | Have Bixby play music on your Galaxy phone
https://www.samsung.com/us/support/answer/ANS10001391/#:~:text=3,Next%20song (dostopano: 18. 11. 2025)

[19] support.spotify.com | Spotify on Alexa devices – Spotify
https://support.spotify.com/us/article/spotify-on-alexa-devices/ (dostopano: 18. 11. 2025

[20] videosdk.live| Vosk Speech Recognition
https://www.videosdk.live/developer-hub/stt/vosk-speech-recognition (1. 1. 2025)

[21] www.openhab.org| Vosk Speech-to-Text
https://www.openhab.org/addons/voice/voskstt/ (dostopano: 12. 11. 2025)

[22] www. earn.microsoft.com| Visual Studio release notes
https://learn.microsoft.com/en-us/visualstudio/releases/2022/release-notes (dostopano: 12. 11. 2025)

[23] www. voicemacro.net|Voice macro
https://www.voicemacro.net/screenshots/(23. 1. 2026)

[24] www.theverge.com|Spotify is finally getting Siri support with iOS 13
https://www.theverge.com/2019/9/27/20886783/spotify-siri-integration-support-ios-13-beta-launch-airpods(27. 9. 2019)

[25] www.dotnet.microsoft.com/|C#
https://dotnet.microsoft.com/en-us/languages/csharp(15. 1. 2026)

[26] www. learn.microsoft.com/|XAML overview
https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/(15. 1. 2026)

[27] www. en.wikipedia.org/|Figma
https://en.wikipedia.org/wiki/Figma(15. 1. 2026)

[28] www.apple.com/siri| Siri
https://www.apple.com/siri/(15. 1. 2026)

[29] developer.amazon.com/|What Is Alexa?
https://developer.amazon.com/en-GB/alexa(15. 1. 2026)

PRILOGE

Priloga 1: Izjava avtorjev

Izjavljamo, da smo pri pripravi raziskovalne naloge upoštevali etična načela in smernice v skladu z veljavnimi pravnimi akti raziskovalnega področja.

Podpisani: Tej Verbič (avtor), Gregor Hrastnik (mentor)

Avtor:

Mentor: